2015 IEEE International Conference on Computer Vision

Semantic Image Segmentation via Deep Parsing Network

Ziwei Liu* Xiaoxiao Li* Ping Luo Chen Change Loy Xiaoou Tang
Department of Information Engineering, The Chinese University of Hong Kong
{lz013,lx015,pluo,ccloy,xtang}@ie.cuhk.edu.hk

Abstract

This paper addresses semantic image segmentation by incorporating rich information into Markov Random Field (MRF), including high-order relations and mixture of label contexts. Unlike previous works that optimized MRFs using iterative algorithm, we solve MRF by proposing a Convolutional Neural Network (CNN), namely Deep Parsing Network (DPN)[1], which enables deterministic end-to-end computation in a single forward pass. Specifically, DPN extends a contemporary CNN architecture to model unary terms and additional layers are carefully devised to approximate the mean field algorithm (MF) for pairwise terms. It has several appealing properties. First, different from the recent works that combined CNN and MRF, where many iterations of MF were required for each training image during back-propagation, DPN is able to achieve high performance by approximating one iteration of MF. Second, DPN represents various types of pairwise terms, making many existing works as its special cases. Third, DPN makes MF easier to be parallelized and speeded up in Graphical Processing Unit (GPU). DPN is thoroughly evaluated on the PASCAL VOC 2012 dataset, where a single DPN model yields a new state-of-the-art segmentation accuracy of 77.5%.

1. Introduction

Markov Random Field (MRF) or Conditional Random Field (CRF) has achieved great successes in semantic image segmentation, which is one of the most challenging problems in computer vision. Existing works such as [31, 29, 9, 34, 11, 2, 8, 25, 22] can be generally categorized into two groups based on their definitions of the unary and pairwise terms of MRF.

In the first group, researchers improved labeling accuracy by exploring rich information to define the pairwise functions, including long-range dependencies [16, 17], high-order potentials [37, 36], and semantic label contexts [21, 26, 38]. For example, Krähenbühl *et al*. [16] attained accurate segmentation boundary by inferring on a fully-connected graph. Vineet *et al*. [37] extended [16] by defining both high-order and long-range terms between pixels. Global or local semantic contexts between labels were also investigated by [38]. Although they accomplished promising results, they modeled the unary terms as SVM or Adaboost, whose learning capacity becomes a bottleneck. The learning and inference of complex pairwise terms are often expensive.

In the second group, people learned a strong unary classifier by leveraging the recent advances of deep learning, such as the Convolutional Neural Network (CNN). With deep models, these works [23, 24, 25, 22, 3, 28, 39, 30, 19] demonstrated encouraging results using simple definition of the pairwise function or even ignore it. For instance, Long *et al*. [22] transformed fully-connected layers of CNN into convolutional layers, making accurate per-pixel classification possible using the contemporary CNN architectures that were pre-trained on ImageNet [6]. Chen *et al*. [3] improved [22] by feeding the outputs of CNN into a MRF with simple pairwise potentials, but it treated CNN and MRF as separated components. A recent advance was obtained by [30], which jointly trained CNN and MRF by passing the error of MRF inference backward into CNN, but iterative inference of MRF such as the mean field algorithm (MF) [27] is required for each training image during back-propagation (BP). Zheng *et al*. [39] further showed that the procedure of MF inference can be represented as a Recurrent Neural Network (RNN), but their computational costs are similar. We found that directly combing CNN and MRF as above is inefficient, because CNN typically has millions of parameters while MRF infers thousands of latent variables; and even worse, incorporating complex pairwise terms into MRF becomes impractical, limiting the performance of the entire system.

This work proposes a novel Deep Parsing Network (DPN), which is able to jointly train CNN and complex pairwise terms. DPN has several appealing **properties**.

*indicates shared first authorship.

[1]Project page: http://personal.ie.cuhk.edu.hk/~lz013/projects/DPN.html. For more technical details, please contact the corresponding author Ping Luo via pluo.lhi@gmail.com.

1550-5499/15 $31.00 © 2015 IEEE
DOI 10.1109/ICCV.2015.162

(1) DPN solves MRF with a single feed-forward pass, reducing computational cost and meanwhile maintaining high performance. Specifically, DPN models unary terms by extending the VGG-16 network (VGG_{16}) [32] pre-trained on ImageNet, while additional layers are carefully designed to model complex pairwise terms. Learning of these terms is transformed into deterministic end-to-end computation by BP, instead of embedding MF into BP as [30, 19] did. Although MF can be represented by RNN [39], it needs to recurrently compute the forward pass so as to achieve good performance and thus is time-consuming, *e.g.* each forward pass contains hundred thousands of weights. DPN approximates MF by using only one iteration. This is made possible by joint learning strong unary terms and rich pairwise information. (2) Pairwise terms determine the graphical structure. In previous works, if the former is changed, so is the latter as well as its inference procedure. But with DPN, modifying the complexity of pairwise terms, *e.g.* range of pixels and contexts, is as simple as modifying the receptive fields of convolutions, without varying BP. DPN is able to represent multiple types of pairwise terms, making many previous works [3, 39, 30] as its special cases. (3) DPN approximates MF with convolutional and pooling operations, which can be speeded up by low-rank approximation [14] and easily parallelized [4] in a Graphical Processing Unit (GPU).

Our **contributions** are summarized as below. (1) A novel DPN is proposed to jointly train VGG_{16} and rich pairwise information, *i.e. mixture of label contexts* and *high-order relations*. Compared to existing deep models, DPN can approximate MF with only *one iteration*, reducing computational cost but still maintaining high performance. (2) We disclose that DPN represents multiple types of MRFs, making many previous works such as RNN [39] and DeepLab [3] as its special cases. (3) Extensive experiments investigate which component of DPN is crucial to achieve high performance. A single DPN model achieves a new state-of-the-art accuracy of 77.5% on the PASCAL VOC 2012 [7] test set. (4) We analyze the time complexity of DPN on GPU.

2. Our Approach

DPN learns MRF by extending VGG_{16} to model unary terms and additional layers are carefully designed for pairwise terms.

Overview MRF [10] is an undirected graph where each node represents a pixel in an image $\mathbf{I}$, and each edge represents relation between pixels. Each node is associated with a binary latent variable, $y_u^i \in \{0, 1\}$, indicating whether a pixel i has label u. We have $\forall u \in L = \{1, 2, ..., l\}$, representing a set of l labels. The energy function of MRF is written as

$$E(\mathbf{y}) = \sum_{\forall i \in \mathcal{V}} \Phi(y_i^u) + \sum_{\forall i,j \in \mathcal{E}} \Psi(y_i^u, y_j^v), \tag{1}$$

where $\mathbf{y}$, $\mathcal{V}$, and $\mathcal{E}$ denote a set of latent variables, nodes, and edges, respectively. $\Phi(y_i^u)$ is the unary term, measuring the cost of assigning label u to the i-th pixel. For instance, if pixel i belongs to the first category other than the second one, we should have $\Phi(y_i^1) < \Phi(y_i^2)$. Moreover, $\Psi(y_i^u, y_j^v)$ is the pairwise term that measures the penalty of assigning labels u, v to pixels i, j respectively.

Intuitively, the unary terms represent per-pixel classifications, while the pairwise terms represent a set of smoothness constraints. The unary term in Eqn.(1) is typically defined as

$$\Phi(y_i^u) = -\ln p(y_i^u = 1|\mathbf{I}) \tag{2}$$

where $p(y_i^u = 1|\mathbf{I})$ indicates the probability of the presence of label u at pixel i, modeling by VGG_{16}. To simplify discussions, we abbreviate it as p_i^u. The smoothness term can be formulated as

$$\Psi(y_i^u, y_j^v) = \mu(u, v)\mathrm{d}(i, j), \tag{3}$$

where the first term learns the penalty of global co-occurrence between any pair of labels, *e.g.* the output value of $\mu(u, v)$ is large if u and v should not coexist, while the second term calculates the distances between pixels, *e.g.* $\mathrm{d}(i, j) = \omega_1 \|\mathbf{I}_i - \mathbf{I}_j\|^2 + \omega_2 \|[x_i \ y_i] - [x_j \ y_j]\|^2$. Here, $\mathbf{I}_i$ indicates a feature vector such as RGB values extracted from the i-th pixel, x, y denote coordinates of pixels' positions, and ω_1, ω_2 are the constant weights. Eqn.(3) implies that if two pixels are close and look similar, they are encouraged to have labels that are compatible. It has been adopted by most of the recent deep models [3, 39, 30] for semantic image segmentation.

However, Eqn.(3) has two main drawbacks. First, its first term captures the co-occurrence frequency of two labels in the training data, but neglects the spatial context between objects. For example, 'person' may appear beside 'table', but not at its bottom. This spatial context is a mixture of patterns, as different object configurations may appear in different images. Second, it defines only the pairwise relations between pixels, missing their high-order interactions.

To resolve these issues, we define the smoothness term by leveraging rich information between pixels, which is one of the **advantages** of DPN over existing deep models. We have

$$\Psi(y_i^u, y_j^v) = \sum_{k=1}^{K} \lambda_k \mu_k(i, u, j, v) \sum_{\forall z \in \mathcal{N}_j} \mathrm{d}(j, z) p_z^v. \tag{4}$$

The first term in Eqn.(4) learns a **mixture of local label contexts**, penalizing label assignment in a local region,

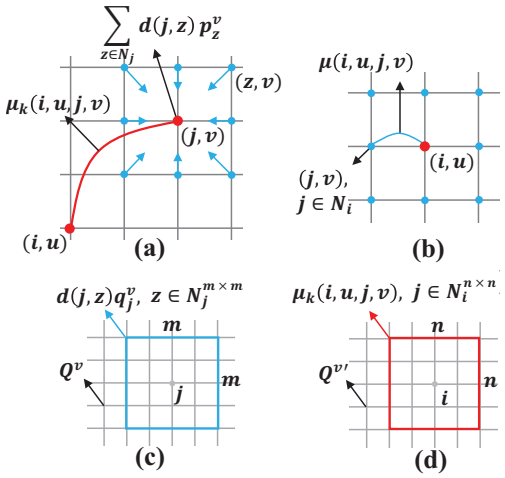

Figure 1: (a) Illustration of the pairwise terms in DPN. (b) explains the label contexts. (c) and (d) show that mean field update of DPN corresponds to convolutions.

where K is the number of components in mixture and λ_k is an indicator, determining which component is activated. We define $\lambda_k \in \{0,1\}$ and $\sum_{k=1}^{K}\lambda_k = 1$. An intuitive illustration is given in Fig.1 (b), where the dots in red and blue represent a center pixel i and its neighboring pixels j, *i.e.* $j \in \mathcal{N}_i$, and (i,u) indicates assigning label u to pixel i. Here, $\mu(i,u,j,v)$ outputs labeling cost between (i,u) and (j,v) with respect to their relative positions. For instance, if u,v represent 'person' and 'table', the learned penalties of positions j that are at the bottom of center i should be large. The second term basically models a **triple penalty**, which involves pixels i, j, and j's neighbors, implying that if (i,u) and (j,v) are compatible, then (i,u) should be also compatible with j's nearby pixels (z,v), $\forall z \in \mathcal{N}_j$, as shown in Fig.1 (a).

Learning parameters (*i.e.* weights of VGG_{16} and costs of label contexts) in Eqn.(1) is to minimize the distances between ground-truth label map and $\mathbf{y}$, which needs to be inferred subject to the smoothness constraints.

Inference Overview Inference of Eqn.(1) can be obtained by the mean field (MF) algorithm [27], which estimates the joint distribution of MRF, $P(\mathbf{y})=\frac{1}{Z}\exp\{-E(\mathbf{y})\}$, by using a fully-factorized proposal distribution, $Q(\mathbf{y}) = \prod_{\forall i\in\mathcal{V}}\prod_{\forall u\in L} q_i^u$, where each q_i^u is a variable we need to estimate, indicating the predicted probability of assigning label u to pixel i. To simplify the discussion, we denote $\Phi(y_i^u)$ and $\Psi(y_i^u, y_j^v)$ as Φ_i^u and Ψ_{ij}^{uv}, respectively. $Q(\mathbf{y})$ is typically optimized by minimizing a free energy function [15] of MRF,

$$F(Q) = \sum_{\forall i\in\mathcal{V}}\sum_{\forall u\in L} q_i^u\Phi_i^u + \sum_{\forall i,j\in\mathcal{E}}\sum_{\forall u\in L}\sum_{\forall v\in L} q_i^u q_j^v \Psi_{ij}^{uv} + \sum_{\forall i\in\mathcal{V}}\sum_{\forall u\in L} q_i^u \ln q_i^u. \quad (5)$$

Specifically, the first term in Eqn.(5) characterizes the cost of each pixel's predictions, while the second term characterizes the consistencies of predictions between pixels. The last term is the entropy, measuring the confidences of predictions. To estimate q_i^u, we differentiate Eqn.(5) with respect to it and equate the resulting expression to zero. We then have a closed-form expression,

$$q_i^u \propto \exp\Big\{-(\Phi_i^u + \sum_{\forall j\in\mathcal{N}_i}\sum_{\forall v\in L} q_j^v\Psi_{ij}^{uv})\Big\}, \quad (6)$$

such that the predictions for each pixel is independently attained by repeating Eqn.(6), which implies whether pixel i have label u is proportional to the estimated probabilities of all its neighboring pixels, weighted by their corresponding smoothness penalties. Substituting Eqn.(4) into (6), we have

$$q_i^u \propto \exp\Big\{-\Phi_i^u - \sum_{k=1}^{K}\lambda_k \sum_{\forall v\in L}\sum_{\forall j\in\mathcal{N}_i} \mu_k(i,u,j,v)\sum_{\forall z\in\mathcal{N}_j} \mathrm{d}(j,z) q_j^v q_z^v\Big\}, \quad (7)$$

where each q_i^u is initialized by the corresponding p_i^u in Eqn.(2), which is the unary prediction of VGG_{16}. Eqn.(7) satisfies the smoothness constraints.

In the following, DPN approximates one iteration of Eqn.(7) by decomposing it into two steps. Let Q^v be a predicted label map of the v-th category. In the first step as shown in Fig.1 (c), we calculate the triple penalty term in (7) by applying a $m\times m$ filter on each position j, where each element of this filter equals $\mathrm{d}(j,z)q_j^v$, resulting in $Q^{v'}$. Apparently, this step smoothes the prediction of pixel j with respect to the distances between it and its neighborhood. In the second step as illustrated in (d), the labeling contexts can be obtained by convolving $Q^{v'}$ with a $n\times n$ filter, each element of which equals $\mu_k(i,u,j,v)$, penalizing the triple relations as shown in (a).

3. Deep Parsing Network

This section describes the implementation of Eq.(7) in a Deep Parsing Network (DPN). DPN extends VGG_{16} as unary term and additional layers are designed to approximate one iteration of MF inference as the pairwise term. The hyper-parameters of VGG_{16} and DPN are compared in Table 1.

VGG_{16} As listed in Table 1 (a), the first row represents the *name* of layer and 'x-y' in the second row represents the *size* of the receptive field and the *stride* of convolution, respectively. For instance, '3-1' in the convolutional layer implies that the receptive field of each filter is 3×3 and it is applied on every single pixel of an input feature map, while '2-2' in the max-pooling layer indicates each feature

(a) **VGG$_{16}$:** 224×224×3 *input image*; 1×1000 *output labels*

	1	2	3	4	5	6	7	8	9	10	11	12
layer	2×**conv**	**max**	2×**conv**	**max**	3×**conv**	**max**	3×**conv**	**max**	3×**conv**	**max**	2×**fc**	**fc**
filter–stride	3-1	2-2	3-1	2-2	3-1	2-2	3-1	2-2	3-1	2-2	-	-
#channel	64	64	128	128	256	256	512	512	512	512	1	1
activation	relu	idn	relu	idn	relu	idn	relu	idn	relu	idn	relu	soft
size	224	112	112	56	56	28	28	14	14	7	4096	1000

(b) **DPN:** 512×512×3 *input image*; 512×512×21 *output label maps*

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
layer	2×**conv**	**max**	2×**conv**	**max**	3×**conv**	**max**	3×**conv**	3×**conv**	**conv**	**conv**	**conv**	**lconv**	**conv**	**bmin**	**sum**
filter–stride	3-1	2-2	3-1	2-2	3-1	2-2	3-1	5-1	25-1	1-1	1-1	50-1	9-1	1-1	1-1
#channel	64	64	128	128	256	256	512	512	4096	4096	21	21	105	21	21
activation	relu	idn	relu	idn	relu	idn	relu	relu	relu	relu	sigm	lin	lin	idn	soft
size	512	256	256	128	128	64	64	64	64	64	512	512	512	512	512

Table 1: The comparisons between the network architectures of VGG$_{16}$ and DPN, as shown in (a) and (b) respectively. Each table contains five rows, representing the '**name of layer**', '**receptive field of filter**'−'**stride**', '**number of output feature maps**', '**activation function**' and '**size of output feature maps**', respectively. Furthermore, '**conv**', '**lconv**','**max**', '**bmin**', '**fc**', and '**sum**' represent the convolution, local convolution, max pooling, block min pooling, fully connection, and summation, respectively. Moreover, 'relu', 'idn', 'soft', 'sigm', and 'lin' represent the activation functions, including rectified linear unit [18], identity, softmax, sigmoid, and linear, respectively.

map is pooled over every other pixel within a 2×2 local region. The last three rows show the number of the output feature maps, activation functions, and the size of output feature maps, respectively. As summarized in Table 1 (a), VGG$_{16}$ contains thirteen convolutional layers, five max-pooling layers, and three fully-connected layers. These layers can be partitioned into twelve groups, each of which covers one or more homogenous layers. For example, the first group comprises two convolutional layers with 3×3 receptive field and 64 output feature maps, each of which is 224×224.

3.1. Modeling Unary Terms

To make full use of VGG$_{16}$, which is pre-trained by ImageNet, we adopt all its parameters to initialize the filters of the first ten groups of DPN. To simplify the discussions, we take PASCAL VOC 2012 (VOC12) [7] as an example. Note that DPN can be easily adapted to any other semantic image segmentation dataset by modifying its hyper-parameters. VOC12 contains 21 categories and each image is rescaled to 512×512 in training. Therefore, DPN needs to predict totally 512×512×21 labels, *i.e.* one label for each pixel. To this end, we extends VGG$_{16}$ in two aspects.

In particular, let ai and bi denote the i-th group in Table 1 (a) and (b), respectively. First, we **increase resolution** of VGG$_{16}$ by removing its max pooling layers at a8 and a10, because most of the information is lost after pooling, *e.g.* a10 reduces the input size by 32 times, *i.e.* from 224×224 to 7×7. As a result, the smallest size of feature map in DPN is 64×64, keeping much more information compared with VGG$_{16}$. Note that the filters of b8 are initialized as the filters of a9, but the 3×3 receptive field is padded into 5×5 as shown in Fig.2 (a), where the cells in white are the original values of the a9's filter and the cells in gray are zeros. This is done because a8 is not presented in DPN, such that each filter in a9 should be convolved on every other

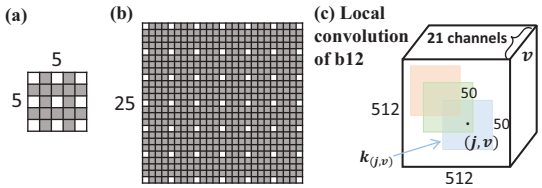

Figure 2: (a) and (b) show the padding of the filters. (c) illustrates local convolution of b12.

pixel of a7. To maintain the convolution with one stride, we pad the filters with zeros. Furthermore, the feature maps in b11 are up-sampled to 512×512 by bilinear interpolation. Since DPN is trained with label maps of the entire images, the missing information in the preceding layers of b11 can be recovered by BP.

Second, two fully-connected layers at a11 are transformed to two convolutional layers at b9 and b10, respectively. As shown in Table 1 (a), the first 'fc' layer learns 7×7×512×4096 parameters, which can be altered to 4096 filters in b9, each of which is 25×25×512. Since a8 and a10 have been removed, the 7×7 receptive field is padded into 25×25 similar as above and shown in Fig.2 (b). The second 'fc' layer learns a 4096×4096 weight matrix, corresponding to 4096 filters in b10. Each filter is 1×1×4096.

Overall, b11 generates the unary labeling results, producing twenty-one 512×512 feature maps, each of which represents the probabilistic label map of each category.

3.2. Modeling Smoothness Terms

The last four layers of DPN, *i.e.* from b12 to b15, are carefully designed to smooth the unary labeling results.

• **b12** As listed in Table 1 (b), 'lconv' in b12 indicates a **locally convolutional layer**, which is widely used in face recognition [33, 35] to capture different information from different facial positions. Similarly, distinct spatial positions of b12 have different filters, and each filter is shared across 21 input channels, as shown in Fig.2 (c). It

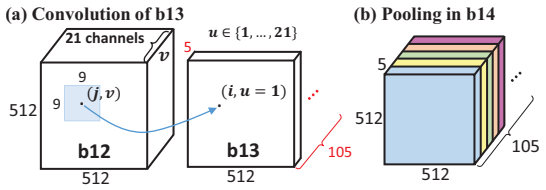

Figure 3: (a) and (b) illustrates the convolutions of b13 and the poolings in b14.

can be formulated as

$$\mathbf{o}^{12}_{(j,v)} = \mathrm{lin}(\mathbf{k}_{(j,v)} * \mathbf{o}^{11}_{(j,v)}), \tag{8}$$

where $\mathrm{lin}(x) = ax + b$ representing the linear activation function, '$*$' is the convolutional operator, and $\mathbf{k}_{(j,v)}$ is a 50×50×1 filter at position j of channel v. We have $\mathbf{k}_{(j,1)} = \mathbf{k}_{(j,2)} = ... = \mathbf{k}_{(j,21)}$ shared across 21 channels. $\mathbf{o}^{11}_{(j,v)}$ indicates a local patch in b11, while $\mathbf{o}^{12}_{(j,v)}$ is the corresponding output of b12. Since b12 has stride one, the result of $\mathbf{k}_j * \mathbf{o}^{11}_{(j,v)}$ is a scalar. In summary, b12 has 512×512 different filters and produces 21 output feature maps.

Eqn.(8) implements the **triple penalty** of Eqn.(7). Recall that each output feature map of b11 indicates a probabilistic label map of a specific object appearing in the image. As a result, Eqn.(8) suggests that the probability of object v presented at position j is updated by weighted averaging over the probabilities at its nearby positions. Thus, as shown in Fig.1 (c), $\mathbf{o}^{11}_{(j,v)}$ corresponds to a patch of Q^v centered at j, which has values p^v_z, $\forall z \in \mathcal{N}^{50\times50}_j$. Similarly, $\mathbf{k}_{(j,v)}$ is initialized by $\mathrm{d}(j,z)p^v_j$, implying each filter captures dissimilarities between positions. These filters remain fixed during BP, other than learned as in conventional CNN[2].

• **b13** As shown in Table 1 (b) and Fig.3 (a), b13 is a convolutional layer that generates 105 feature maps by using 105 filters of size 9×9×21. For example, the value of $(i, u = 1)$ is attained by applying a 9×9×21 filter at positions $\{(j, v = 1, ..., 21)\}$. In other words, b13 learns a filter for each category to penalize the probabilistic label maps of b12, corresponding to the **local label contexts** in Eqn.(7) by assuming $K = 5$ and $n = 9$, as shown in Fig.1 (d).

• **b14** As illustrated in Table 1 and Fig.3 (b), b14 is a block min pooling layer that pools over every 1×1 region with one stride across every 5 input channels, leading to 21 output channels, *i.e.* 105÷5=21. b14 activates the contextual pattern with the smallest penalty.

[2]Each filter in b12 actually represents a distance metric between pixels in a specific region. In VOC12, the patterns of all the training images in a specific region are heterogenous, because of various object shapes. Therefore, we initialize each filter with Euclidean distance. Nevertheless, Eqn.(8) is a more general form than the triple penalty in Eqn.(7), *i.e.* filters in (8) can be automatically learned from data, if the patterns in a specific region are homogenous, such as face or human images, which have more regular shapes than images in VOC12.

• **b15** This layer combines both the unary and smoothness terms by summing the outputs of b11 and b14 in an element-wise manner similar to Eqn.(7),

$$\mathbf{o}^{15}_{(i,u)} = \frac{\exp\left\{\ln(\mathbf{o}^{11}_{(i,u)}) - \mathbf{o}^{14}_{(i,u)}\right\}}{\sum_{u=1}^{21}\exp\left\{\ln(\mathbf{o}^{11}_{(i,u)}) - \mathbf{o}^{14}_{(i,u)}\right\}}, \tag{9}$$

where probability of assigning label u to pixel i is normalized over all the labels.

Relation to Previous Deep Models Many existing deep models such as [39, 3, 30] employed Eqn.(3) as the pairwise terms, which are the special cases of Eqn.(7). To see this, let K=1 and j=i, the right hand side of (7) reduces to

$$\begin{aligned} & \exp\{-\Phi^u_i - \sum_{v\in L}\lambda_1\mu_1(i,u,i,v)\sum_{z\in\mathcal{N}_i}\mathrm{d}(i,z)p^v_i p^v_z\} \\ = \ & \exp\{-\Phi^u_i - \sum_{v\in L}\mu(u,v)\sum_{z\in\mathcal{N}_i, z\neq i}\mathrm{d}(i,z)p^v_z\}, \end{aligned} \tag{10}$$

where $\mu(u,v)$ and $\mathrm{d}(i,z)$ represent the global label co-occurrence and pairwise pixel similarity of Eqn.(3), respectively. This is because λ_1 is a constant, $\mathrm{d}(i,i) = 0$, and $\mu(i,u,i,v) = \mu(u,v)$. Eqn.(10) is the corresponding MF update equation of (3).

3.3. Learning Algorithms

Learning The first ten groups of DPN are initialized by VGG_{16}[3], while the last four groups can be initialized randomly. DPN is then fine-tuned in an incremental manner with four stages. During fine-tuning, all these stages solve the pixelwise softmax loss [22], but updating different sets of parameters.

First, we add a loss function to b11 and fine-tune the weights from b1 to b11 without the last four groups, in order to learn the unary terms. Second, to learn the triple relations, we stack b12 on top of b11 and update its parameters (*i.e.* ω_1, ω_2 in the distance measure), but the weights of the preceding groups (*i.e.* b1∼b11) are fixed. Third, b13 and b14 are stacked onto b12 and similarly, their weights are updated with all the preceding parameters fixed, so as to learn the local label contexts. Finally, all the parameters are jointly fine-tuned.

Implementation DPN transforms Eqn.(7) into convolutions and poolings in the groups from b12 to b15, such that filtering at each pixel can be performed in a parallel manner. Assume we have f input and f' output feature maps, $N \times N$ pixels, filters with $s \times s$ receptive field, and a mini-batch with M samples. b12 takes a total $f \cdot N^2 \cdot s^2 \cdot M$ operations, b13 takes $f \cdot f' \cdot N^2 \cdot s^2 \cdot M$ operations, while both b14 and b15 require $f \cdot N^2 \cdot M$ operations.

[3]We use the released VGG_{16} model, which is public available at http://www.robots.ox.ac.uk/~vgg/research/very_deep/

For example, when $M=10$ as in our experiment, we have $21\times512^2\times50^2\times10=1.3\times10^{11}$ operations in b12, which has the highest complexity in DPN. We parallelize these operations using matrix multiplication on GPU as [4] did, b12 can be computed within 30ms. The total runtime of the last four layers of DPN is 75ms. Note that convolutions in DPN can be further speeded up by low-rank decompositions [14] of the filters and model compressions [13].

However, direct calculation of Eqn.(7) is accelerated by fast Gaussian filtering [1]. For a mini-batch of ten 512×512 images, a recently optimized implementation [16] takes 12 seconds on CPU to compute one iteration of (7). Therefore, DPN makes (7) easier to be parallelized and speeded up.

4. Experiments

Dataset We evaluate the proposed approach on the PASCAL VOC 2012 (VOC12) [7] dataset, which contains 20 object categories and one background category. Following previous works such as [12, 22, 3], we employ $10,582$ images for training, $1,449$ images for validation, and $1,456$ images for testing.

Evaluation Metrics All existing works employed mean pixelwise intersection-over-union (denoted as mIoU) [22] to evaluate their performance. To fully examine the effectiveness of DPN, we introduce another three metrics, including tagging accuracy (TA), localization accuracy (LA), and boundary accuracy (BA). (1) TA compares the predicted image-level tags with the ground truth tags, calculating the accuracy of multi-class image classification. (2) LA evaluates the IoU between the predicted object bounding boxes[4] and the ground truth bounding boxes (denoted as bIoU), measuring the precision of object localization. (3) For those objects that have been correctly localized, we compare the predicted object boundary with the ground truth boundary, measuring the precision of semantic boundary similar to [12].

Comparisons DPN is compared with the best-performing methods on VOC12, including FCN [22], Zoom-out [25], DeepLab [3], WSSL [28], BoxSup [5], Piecewise [19], and RNN [39]. All these methods are based on CNNs and MRFs, and trained on VOC12 data following [22]. They can be grouped according to different aspects: (1) **joint-train**: Piecewise and RNN; (2) **w/o joint-train**: DeepLab, WSSL, FCN, and BoxSup; (3) **pre-train on COCO**: RNN, WSSL, and BoxSup. The first and the second groups are the methods with and without joint training CNNs and MRFs, respectively. Methods in the last group also employed MS-COCO [20] to pre-train deep models. To conduct a comprehensive comparison, the performance of DPN are reported on both settings, *i.e.*, with and without pre-training on COCO.

[4]They are the bounding boxes of the predicted segmentation regions.

Receptive Field	baseline	10×10	50×50	100×100
mIoU (%)	63.4	63.8	**64.7**	64.3

(a) Comparisons between different receptive fields of b12.

Receptive Field	1×1	5×5	9×9	9×9 mixtures
mIoU (%)	64.8	66.0	66.3	**66.5**

(b) Comparisons between different receptive fields of b13.

Pairwise Terms	DSN [30]	DeepLab [3]	DPN
improvement (%)	2.6	3.3	**5.4**

(c) Comparing pairwise terms of different methods.

Table 2: Ablation study of hyper-parameters.

In the following, Sec.4.1 investigates the effectiveness of different components of DPN on the VOC12 *validation set*. Sec.4.2 compares DPN with the state-of-the-art methods on the VOC12 *test set*.

4.1. Effectiveness of DPN

All the models evaluated in this section are trained and tested on VOC12.

Triple Penalty The receptive field of b12 indicates the range of triple relations for each pixel. We examine different settings of the receptive fields, including '10×10', '50×50', and '100×100', as shown in Table 2 (a), where '50×50' achieves the best mIoU, which is sightly better than '100×100'. For a 512×512 image, this result implies that 50×50 neighborhood is sufficient to capture relations between pixels, while smaller or larger regions tend to under-fit or over-fit the training data. Moreover, all models of triple relations outperform the 'baseline' method that models dense pairwise relations, *i.e.* VGG_{16}+denseCRF [16].

Label Contexts Receptive field of b13 indicates the range of local label context. To evaluate its effectiveness, we fix the receptive field of b12 as 50×50. As summarized in Table 2 (b), '9×9 mixtures' improves preceding settings by 1.7, 0.5, and 0.2 percent respectively. We observe large gaps exist between '1×1' and '5×5'. Note that the 1×1 receptive field of b13 corresponds to learning a global label co-occurrence without considering local spatial contexts. Table 2 (c) shows that the pairwise terms of DPN are more effective than DSN and DeepLab[5].

More importantly, mIoU of all the categories can be improved when increasing the size of receptive field and learning a mixture. Specifically, for each category, the improvements of the last three settings in Table 2 (b) over the first one are 1.2±0.2, 1.5±0.2, and 1.7±0.3, respectively.

We also visualize the learned label compatibilities and contexts in Fig.4 (a) and (b), respectively. (a) is obtained by summing each filter in b13 over 9×9 region, indicating

[5]The other deep models such as RNN and Piecewise did not report the exact imrprovements after combining unary and pairwise terms.

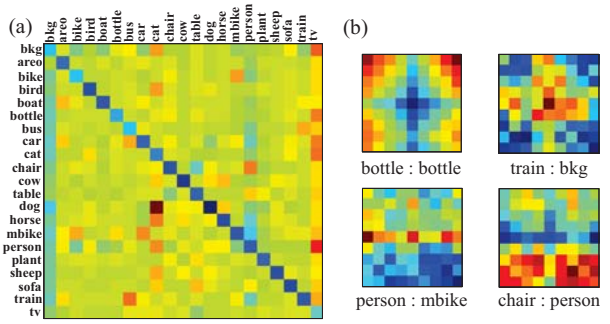

Figure 4: Visualization of (a) learned label compatibility (b) learned contextual information. **(Best viewed in color)**

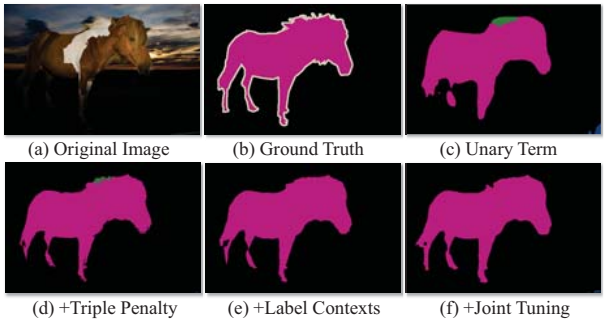

Figure 5: Step-by-step visualization of DPN. **(Best viewed in color)**

how likely a column object would present when a row object is presented. Blue represents high possibility. (a) is non-symmetry. For example, when 'horse' is presented, 'person' is more likely to present than the other objects. Also, 'chair' is compatible with 'table' and 'bkg' is compatible with all the objects. (b) visualizes some contextual patterns, where 'A:B' indicates that when 'A' is presented, where 'B' is more likely to present. For example, 'bkg' is around 'train', 'motor bike' is below 'person', and 'person' is sitting on 'chair'.

Incremental Learning As discussed in Sec.3.3, DPN is trained in an incremental manner. The right hand side of Table 3 (a) demonstrates that each stage leads to performance gain compared to its previous stage. For instance, 'triple penalty' improves 'unary term' by 2.3 percent, while 'label contexts' improves 'triple penalty' by 1.8 percent. More importantly, joint fine-tuning all the components (*i.e.* unary terms and pairwise terms) in DPN achieves another gain of 1.3 percent. A step-by-step visualization is provided in Fig.5.

We also compare 'incremental learning' with 'joint learning', which fine-tunes all the components of DPN at the same time. The training curves of them are plotted in Fig.6 (a), showing that the former leads to higher and more stable accuracies with respect to different iterations, while the latter may get stuck at local minima. This difference

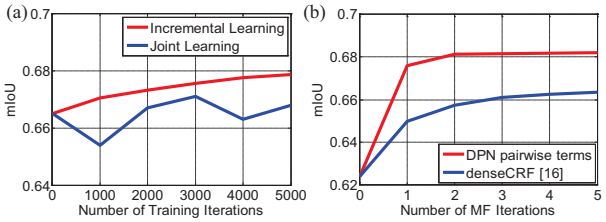

Figure 6: Ablation study of (a) training strategy (b) required MF iterations. **(Best viewed in color)**

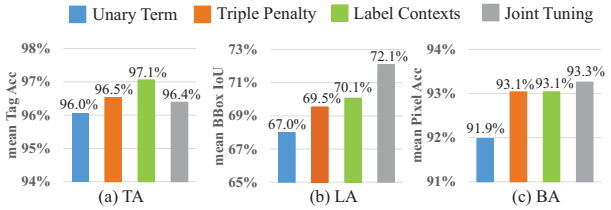

Figure 7: Stage-wise analysis of (a) mean tagging accuracy (b) mean localization accuracy (c) mean boundary accuracy.

is easy to understand, because incremental learning only introduces new parameters until all existing parameters have been fine-tuned.

One-iteration MF DPN approximates one iteration of MF. Fig.6 (b) illustrates that DPN reaches a good accuracy with one MF iteration. A CRF [16] with dense pairwise edges needs more than 5 iterations to converge. It also has a large gap compared to DPN. Note that the existing deep models such as [3, 39, 30] required 5∼10 iterations to converge as well.

Different Components Modeling Different Information We further evaluate DPN using three metrics. The results are given in Fig.7. For example, (a) illustrates that the tagging accuracy can be improved in the third stage, as it captures label co-occurrence with a mixture of contextual patterns. However, TA is decreased a little after the final stage. Since joint tuning maximizes segmentation accuracies by optimizing all components together, extremely small objects, which rarely occur in VOC training set, are discarded. As shown in (b), accuracies of object localization are significantly improved in the second and the final stages. This is intuitive because the unary prediction can be refined by long-range and high-order pixel relations, and joint training further improves results. (c) discloses that the second stage also captures object boundary, since it measures dissimilarities between pixels.

Per-class Analysis Table 3 (a) reports the per-class accuracies of four evaluation metrics, where the first four rows represent the mIoU of four stages, while the last three rows represent TA, LA, and BA, respectively. We have several valuable observations, which motivate future researches. (1) Joint training benefits most of the categories,

	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	**Avg.**
Unary Term (mIoU)	77.5	34.1	76.2	58.3	63.3	78.1	72.5	76.5	26.6	59.9	40.8	70.0	62.9	69.3	76.3	39.2	70.4	37.6	72.5	57.3	62.4
+ Triple Penalty	82.3	35.9	80.6	60.1	64.8	79.5	74.1	80.9	27.9	63.5	40.4	73.8	66.7	70.8	79.0	42.0	74.1	39.1	73.2	58.5	64.7
+ Label Contexts	83.2	35.6	**82.6**	61.6	65.5	80.5	74.3	**82.6**	29.9	**67.9**	47.5	75.2	**70.3**	71.4	79.6	42.7	**77.8**	40.6	75.3	59.1	66.5
+ Joint Tuning	**84.8**	**37.5**	80.7	**66.3**	**67.5**	**84.2**	**76.4**	81.5	**33.8**	65.8	**50.4**	**76.8**	67.1	**74.9**	**81.1**	**48.3**	75.9	**41.8**	**76.6**	**60.4**	**67.8**
TA (tagging Acc.)	98.8	97.9	98.4	97.7	96.1	98.6	95.2	96.8	90.1	97.5	95.7	96.7	96.3	98.1	93.3	96.1	98.7	92.2	97.4	96.3	96.4
LA (bIoU)	81.7	76.3	75.5	70.3	54.4	86.4	70.6	85.6	51.8	79.6	57.1	83.3	79.2	80.0	74.1	53.1	79.1	68.4	76.3	58.8	72.1
BA (boundary Acc.)	95.9	83.9	96.9	92.6	93.8	94.0	95.7	95.6	89.5	93.3	91.4	95.2	94.2	92.7	94.5	90.4	94.8	90.5	93.7	96.6	93.3

(a) Per-class results on VOC12 *val*.

	areo	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
FCN [22]	76.8	34.2	68.9	49.4	60.3	75.3	74.7	77.6	21.4	62.5	46.8	71.8	63.9	76.5	73.9	45.2	72.4	37.4	70.9	55.1	62.2
Zoom-out [25]	85.6	37.3	83.2	62.5	66.0	85.1	80.7	84.9	27.2	73.2	57.5	78.1	79.2	81.1	77.1	53.6	74.0	49.2	71.7	63.3	69.6
Piecewise [19]	87.5	37.7	75.8	57.4	72.3	88.4	82.6	80.0	33.4	71.5	55.0	79.3	78.4	81.3	82.7	56.1	79.8	48.6	77.1	66.3	70.7
DeepLab [3]	84.4	54.5	81.5	63.6	65.9	85.1	79.1	83.4	30.7	74.1	59.8	79.0	76.1	83.2	80.8	59.7	82.2	50.4	73.1	63.7	71.6
RNN [39]	87.5	39.0	79.7	64.2	68.3	87.6	80.8	84.4	30.4	78.2	60.4	80.5	77.8	83.1	80.6	59.5	82.8	47.8	78.3	67.1	72.0
WSSL† [28]	89.2	46.7	88.5	63.5	68.4	87.0	81.2	86.3	32.6	80.7	62.4	81.0	81.3	84.3	82.1	56.2	84.6	58.3	76.2	67.2	73.9
RNN† [39]	90.4	55.3	88.7	68.4	69.8	88.3	82.4	85.1	32.6	78.5	64.4	79.6	81.9	**86.4**	81.8	58.6	82.4	53.5	77.4	70.1	74.7
BoxSup† [5]	**89.8**	38.0	**89.2**	**68.9**	68.0	89.6	83.0	**87.7**	34.4	83.6	**67.1**	81.5	83.7	85.2	83.5	58.6	84.9	55.8	**81.2**	**70.7**	75.2
DPN	87.7	59.4	78.4	64.9	70.3	89.3	83.5	86.1	31.7	79.9	62.6	81.9	80.0	83.5	82.3	60.5	83.2	53.4	77.9	65.0	74.1
DPN†	89.0	**61.6**	87.7	66.8	**74.7**	**91.2**	**84.3**	87.6	**36.5**	**86.3**	66.1	**84.4**	**87.8**	85.6	**85.4**	**63.6**	**87.3**	**61.3**	79.4	66.4	**77.5**

(b) Per-class results on VOC12 *test*. The approaches pre-trained on COCO [20] are marked with †.

Table 3: Per-class results on VOC12.

except animals such as 'bird', 'cat', and 'cow'. Some instances of these categories are extremely small so that joint training discards them for smoother results. (2) Training DPN with pixelwise label maps implicitly models image-level tags, since it achieves a high averaged TA of 96.4%. (3) Object localization always helps. However, for the object with complex boundary such as 'bike', its mIoU is low even it can be localized, *e.g.* 'bike' has high LA but low BA and mIoU. (4) Failures of different categories have different factors. With these three metrics, they can be easily identified. For example, the failures of 'chair', 'table', and 'plant' are caused by the difficulties to accurately capture their bounding boxes and boundaries. Although 'bottle' and 'tv' are also difficult to localize, they achieve moderate mIoU because of their regular shapes. In other words, mIoU of 'bottle' and 'tv' can be significantly improved if they can be accurately localized.

4.2. Overall Performance

As shown in Table 3 (b), we compare DPN with the best-performing methods[6] on VOC12 test set based on two settings, *i.e.* with and without pre-training on COCO. The approaches pre-trained on COCO are marked with '†'. We evaluate DPN on several scales of the images and then average the results following [3, 19].

DPN outperforms all the existing methods that were trained on VOC12, but DPN needs only one MF iteration to solve MRF, other than 10 iterations of RNN, DeepLab, and Piecewise. By averaging the results of two DPNs, we achieve 74.1% accuracy on VOC12 without outside training data. As discussed in Sec.3.3, MF iteration is the most complex step even when it is implemented as convolutions. Therefore, DPN at least reduces 10× runtime compared to previous works.

Following [39, 5], we pre-train DPN with COCO, where 20 object categories that are also presented in VOC12 are selected for training. A single DPN† has achieved 77.5% mIoU on VOC12 test set. As shown in Table 3 (b), we observe that DPN† achieves best performances on more than half of the object classes.

[6]The results of these methods were presented in either the published papers or arXiv pre-prints.

5. Conclusion

We proposed Deep Parsing Network (DPN) to address semantic image segmentation, which has several appealing properties. First, DPN unifies the inference and learning of unary term and pairwise terms in a single convolutional network. No iterative inference are required during back-propagation. Second, high-order relations and mixtures of label contexts are incorporated to its pairwise terms modeling, making existing works serve as special cases. Third, DPN is built upon conventional operations of CNN, thus easy to be parallelized and speeded up.

DPN achieves state-of-the-art performance on VOC12, and multiple valuable facts about semantic image segmentation are revealed through extensive experiments. Future directions include investigating the generalizability of DPN to more challenging scenarios, *e.g.* large number of object classes and substantial appearance/scale variations.

References

[1] A. Adams, J. Baek, and M. A. Davis. Fast high-dimensional filtering using the permutohedral lattice. In *Computer Graphics Forum*, volume 29, pages 753–762, 2010.

[2] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik. Contour detection and hierarchical image segmentation. *PAMI*, 33(5):898–916, 2011.

[3] 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 *ICLR*, 2015.

[4] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer. cudnn: Efficient primitives for deep learning. In *NIPS Deep Learning Workshop*, 2014.

[5] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. *arXiv:1503.01640v2*, 18 May 2015.

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009.

[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010.

[8] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *PAMI*, 35(8):1915–1929, 2013.

[9] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *IJCV*, 70(1):41–54, 2006.

[10] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael. Learning low-level vision. *IJCV*, 40(1):25–47, 2000.

[11] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *ICCV*, pages 670–677, 2009.

[12] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998, 2011.

[13] G. E. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. In *NIPS Deep Learning Workshop*, 2014.

[14] M. Jaderberg, A. Vedaldi, and A. Zisserman. Speeding up convolutional neural networks with low rank expansions. In *BMVC*, 2014.

[15] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233, 1999.

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

[17] P. Krähenbühl and V. Koltun. Parameter learning and convergent inference for dense random fields. In *ICML*, pages 513–521, 2013.

[18] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012.

[19] G. Lin, C. Shen, I. Reid, and A. Hengel. Efficient piecewise training of deep structured models for semantic segmentation. *arXiv:1504.01013v2*, 23 Apr 2015.

[20] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755. 2014.

[21] C. Liu, J. Yuen, and A. Torralba. Nonparametric scene parsing via label transfer. *PAMI*, 33(12):2368–2382, 2011.

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

[23] P. Luo, X. Wang, and X. Tang. Hierarchical face parsing via deep learning. In *CVPR*, pages 2480–2487, 2012.

[24] P. Luo, X. Wang, and X. Tang. Pedestrian parsing via deep decompositional network. In *ICCV*, pages 2648–2655, 2013.

[25] M. Mostajabi, P. Yadollahpour, and G. Shakhnarovich. Feedforward semantic segmentation with zoom-out features. In *CVPR*, pages 3376–3385, 2015.

[26] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille. The role of context for object detection and semantic segmentation in the wild. In *CVPR*, pages 891–898, 2014.

[27] M. Opper, O. Winther, et al. From naive mean field theory to the tap equations. 2001.

[28] G. Papandreou, L.-C. Chen, K. Murphy, and A. L. Yuille. Weakly-and semi-supervised learning of a dcnn for semantic image segmentation. *arXiv:1502.02734v2*, 8 May 2015.

[29] X. Ren and J. Malik. Learning a classification model for segmentation. In *ICCV*, pages 10–17, 2003.

[30] A. G. Schwing and R. Urtasun. Fully connected deep structured networks. *arXiv:1503.02351v1*, 9 Mar 2015.

[31] J. Shi and J. Malik. Normalized cuts and image segmentation. *PAMI*, 22(8):888–905, 2000.

[32] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[33] Y. Sun, X. Wang, and X. Tang. Deep learning face representation by joint identification-verification. In *NIPS*, 2014.

[34] M. Szummer, P. Kohli, and D. Hoiem. Learning crfs using graph cuts. In *ECCV*, pages 582–595. 2008.

[35] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification. In *CVPR*, pages 1701–1708, 2014.

[36] V. Vineet, G. Sheasby, J. Warrell, and P. H. Torr. Posefield: An efficient mean-field based method for joint estimation of human pose, segmentation, and depth. In *Energy Minimization Methods in Computer Vision and Pattern Recognition*, pages 180–194. Springer, 2013.

[37] V. Vineet, J. Warrell, and P. H. Torr. Filter-based mean-field inference for random fields with higher-order terms and product label-spaces. In *ECCV*, pages 31–44. 2012.

[38] J. Yang, B. Price, S. Cohen, and M.-H. Yang. Context driven scene parsing with attention to rare classes. In *CVPR*, pages 3294–3301, 2014.

[39] 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. *arXiv:1502.03240v2*, 30 Apr 2015.