Dense and Low-Rank Gaussian CRFs Using Deep Embeddings

Siddhartha Chandra[1]
siddhartha.chandra@inria.fr
[1] INRIA GALEN, CentraleSupélec

Nicolas Usunier[2]
usunier@fb.com

Iasonas Kokkinos[2]
iasonask@fb.com
[2] Facebook AI Research, Paris

Abstract

In this work we introduce a structured prediction model that endows the Deep Gaussian Conditional Random Field (G-CRF) with a densely connected *graph structure. We keep memory and computational complexity under control by expressing the pairwise interactions as inner products of low-dimensional, learnable embeddings. The G-CRF system matrix is therefore low-rank, allowing us to solve the resulting system in a few milliseconds on the GPU by using conjugate gradient. As in G-CRF, inference is exact, the unary and pairwise terms are jointly trained end-to-end by using analytic expressions for the gradients, while we also develop even faster, Potts-type variants of our embeddings.*

We show that the learned embeddings capture pixel-to-pixel affinities in a task-specific manner, while our approach achieves state of the art results on three challenging benchmarks, namely semantic segmentation, human part segmentation, and saliency estimation. Our implementation is fully GPU based, built on top of the Caffe library, and is available at https://github.com/siddharthachandra/gcrf-v2.0.

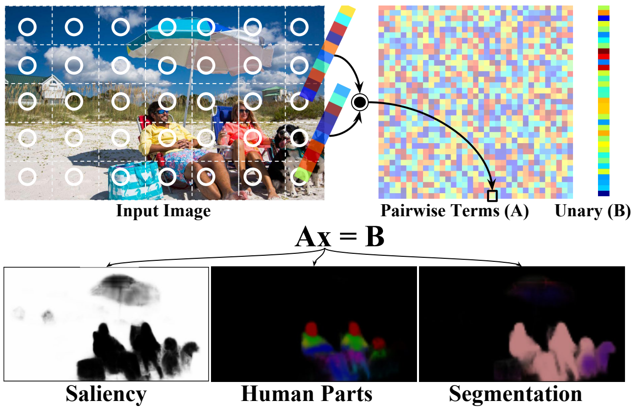

Figure 1. Method overview: each image patch amounts to a node in our fully-connected graph structure. As in the G-CRF model, we infer the prediction $\mathbf{x}$ by solving a system of linear equations $A\mathbf{x} = B$, based on CNN-based unary (B) and pairwise (A) terms. We express pairwise terms as dot products of low-dimensional embeddings $(A_{i,j} = \langle \mathcal{A}_i, \mathcal{A}_j \rangle)$, delivered by a devoted sub-network. This ensures that A is low-rank, allowing for efficient, conjugate gradient-based solutions. The embeddings are optimized in a task-specific manner through end-to-end training.

1. Introduction

Structured prediction combined with deep learning has delivered excellent results on a variety of computer vision benchmarks [2, 5, 7, 8, 33, 34, 40]. Deeplab [5, 7] successfully exploited the Dense-CRF [20] framework, allowing a CNN trained for semantic segmentation to refine object boundaries while compensating for the effects of spatial downsampling within the network. Several works extended this approach to allow for (a) end-to-end training (b) learning of pairwise interaction terms, and (c) using exact inference (table 1). Regarding (a), [2, 26, 27, 28, 33, 34, 40] showed that structured prediction can be unfolded in time and thus be trained end-to-end with CNNs for both sparsely-connected [33] and fully-connected [26, 28, 34, 40] graphical structures. Regarding (b), [2, 26, 28, 33, 34] learned non-parametric, CNN-based pairwise terms for sparsely-connected CRFs, while [40, 15] back-propagated on the parameters of the bilateral filter-type kernels defining their dense pairwise terms. Regarding (c), [2] showed that efficient exact inference can be used for the sparsely-connected case using conjugate gradient, while [1] showed that for densely-connected graphs with bilateral-type pairwise terms linear system methods can be used for efficient inference and backpropagation.

Our work is the first to combine all of the above advances in the case of densely-connected CRFs: we show that we can train in end-to-end manner densely-connected CRFs with non-parametric pairwise terms, while using efficient and exact inference by relying on linear system methods. For this, we build on [2] which combined these advances for sparsely-connected CRFs and extend it to make the densely-connected case tractable. Figure 1 provides an overview of our approach. As in [2] we perform structured prediction by solving a linear system $A\mathbf{x} = B$, where A

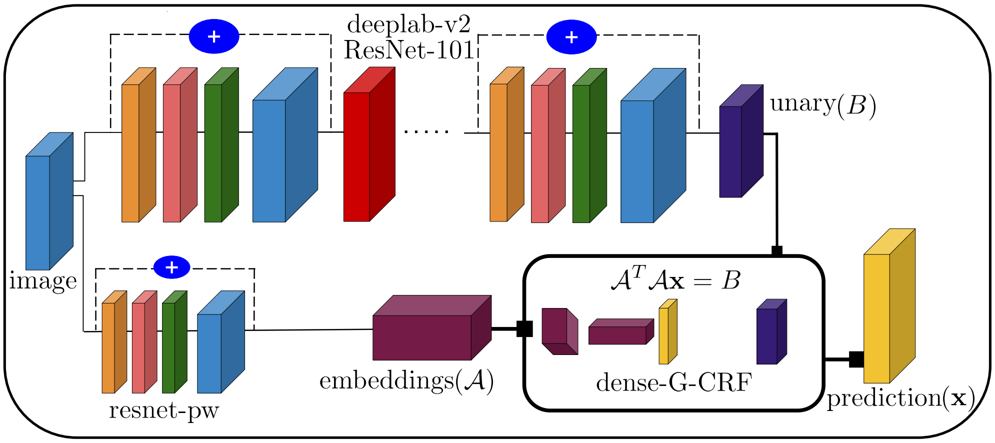

Figure 2. Illustration of our end-to-end trainable, fully convolutional network employing a dense-G-CRF module. We get our unary terms from Deeplab-v2 (we only show one of its three ResNet-101 branches, for simplicity). Our pairwise terms are generated by a parallel sub-network, resnet-pw, which outputs the pixel embeddings of our formulation. The unary terms and pairwise embeddings are combined by our fully connected G-CRF module (dense-G-CRF). This outputs the prediction $\mathbf{x}$ by solving the inference equation $\mathcal{A}^T\mathcal{A}\mathbf{x} = B$.

method	*dense*	*end2end*	*non-parametric*	*exact*
[5, 7]	✓	✗	✗	✗
[40]	✓	✓	✗	✗
[27]	✗	✓	✗	✓
[33]	✗	✓	✓	✗
[34]	✗	✓	✓	✗
[28]	✗	✓	✓	✗
[15]	✗	✓	✓	✗
[2]	✗	✓	✓	✓
[26]	✓	✗	✓	✗
[1]	✓	✓	✗	✓
Ours	✓	✓	✓	✓

Table 1. Comparison of deep structured prediction approaches in terms of whether they accommodate dense connectivity, end-to-end training, use of non-parametric, CNN-based pairwise terms, and exact inference. Our method combines all of these favorable properties.

and B respectively correspond to pairwise and unary terms, delivered by an end-to-end trainable CNN. Solving this system of linear equations results in couplings among all the node variables.

The core development (Sec. 3) consists in replacing the sparse system matrix used to couple the labels of neighboring nodes in [2] with a low-rank matrix that connects any node with all other image nodes through inner products of learnable, D-dimensional embeddings: $A_{i,j} = \langle \mathcal{A}_i, \mathcal{A}_j \rangle$, where $i, j \in \{1, \ldots, N\}$, with N indexing the Cartesian product of pixels and labels. Rather than computing and inverting the full $N \times N$ matrix A, our network only needs to deliver the much smaller $N \times D$ embedding matrix $\mathcal{A}$, which is all that is needed by the conjugate gradient method. Apart from low memory complexity, this can also result in fast conjugate-gradient based structured prediction.

We note that several other works have concurrently explored the use of embeddings in the context of grouping tasks, employing them as a soft, differentiable proxy for cluster assignments [11, 13, 14, 30]. Ours however is the first to make the connection between embeddings, low-rank matrices and densely connected random fields, effectively training embeddings for the propagation of information across the full image domain through the solution of a linear system.

We further exploit the structure of the problem by developing Potts-type embeddings that allow us to reduce the memory complexity by L^2 and computational complexity by a factor of L, where L is the number of classes. The computation time of our fastest method is 0.004s on a GPU for a 321×321 image, 2 orders of magnitude less than GPU-based implementations of Dense-CRF inference, while at the same time achieving higher accuracy across all tasks.

Our approach is *loss-agnostic* and works with arbitrary differentiable losses. As shown in Figures 3 and 4, our embeddings can learn task-specific affinities through end-to-end training. The resulting networks deliver systematic improvements when compared to strong baselines on saliency estimation, human part segmentation, and semantic segmentation.

We first give a brief review of the G-CRF model in Sec. 2, then provide a detailed description of our approach in Sec. 3, and finally demonstrate the merits of our approach on three challenging tasks, namely, semantic segmentation (Sec. 4.1), human part segmentation (Sec. 4.2), and saliency estimation (Sec. 4.3).

2. Deep Gaussian CRF

We briefly describe the Deep Gaussian CRF formulation of [2], following the notation of [2]; further information can be found in [2, 16, 32, 33].

We consider an image I containing P patches where each patch can take a label $l \in \{1, \ldots, L\}$. The predictions are represented as a real-valued vector that gives the score for every patch-label combination, $\mathbf{x} \in \mathbb{R}^N$, where we denote the number of variables in our formulation by $N = P \times L$ for brevity. The L continuous variables associated to every patch can be interpreted as inputs to a softmax function that yields the label posteriors.

In particular, given an image I the G-CRF model defines a joint posterior distribution through a Gaussian multivariate density:

$$p(\mathbf{x}|I) \propto \exp(-\frac{1}{2}\mathbf{x}^T A_I \mathbf{x} + B_I \mathbf{x}),$$

where B_I, A_I denote the unary and pairwise terms respectively, with $B_I \in \mathbb{R}^N$ and $A_I \in \mathbb{R}^{N \times N}$. Dropping the dependence on the image I for simplicity, and assuming a positive-definite matrix A, we see that Maximum-A-Posterior inference amounts to solving the system of linear equations $A\mathbf{x} = B$. For a sparse matrix A, as is the case for grid-structured CRFs, this system can be efficiently solved through the conjugate gradient [31] algorithm.

In [2] the authors drop the probabilistic formulation and treat the G-CRF as a structured prediction layer that is incorporated in a larger network. In the forward pass, the inputs to the layer are A and B, which are delivered by a feed-forward CNN. The output of the layer $\mathbf{x}$ is the solution of the linear system:

$$(A + \lambda \mathbf{I})\mathbf{x} = B, \tag{1}$$

where λ is a positive constant added to the diagonal entries of A to make it positive definite.

In the backward pass, considering that the G-CRF *layer* obtains a gradient for the loss $\mathcal{L}$ with respect to its output $\mathbf{x}$, $\frac{\partial \mathcal{L}}{\partial \mathbf{x}}$, the gradients of the unary terms $\frac{\partial \mathcal{L}}{\partial B}$ can be obtained by solving a new system of linear equations:

$$(A + \lambda \mathbf{I})\frac{\partial \mathcal{L}}{\partial B} = \frac{\partial \mathcal{L}}{\partial \mathbf{x}}, \tag{2}$$

while the gradients of the pairwise terms $\frac{\partial \mathcal{L}}{\partial A}$ are given by:

$$\frac{\partial \mathcal{L}}{\partial A} = -\frac{\partial \mathcal{L}}{\partial B} \otimes \mathbf{x}, \tag{3}$$

where $\otimes$ denotes the Kronecker product operator.

3. Deep-and-Dense Gaussian-CRF

3.1. Low-Rank G-CRF through Embeddings

While the Deep G-CRF model described above allows for efficient and exact inference, in practice it only captures interactions in small ($4-$,$8-$ and $12-$connected) neighborhoods. The model may thereby lose some of its power by ignoring a richer set of long-range interactions. The extension to fully-connected graphs is technically challenging because of the non-sparse matrix A it involves. Assuming an image size of 800×800 pixels, 21 labels (PASCAL VOC benchmark), and a network with a spatial downsampling factor of 8 [5, 6], the number of variables is $N = (100 \times 100) \times 21$ and the number of elements in A would be $N^2 \sim 10^{10}$. This is prohibitively large due to both memory and computational requirements.

To overcome this challenge, we advocate forcing A to be a *low-rank*. In particular, we propose decomposing the $N \times N$ matrix A into a product of the form

$$A = \mathcal{A}^T \mathcal{A}, \tag{4}$$

where $\mathcal{A}$ is a $D \times N$ matrix associating every pixel-label combination with a D-dimensional vector ('embedding'), where $D << N$. This amounts to expressing the pairwise terms for every pair of pixels and labels in the label set as the inner product of their respective embeddings, as follows:

$$A_{p_i,p_j}(l_m, l_n) = \langle \mathcal{A}_{p_i}^{l_m}, \mathcal{A}_{p_j}^{l_n} \rangle,$$

where $i, j \in \{1, \ldots, P\}$ and $m, n \in \{1, \ldots, L\}$.

Since A is symmetric and positive semi definite by design, $A' = \mathcal{A}^T \mathcal{A} + \lambda \mathbf{I}$ is positive definite for any $\lambda > 0$, unlike the case of [2], where λ had to be set empirically.

Adapting the development leading to Eq. 1, we see that we now have to solve the system:

$$(\mathcal{A}^T \mathcal{A} + \lambda \mathbf{I})\mathbf{x} = B. \tag{5}$$

We take advantage of the positive definiteness of A' and use the conjugate gradient method [31] for solving the system of linear equations iteratively.

Setting D allows us to control both the memory and the computational complexity of inference: solving the linear system with conjugate gradient only requires keeping $\mathcal{A}$ in memory and forming inner products between $\mathcal{A}$ and a vector. As such we have a way of trading-off accuracy with speed and memory demands; as indicated in our experiments, with a sufficiently low embedding dimension we obtain excellent results.

3.2. Gradients of the dense G-CRF parameters

We now turn to learning the model parameters via end-to-end network training. To achieve this we require derivatives of the overall loss $\mathcal{L}$ with respect to the model parameters, namely $\frac{\partial \mathcal{L}}{\partial \mathcal{A}}$ and $\frac{\partial \mathcal{L}}{\partial B}$. As described in Eq. 5, we have an analytical closed form relationship between our model parameters $\mathcal{A}$,B, and the prediction $\mathbf{x}$. Therefore, by applying the chain rule of differentiation, we can analytically express the gradients of the model parameters in terms of

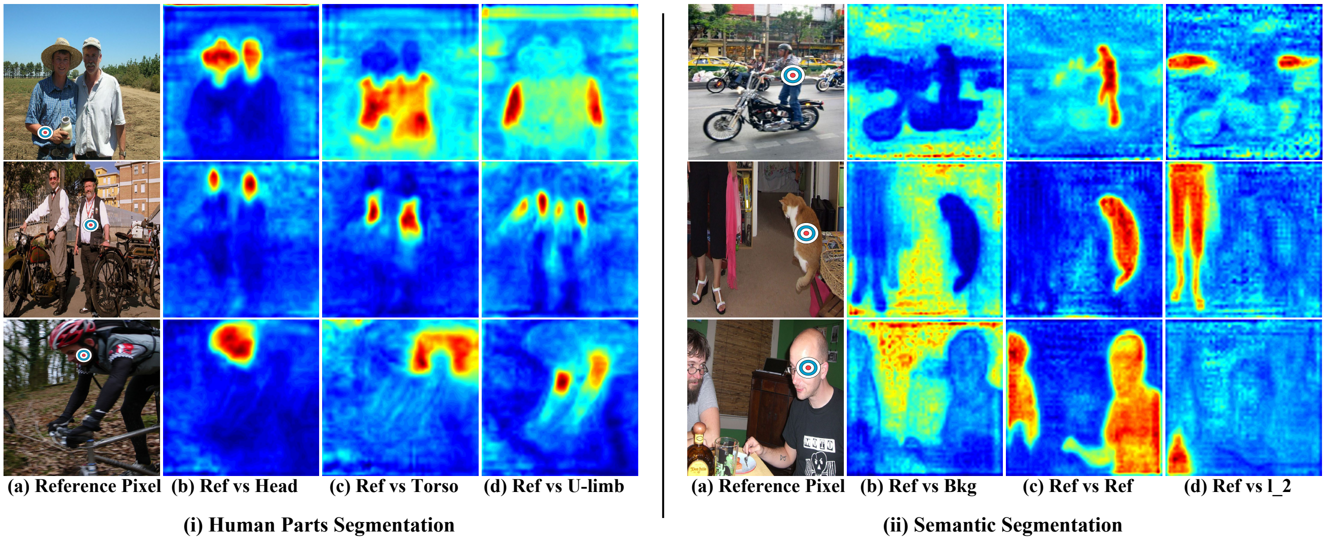

Figure 3. Visualization of pairwise terms obtained by our G-CRF embeddings trained for the (i) human part segmentation, and (ii) semantic segmentation tasks. Column (a) shows the reference pixel (p^*), marked with a dartboard, on the image. The pairwise term corresponding to p^* taking the ground truth label l^* and any other pixel p taking the label l is given by the inner product $A_{p^*,p}\left(l^*,l\right)=\langle\mathcal{A}_p^l,\mathcal{A}_{p^*}^{l^*}\rangle$. In (i) we show the pairwise terms $A_{p^*,p}\left(l^*,\textit{head}\right)$ in (b), $A_{p^*,p}\left(l^*,\textit{torso}\right)$ in (c), and $A_{p^*,p}\left(l^*,\textit{upper-limb}\right)$ in (d). In (ii) we show the pairwise terms $A_{p^*,p}\left(l^*,\textit{bkg}\right)$ in (b), $A_{p^*,p}\left(l^*,l^*\right)$ in (c), and $A_{p^*,p}\left(l^*,l_2\right)$ in (d), where l_2 is the most dominant class in the image besides l^*.

the gradients of the prediction. The gradients of the prediction are delivered by the neural network layer on top of our dense-G-CRF module through backpropagation.

The gradients of the unary terms are straightforward to obtain by substituting Eq. 4 in Eq. 2 as:

$$(\mathcal{A}^T\mathcal{A}+\lambda\mathbf{I})\frac{\partial\mathcal{L}}{\partial B}=\frac{\partial\mathcal{L}}{\partial\mathbf{x}}. \tag{6}$$

We thus obtain the gradients of the unary terms by solving a system of linear equations.

Turning to the gradients of the pixel embeddings, $\mathcal{A}$, we use the chain rule of differentiation as follows:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{A}}=\left(\frac{\partial\mathcal{L}}{\partial A}\right)\left(\frac{\partial A}{\partial\mathcal{A}}\right)=\left(\frac{\partial\mathcal{L}}{\partial A}\right)\left(\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A}\right). \tag{7}$$

We know the expression for $\frac{\partial\mathcal{L}}{\partial A}$ from Eq. 3, but to obtain the expression for $\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A}$ we need to follow some more tedious steps. As in [10], we define a permutation matrix $T_{m,n}$ of size $mn\times mn$ as follows:

$$T_{m,n}\text{vec}(M)=\text{vec}(M^T), \tag{8}$$

where $\text{vec}(M)$ is the vectorization operator that vectorizes a matrix M by stacking its columns. When premultiplied with another matrix, $T_{m,n}$ rearranges the ordering of rows of that matrix, while when postmultiplied with another matrix, $T_{m,n}$ rearranges its columns. Using this matrix, we can form the following expression [10]:

$$\frac{\partial}{\partial\mathcal{A}}\mathcal{A}^T\mathcal{A}=\left(\mathbf{I}\otimes\mathcal{A}^T\right)+\left(\mathcal{A}^T\otimes\mathbf{I}\right)T_{D,N}, \tag{9}$$

where $\mathbf{I}$ is the $N\times N$ identity matrix. Substituting Eq. 3 and Eq. 9 into Eq. 7, we obtain:

$$\frac{\partial\mathcal{L}}{\partial\mathcal{A}}=-\left(\frac{\partial\mathcal{L}}{\partial B}\otimes\mathbf{x}\right)\left(\left(\mathbf{I}\otimes\mathcal{A}^T\right)+\left(\mathcal{A}^T\otimes\mathbf{I}\right)T_{D,N}\right). \tag{10}$$

Despite the apparently complex form, this final expression is particularly simple to implement.

These equations allow us to train embeddings in a task-specific manner, capturing the patch-to-patch affinities that are desirable for a particular structured prediction task. We visualize the affinities learned by our embeddings in Fig. 3 - we observe that our embeddings indeed learn to group pixels in a way that is dictated by the task: on the left pixels belonging to similar human parts are grouped together, while on the right this is done for patches belonging to similar object classes. Similar results can also be seen in Fig. 4 for the more compact embeddings described below.

3.3. Potts Type G-CRF Pixel Embeddings

We now describe *class-agnostic* G-CRF pixel embeddings, which simplify and accelerate the G-CRF model by sharing the pairwise terms between pairs of classes. More specifically, these *Potts*-type embeddings compose pairwise terms between a pair of pixels that depend only on whether they take the same label or not, and are invariant to the particular labels they take. As in [2] we denote by $A_{p_i,p_j}(l_i,l_j)$ the pairwise energy term for pixel p_i taking the label l_i, and pixel p_j taking the label l_j. The *Potts*-type embeddings describe the following model:

$$A_{p_i,p_j}(l_i,l_j) = \left\{ \begin{array}{ll} 0 & l_i = l_j \\ A_{p_i,p_j} & l_i \neq l_j. \end{array} \right\} \qquad (11)$$

The model in Eq. 11 reduces the size of the embeddings from $P \times L$ to P, and allows for significantly faster inference (Sec. 3.4) since the number of computations are reduced by a factor of L. As demonstrated in Sec. 4, this leads to fewer model parameters and better performance. The *Potts*-type embeddings are realized by posing our inference problem in Eq. 5 as:

$$\begin{bmatrix} \lambda\mathbf{I} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \lambda\mathbf{I} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ & & \vdots & \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \lambda\mathbf{I} \end{bmatrix} \times \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \vdots \\ \mathbf{x}_L \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \\ \vdots \\ \mathbf{b}_L \end{bmatrix} \qquad (12)$$

where $\mathbf{x}_k$, denotes the scores for all the pixels for the class k. The per-class unaries are denoted by $\mathbf{b}_k$, and the embeddings $\hat{\mathcal{A}}$ are shared between all class pairs. In [2] solving this large linear system was reduced to solving $L + 1$ smaller linear systems. We have realized that this is not necessary: (1) the same gain in computation speed can be achieved by adapting the conjugate gradient implementation to this structure and avoiding redundant computations, (2) their proposed decomposition of a positive definite linear system may result into smaller non-positive definite systems. These points are detailed in the following subsection.

3.4. Implementation and Efficiency

We now provide numerical analysis details that will be useful for the reproduction of our method. Our approach is implemented as a layer in *Caffe* [17]. We exploit fast linear algebra routines of the CUDA blas library to efficiently implement the conjugate gradient method.

For these timing comparisons, we use a GTX-1080 GPU. Our general-inference procedure takes 0.029s, and Potts-type inference takes 0.004s on average for the semantic segmentation task (21 labels) for an image patch of size 321×321 pixels downsampled by a factor of 8, and for an embedding dimension of 128. This is an order of magnitude faster than the approximate dense CRF mean-field inference which takes 0.2s on average. The sparse G-CRF, and the Potts-type sparse G-CRF from [2] take 0.021s and 0.003s respectively for the same input size. Thus, our dense inference procedure comes at negligible extra cost compared to the sparse G-CRF.

We now describe our approach to efficiently implement the conjugate gradient method for G-CRF pixel embeddings. We begin by describing the conjugate gradient algorithm in Algorithm 1.

The conjugate gradient algorithm thus relies on computing the matrix-vector product $\mathbf{q} = A\mathbf{p}$ in each iteration (Algorithm 1, `line:10`). This operation is computationally the most expensive step of this method. We now describe how to efficiently compute this quantity for our case.

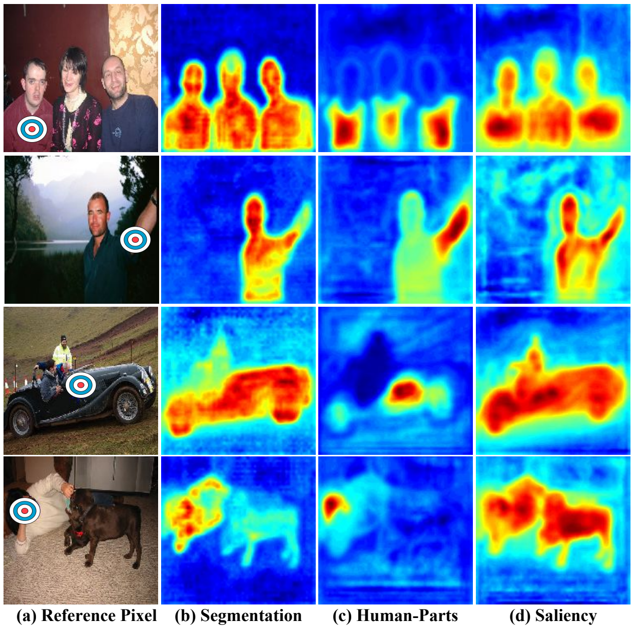
(a) Reference Pixel (b) Segmentation (c) Human-Parts (d) Saliency

Figure 4. Visualization of pairwise terms obtained by our Potts-Type *task-specific* G-CRF embeddings. The first column shows the reference pixel (p^*), marked with a dartboard, on the image. The pairwise term between p^* and any other pixel p is given by the dot product $A_{p^*,p} = \mathcal{A}_p{}^T\mathcal{A}_{p^*}$. We show the pairwise terms $A_{p^*,p}$ for the segmentation task in (b), human part estimation in (c), and saliency estimation in (d).

Algorithm 1 Conjugate Gradient Algorithm

```
1:  procedure ConjugateGradient
2:      Input: $\mathbf{A}$, $\mathbf{B}$, $\mathbf{x}_0$
3:      Output: $\mathbf{x} \mid \mathbf{Ax} = \mathbf{B}$
4:      $\mathbf{r}_0 := \mathbf{B} - \mathbf{A}\mathbf{x}_0$
5:      $\mathbf{p}_0 := \mathbf{r}_0$
6:      $k := 0$
7:      repeat
8:          $\alpha_k := \frac{\mathbf{r}_k^T\mathbf{r}_k}{\mathbf{p}_k^T\mathbf{A}\mathbf{p}_k}$
9:          $\mathbf{x}_{k+1} := \mathbf{x}_k + \alpha_k\mathbf{p}_k$
10:         $\mathbf{r}_{k+1} := \mathbf{r}_k - \alpha_k\mathbf{A}\mathbf{p}_k$
11:         if $r_{k+1}$ is sufficiently small, then exit loop
12:         $\beta_k := \frac{\mathbf{r}_{k+1}^T\mathbf{r}_{k+1}}{\mathbf{r}_k^T\mathbf{r}_k}$
13:         $\mathbf{p}_{k+1} := \mathbf{r}_{k+1} + \beta_k\mathbf{p}_k$
14:         $k := k + 1$
15:     end repeat
16:     $\mathbf{x} = \mathbf{x}_{k+1}$
```

Conjugate Gradient for G-CRF Embeddings To solve Eq. 5, each iteration of the conjugate gradient algorithm involves computing $\mathbf{q} = (\mathcal{A}^T\mathcal{A}+\lambda\mathbf{I})\mathbf{p}$. Explicitly computing $(\mathcal{A}^T\mathcal{A}+\lambda\mathbf{I})$ is unnecessary because (a) it requires us to keep

$PL \times PL$ terms in memory, and (b) it is computationally expensive. We therefore compute $\mathbf{q}$ as

$$\bar{\mathbf{q}} = \mathcal{A}\mathbf{p}; \quad \mathbf{q} = \mathcal{A}^T \bar{\mathbf{q}} + \lambda \mathbf{p}. \tag{13}$$

Conjugate Gradient for Potts-type G-CRF Embeddings The recurring matrix-vector product for this case is given by

$$\mathbf{q} = \begin{bmatrix} \mathbf{q}_1 \\ \mathbf{q}_2 \\ \vdots \\ \mathbf{q}_L \end{bmatrix} = \begin{bmatrix} \lambda \mathbf{I} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \lambda \mathbf{I} & \cdots & \hat{\mathcal{A}}^T\hat{\mathcal{A}} \\ & & \vdots & \\ \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \hat{\mathcal{A}}^T\hat{\mathcal{A}} & \cdots & \lambda \mathbf{I} \end{bmatrix} \begin{bmatrix} \mathbf{p}_1 \\ \mathbf{p}_2 \\ \vdots \\ \mathbf{p}_L \end{bmatrix}. \tag{14}$$

We make two observations by carefully examining Eq. 14:

(1) The terms $\hat{\mathcal{A}}^T\hat{\mathcal{A}}$ are repeated $L-1$ times per column of the precision matrix. A naive implementation would compute $(\hat{\mathcal{A}}^T\hat{\mathcal{A}})\mathbf{p}_k$ exactly $L-1$ times for each class k.

(2) Each $\mathbf{q}_k$ can be computed as a sum of L terms, and for each pair $(\mathbf{q}_k, \mathbf{q}_{k' \neq k})$, $L-2$ of these terms are equal.

Using these observations, and further simplifications, we compute $\mathbf{q}_k$ for each class as

$$\bar{\hat{\mathbf{q}}} = \hat{\mathcal{A}} \sum_{i=1}^{L} \mathbf{p}_i; \quad \hat{\mathbf{q}} = \hat{\mathcal{A}}^T \bar{\hat{\mathbf{q}}} \tag{15}$$

$$\bar{\hat{\mathbf{q}}}_k = \hat{\mathcal{A}}\mathbf{p}_k; \quad \mathbf{q}_k = \hat{\mathbf{q}} + \lambda \mathbf{p}_k - \hat{\mathcal{A}}^T \bar{\hat{\mathbf{q}}}_k \tag{16}$$

Please note that the quantity $\hat{\mathbf{q}}$ in Eq. 15 is computed once, and used to compute $\mathbf{q}_k$ for each class using Eq. 16.

4. Experiments and Results

Base network. Our base network is Deeplab-v2-resnet-101 [6], a three branch multi-resolution network which processes the input image at scale factors of $1, 0.75, 0.5$ and then combines the network responses by upsampling the lower scales and taking an element-wise maximum. It uses random horizontal flipping, and random scaling of the input image to achieve data augmentation.

Fully-Connected G-CRF network. Our fully-connected G-CRF (dense-G-CRF) network is shown in Fig. 2. It uses the base network to provide unaries, and a sub-network (resnet-pw) in parallel to the base network to construct the pixel embeddings for the pairwise terms. As dictated by our experiments in Sec. 4.1 the resnet-pw has layers conv1 through res4a. We use a $3-$phase training strategy. We first train the unary network without the pairwise stream. We train the pairwise sub-network next, with the softmax cross-entropy loss to enforce the following objective: $A_{p_1,p_2}(l_1, l_2) < A_{p_1,p_2}(l'_1 \neq l_1, l'_2 \neq l_2)$, where l_1, l_2 are the ground truth labels for pixels p_1, p_2. Finally, we combine the unary and pairwise networks, and train them together in end-to-end fashion. Each training phase uses 20K iterations with a batch size of 10. The initial learning rate for the first two phases is fixed to 0.001, while for the third phase we set it to $2.5e^{-4}$. We use a polynomial decaying learning rate with power$=0.9$. Training each network takes around 2.5 days on a GTX-1080 GPU.

4.1. Semantic Segmentation

Dataset. We use the PASCAL VOC 2012 dataset which has 1464 training, 1449 validation and 1456 test images containing 20 foreground object classes. We also use the additional ground-truth from [12], obtaining 10582 training images in total. The evaluation criterion is the mean pixel intersection-over-union (IOU) metric.

Ablation Studies. In these experiments, we train on the train set, and evaluate on the val set. We study the effect of varying the depth of the pairwise network stream by chopping the resnet-101 at three lengths, indicated by the standard resnet layer names. We also study the effect of changing the size of G-CRF pixel-embeddings. These results are reported in table 2. The best results are obtained at embedding size of 128 and 1024 for general- and Potts-type embeddings respectively. Results improve as we increase the depth of resnet-pw. Even though the Potts-type embeddings are higher dimensional than the general embeddings, we learn less than half the parameters ($128 \times 21 = 2688 > 1024$). Improvement over the base-network is 0.91%.

Base network [6]	**76.30**			
dense-G-CRF	Embedding Dimension →			
resnet-pw size ↓	64	128	256	512
res2a	76.79	76.81	76.80	76.80
res3a	76.98	76.85	76.84	76.71
res4a	76.95	**77.05**	76.95	76.97
densepotts-G-CRF	Embedding Dimension →			
resnet-pw size ↓	256	512	1024	2048
res2a	76.95	76.86	77.10	76.82
res3a	76.98	76.86	77.15	76.85
res4a	76.99	77.10	**77.21**	76.92

Table 2. Ablation study- mean Intersection Over Union (IOU) accuracy on PASCAL VOC 2012 validation set. We compare the performance of our method against that of the base network, and study the effect of varying the depth of the pairwise stream network, and the size of pixel embeddings.

Performance on test set. We now compare our approach with the base network [6], the base network with the sparse deep G-CRF from [2], as well as other leading approaches on this benchmark. In these experiments, we train with the train and val sets, and evaluate performance on the test set. In all of the following sections we use our best configurations from table 2.

Baselines. The mainstream approach on this task is to use fully convolutional networks [5, 6, 29] trained with the Softmax cross-entropy loss. For this task, we compare our approach with the state of the art methods on this benchmark. The baselines include (a) the CRF as RNN network [40], (b) the Deeplab+Boundary network [18] which exploits an edge detection detection network to boost the performance of the Deeplab network, (c) the Adelaide Context network [26], (d) the deep parsing network [28], (e) the Deeplab-v2 base network [6] and (f) the sparse-G-CRF network [2] which combines the Deeplab-v2 network with sparse, Potts-type pairwise terms.

We report the results in table 3. With our dense-Potts embeddings, we get an improvement of 0.8% over the sparse deep G-CRF approach, and 1.3% over the base network. We get a 0.1% boost in performance when we train our dense-Potts model with the sparse G-CRF from [2] (the output after dense-GCRF inference is fed as input to the sparse-GCRF inference module). Qualitative results are shown in Fig. 5. We note that performances of two recent deep-architectures namely PSPNet [38] and Deeplab-v3 [3] are significantly better than those of our baseline and other competing approaches. However, the authors of these works have not yet released their training pipelines publicly. We expect similar improvements by using our approach on these networks. We will experiment with these networks once their training pipelines are made available.

Method	mean IoU
CRFRNN [40]	74.7
Deeplab Multi-Scale + CRF [18]	74.8
Adelaide Context [26]	77.8
Deep Parsing Network [28]	77.4
Deeplab V2 (base network) [6]	79.0
Deeplab V2 + CRF [6]	79.7
sparsepotts-G-CRF [2]	79.5
dense-G-CRF (Ours)	80.1
densepotts-G-CRF (Ours)	80.3
densepotts+sparsepotts-G-CRF (Ours)	**80.4**

Table 3. Semantic segmentation - mean Intersection Over Union (IOU) accuracy on PASCAL VOC 2012 test.

4.2. Human part Segmentation

Dataset. We use the PASCAL Person Parts dataset [9]. As in [24], we merge the annotations to obtain six person part classes, namely the head, torso, upper arms, lower arms, upper legs, and lower legs. This dataset has 1716 train images and 1817 test images. The evaluation criterion is the mean pixel intersection-over-union (IOU) metric.

Baselines. The state of the art approaches on human part segmentation also use fully convolutional networks, sometimes additionally exploiting Long Short Term Memory Units [24, 25]. For this task, we compare our approach to the following methods: (a) the Deeplab attention to scale network [4], (b) the Auto Zoom network [37], (c) the Local Global LSTM network [25] which combines local and global cues via LSTM units, (d) the Graph LSTM network [24], (e) the base network with and without dense CRF post-processing, and (f) the sparse G-CRF Potts model.

We report the results in table 4. While the previous state of the art approach Deeplab-v2 achieves 64.94 with dense-CRF post-processing, out Potts-type model outperforms it by 1.33% mean IoU without using dense-CRF post-processing. Additionally, we outperform the Deeplab-V2 G-CRF Potts baseline from [2] by 1.06%. Using the sparse-G-CRF on top of our results gives us a minor boost of 0.04%. We show qualitative results in Fig. 5.

Attention [4]	56.39
Auto Zoom [37]	57.54
LG-LSTM [25]	57.97
Graph LSTM [24]	60.16
Deeplab-V2 [6]	64.40
Deeplab-V2-CRF [6]	64.94
sparsepotts-G-CRF [2]	65.21
dense-G-CRF (Ours)	66.10
densepotts-G-CRF (Ours)	66.27
densepotts+sparsepotts-G-CRF (Ours)	**66.31**

Table 4. Part segmentation - mean Intersection-Over-Union accuracy on the PASCAL Parts dataset of [9].

4.3. Saliency Estimation

Datasets. As in [19], we use the MSRA-10K saliency dataset [35] for training, and evaluate our performance on the PASCAL-S [23], and the HKU-IS [21] datasets. The

Method	PASCAL-S	HKU-IS
LEGS [36]	0.752	0.770
MC [39]	0.740	0.798
MDF [21]	0.764	0.861
FCN [22]	0.793	0.867
DCL [22]	0.815	0.892
DCL + CRF [22]	0.822	0.904
Ubernet 1-Task [19]	0.835	-
Deeplab-v2 [6]	0.859	0.916
sparse-G-CRF [2]	0.861	0.914
dense-G-CRF (Ours)	**0.872**	**0.927**
dense+sparse-G-CRF (Ours)	0.864	0.927

Table 5. Saliency estimation results: we report the Maximal F-measure (MF) on the PASCAL Saliency dataset of [23], and the HKU-IS dataset of [21].

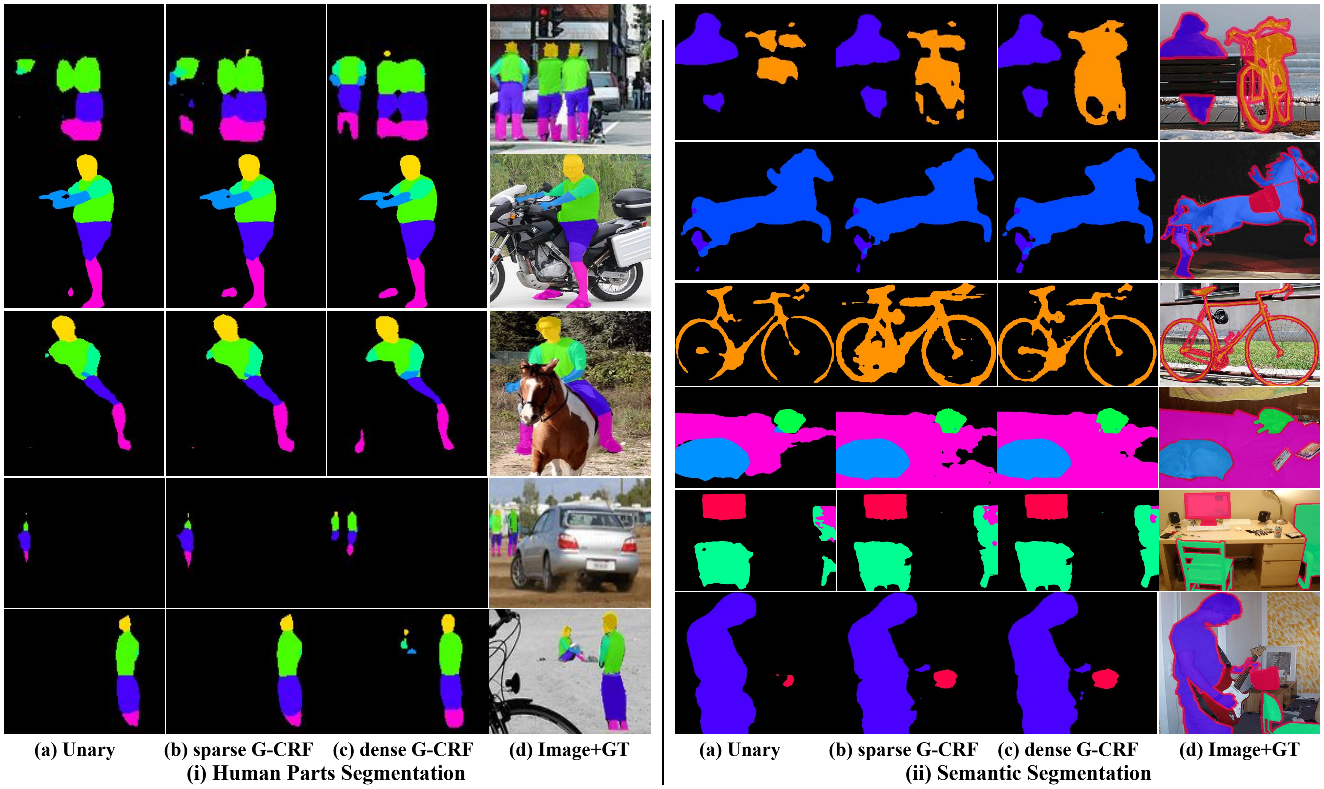

Figure 5. Qualitative Results of (i) Part Segmentation, and (ii) Semantic Segmentation tasks. (a) shows the unary network output, (b) shows the sparsepotts-G-CRF output, (c) shows the densepotts-G-CRF output, and (d) shows the input image and ground truth. In (i), our fully connected model recovers false negatives (rows 1,4), and missing parts (the right foot in rows 2,3), and prevents propagation of erroneous labels (a patch labeled face is eliminated from the right foot in row 5). In (ii), this information flow from the rest of the image helps recover missing object parts (cycle in rows 1,3, person's leg in row 2, table in row 4, sheep's leg in row 5, and left hand in row 7.)

MSRA-10K dataset contains 10000 images with annotated pixel-wise segmentation masks for salient objects. The Pascal-S saliency dataset contains pixel-wise saliency for 850 images. The HKU-IS dataset has 4447 images, with multiple salient objects in each image. The evaluation criterion is the maximal F-Measure as in [19, 23].

Baselines. Our baselines for the saliency estimation task include (a) the Local Estimation and Global Search (LEGS) framework [36], (b) the multi-context network [39], (c) the multiscale deep features network [21], (d) the deep contrast learning networks [22] which proposes a network structure that better exploits object boundaries to improve saliency estimation and additionally uses a fully connected CRF model, (e) the Ubernet architecture [19] which demonstrates that sharing parameters for mutually symbiotic tasks can help improve overall performance of these tasks, (f) our base network, i.e. Deeplab-v2, and (g) the sparse G-CRF Potts model alongside the base network.

Results are tabulated in table 5. Our method significantly outperforms the competing methods on both datasets. Additionally, we do not obtain improvements when combining our method with the sparse G-CRF approach.

5. Conclusions and Future Work

In this work we propose a fully-connected G-CRF model for end-to-end training of deep architectures. We propose strategies for efficient implementation and show that inference over a fully-connected graph comes with neglegible computational overhead compared to a sparsely connected graph. Our experimental evaluation indicates consistent improvements over the state of the art approaches on three challenging public benchmarks for semantic segmentation, human part segmentation and saliency estimation. Future work would involve exploiting this framework on other dense labeling and regression tasks such as depth estimation, image denoising and estimation of surface normals, which can be naturally handled by our model owing to its continuous nature. Further, we would also like to exploit G-CRF embeddings for dense-labeling tasks such as semantic/instance segmentation and optical flow estimation in videos. In the case of videos, we would like to capture not only the spatial context but temporal context as well by expressing temporal pairwise terms between two frames via dot products of embeddings computed on them.

References

[1] J. T. Barron and B. Poole. The fast bilateral solver. In *ECCV*, 2016. 1, 2

[2] S. Chandra and I. Kokkinos. Fast, exact and multi-scale inference for semantic image segmentation with deep gaussian crfs. In *ECCV*, 2016. 1, 2, 3, 4, 5, 6, 7

[3] L. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. *CoRR*, abs/1706.05587, 2017. 7

[4] L. Chen, Y. Yang, J. Wang, W. Xu, and A. L. Yuille. Attention to scale: Scale-aware semantic image segmentation. *CVPR*, 2016. 7

[5] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Semantic image segmentation with deep convolutional nets and fully connected crfs. *arXiv preprint arXiv:1412.7062*, 2014. 1, 2, 3, 7

[6] 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:1606.00915*, 2016. 3, 6, 7

[7] L.-C. Chen, G. Papandreou, K. Murphy, and A. L. Yuille. Weakly- and semi-supervised learning of a deep convolutional network for semantic image segmentation. *ICCV*, 2015. 1, 2

[8] L.-C. Chen, A. G. Schwing, A. L. Yuille, and R. Urtasun. Learning Deep Structured Models. In *ICML*, 2015. 1

[9] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 7

[10] P. L. Fackler. Notes on matrix calculus. 4

[11] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *CoRR*, abs/1703.10277, 2017. 2

[12] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Hypercolumns for object segmentation and fine-grained localization. In *CVPR*, 2015. 6

[13] A. Harley, K. Derpanis, and I. Kokkinos. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015. 2

[14] A. W. Harley, K. G. Derpanis, and I. Kokkinos. Learning dense convolutional embeddings for semantic segmentation. *CoRR*, abs/1511.04377, 2015. 2

[15] V. Jampani, M. Kiefel, and P. V. Gehler. Learning sparse high dimensional filters: Image filtering, dense crfs and bilateral neural networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4452–4461, 2016. 1, 2

[16] J. Jancsary, S. Nowozin, T. Sharp, and C. Rother. Regression tree fields - an efficient, non-parametric approach to image labeling problems. In *CVPR*, 2012. 3

[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. 5

[18] I. Kokkinos. Pushing the Boundaries of Boundary Detection using Deep Learning. In *ICLR*, 2016. 7

[19] I. Kokkinos. Ubernet: A universal cnn for the joint treatment of low-, mid-, and high- level vision problems. In *POCV workshop*, 2016. 7, 8

[20] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 1

[21] G. Li and Y. Yu. Visual saliency based on multiscale deep features. In *CVPR*, 2015. 7, 8

[22] G. Li and Y. Yu. Deep contrast learning for salient object detection. In *CVPR*, 2016. 7, 8

[23] Y. Li, X. Hou, C. Koch, J. M. Rehg, and A. L. Yuille. The secrets of salient object segmentation. In *CVPR*, 2014. 7, 8

[24] X. Liang, X. Shen, J. Feng, L. Liang, and S. Yan. Semantic object parsing with graph lstm. In *arXiv preprint arXiv:1603.07063*, 2016. 7

[25] X. Liang, X. Shen, D. Xiang, J. Feng, L. Lin, and S. Yan. Semantic object parsing with local-global long short-term memory. In *CVPR*, 2016. 7

[26] G. Lin, C. Shen, I. D. Reid, and A. van den Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *CVPR*, 2016. 1, 2, 7

[27] F. Liu, C. Shen, , and G. Lin. Deep convolutional neural fields for depth estimation from a single image. In *CVPR*, 2015. 1, 2

[28] Z. Liu, X. Li, P. Luo, C.-C. Loy, and X. Tang. Semantic image segmentation via deep parsing network. In *CVPR*, pages 1377–1385, 2015. 1, 2, 7

[29] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, pages 3431–3440, 2015. 7

[30] A. Newell and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. *CoRR*, abs/1611.05424, 2016. 2

[31] J. R. Shewchuk. An introduction to the conjugate gradient method without the agonizing pain. 3

[32] M. F. Tappen, C. Liu, E. H. Adelson, and W. T. Freeman. Learning gaussian conditional random fields for low-level vision. In *CVPR*, 2007. 3

[33] R. Vemulapalli, O. Tuzel, M.-Y. Liu, and R. Chellapa. Gaussian conditional random field network for semantic segmentation. In *CVPR*, June 2016. 1, 2, 3

[34] T.-H. Vu, A. Osokin, and I. Laptev. Context-aware cnns for person head detection. In *ICCV*, pages 2893–2901, 2015. 1, 2

[35] K. Wang, L. Lin, J. Lu, C. Li, and K. Shi. PISA: pixelwise image saliency by aggregating complementary appearance contrast measures with edge-preserving coherence. 2015. 7

[36] L. Wang, H. Lu, X. Ruan, and M. Yang. Deep networks for saliency detection via local estimation and global search. In *CVPR*, 2015. 7, 8

[37] F. Xia, P. Wang, L. Chen, and A. L. Yuille. Zoom better to see clearer: Human part segmentation with auto zoom net. In *ECCV*, 2016. 7

[38] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CoRR*, abs/1612.01105, 2016. 7

[39] R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. In *CVPR*, 2015. 7, 8

[40] 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 *ICCV*, 2015. 1, 2, 7