segDeepM: Exploiting Segmentation and Context in Deep Neural Networks for Object Detection

Yukun Zhu   Raquel Urtasun   Ruslan Salakhutdinov   Sanja Fidler
University of Toronto
{yukun,urtasun,rsalakhu,fidler}@cs.toronto.edu

Abstract

In this paper, we propose an approach that exploits object segmentation in order to improve the accuracy of object detection. We frame the problem as inference in a Markov Random Field, in which each detection hypothesis scores object appearance as well as contextual information using Convolutional Neural Networks, and allows the hypothesis to choose and score a segment out of a large pool of accurate object segmentation proposals. This enables the detector to incorporate additional evidence when it is available and thus results in more accurate detections. Our experiments show an improvement of 4.1% in mAP over the R-CNN baseline on PASCAL VOC 2010, and 3.4% over the current state-of-the-art, demonstrating the power of our approach.

1. Introduction

In the past two years, Convolutional Neural Networks (CNNs) have revolutionized computer vision. They have been applied to a variety of general vision problems, such as recognition [15, 9], segmentation [11], stereo [18], flow [24], and even text-from-image generation [13], consistently outperforming past work. This is mainly due to their high generalization power achieved by learning complex, non-linear dependencies across millions of labelled examples.

It has recently been shown that increasing the depth of the network increases the performance by an additional impressive margin on the ImageNet challenge [21, 22]. It remains to be seen whether recognition can be solved by simply pushing the limits of computation (the size of the networks) and increasing the amount of the training data. We believe that the main challenge in the next few years will be to design computationally simpler and more efficient models that can achieve a similar or better performance compared to the very deep networks.

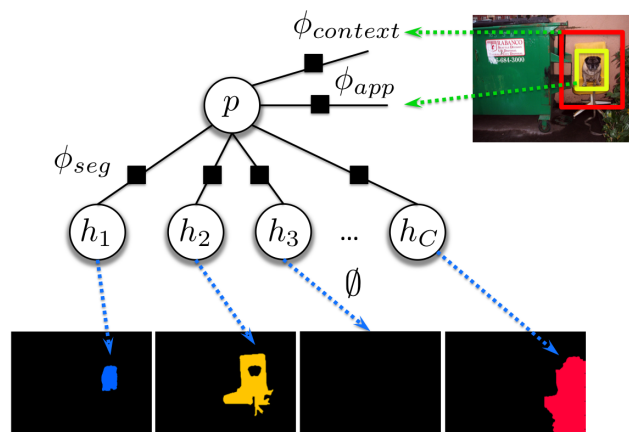For object detection, a successful approach has been to

Figure 1: Proposed segDeepM model.

generate a large pool of candidate boxes [23] and classify them using CNNs [9]. The quality of such a detector thus largely depends on the quality of the object hypotheses. Interestingly, however, using much better proposals obtained via a high-end bottom-up segmentation approach [11] has resulted only in small improvements in accuracy.

In this paper, we show how to exploit a small number of accurate object segment proposals in order to significantly improve object detection performance. We frame the detection problem as inference in a Markov Random Field as in Figure 1, in which each detection hypothesis scores object appearance as well as contextual information using Convolutional Neural Networks. Each hypothesis can choose and score a segment out of a small pool of accurate object segmentation proposals. This enables our approach to place more accurate object bounding boxes in parts of the image where an object segmentation hypothesis [2] exists or where strong contextual cues are available. We additionally show that a significant performance boost can be obtained by a sequential approach, where the network iterates between adjusting its spatial scope (the bounding box) and classifying its content. This strategy reduces the dependency on the initial candidate boxes obtained by [23] and enables our approach to recover from the potentially bad initial localiza-

tion.

We show that our model, called segDeepM, outperforms the baseline R-CNN [9] approach by 3.2% with almost no extra computational cost. We get a total of 5% improvement by incorporating contextual information at the cost of doubling the running time of the method. On PASCAL VOC 2010 test, our method achieves 4.1% improvement over R-CNN and 1.4% over the current state-of-the-art.

2. Related Work

In the past years, a variety of segmentation algorithms that exploit object detections as a top-down cue have been explored. The standard approach has been to use detection features as unary potentials in an MRF [14], or as candidate bounding boxes for holistic MRFs [26, 16]. In [20], segmentation within the detection boxes has been performed using a GrabCut method. In [1], object segmentations are found by aligning the masks obtained from Poselets [1, 17].

There have been a few approaches to use segmentation to improve object detection. [10] cast votes for the object's location by using a Hough transform with a set of regions. [5] uses DPM to find a rough object location and refines it according to color information and occlusion boundaries. In [11], segmentation is used to mask-out the background inside the detection, resulting in improved performance. Segmentation and detection has also been addressed in a joint formulation in [25] by combining shape information obtained via DPM parts as well as color and boundary cues.

Our work is inspired by the success of segDPM [8]. By augmenting the DPM detector [7] with very simple segmentation features that can be computed in constant time, segDPM improved the detection performance by 8% on the challenging PASCAL VOC dataset. The approach used segments computed from the final segmentation output of CPMC [2] in order to place accurate boxes in parts of the image where segmentation for the object class of interest was available. This idea was subsequently exploited in [19] by augmenting the DPM with an additional set of deformable context "parts" which scored contextual segmentation features around the object. In [4], the segDPM detector [8] was augmented with part visibility reasoning, achieving state-of-the-art results for detection of articulated classes. In [6], the authors extended segDPM to incorporate segmentation compatibility also at the part level.

In this paper, we build on R-CNN framework [9] and transfer the core ideas of segDPM. We use appearance features from [15, 9], a rich contextual appearance description around the object, and a MRF model that is able to exploit segmentation in a more efficient way than segDPM.

3. Our Approach

The goal of our approach is to efficiently exploit segmentation and contextual cues in order to facilitate object detec-

tion. Following the R-CNN setup, we compute the Selective Search boxes [23] yielding approximately 2000 object candidates per image. For each box we extract the last feature layer of the CNN network [15], that is fine-tuned on the PASCAL dataset as proposed in [9]. We obtain object segment proposals via the CPMC approach [3], although our approach is independent of this choice. Following [2], we take the top 150 proposals given by an object-independent ranker, and train class-specific classifiers for all classes of interest by the second-order pooling method O2P [2]. We remove all segments that have less than 1500 pixels. Our method will make use of these segments along with their class-specific scores. This is slightly different than segDPM which takes only 1 or 2 segments carved out from the final O2P's pixel-level labeling of the image.

In the remainder of this section we first define our model and describe its segmentation and contextual features. We next discuss inference and learning. Finally, we detail a sequential inference scheme that iterates between correcting the input bounding boxes and scoring them with our model.

3.1. The segDeepM Model

We define our model as a Markov Random Field with random variables that reason about detection boxes, object segments, and context. Similar to [7, 8], we define p as a random variable denoting the location and scale of a candidate bounding box in the image. We also define $\mathbf{h}$ to be a set of random variables, one for each class, i.e. $\mathbf{h} = (h_1, h_2, \ldots, h_C)^T$. Each random variable $h_c \in \{0, 1, \ldots, H(\mathbf{x})\}$ represents an index into the set of all candidate segments. Here C is the total number of object classes of interest and $H(\mathbf{x})$ is the total number of segments in image x. The random variable h_c allows each candidate detection box to *choose* a segment for each class and score its confidence according to the agreement with the segment. The idea is to (1) boost the confidence of boxes that are well aligned with a high scoring object region proposal for the class of interest, and (2) adjust its score based on the proximity and confidence of region proposals for other classes, serving as context for the model. This is different from segDPM that only had a single random variable h which selected a segment belonging to the detector's class. It is also different from [19] in that the model chooses contextual segments, and does not score context in a fixed segmentation window. Note that $h_c = 0$ indicates that no segment is selected for class c. This means that either no segment for a class of interest is in the vicinity of the detection hypothesis, or that none of the regions corresponding to the contextual class c help classification of the current box. We define the energy of a configuration as follows:

$$E(p, \mathbf{h}; \mathbf{x}) = \omega_{app}^T \cdot \phi_{app}(p; \mathbf{x}) + \omega_{seg}^T \cdot \phi_{seg}(p, \mathbf{h}; \mathbf{x}) \quad (1)$$
$$+ \omega_{ctx}^T \cdot \phi_{ctx}(p; \mathbf{x}),$$

where $\phi_{app}(x, p)$, $\phi_{seg}(p, \mathbf{h}; \mathbf{x})$, and $\phi_{ctx}(p; \mathbf{x})$ are the candidate's appearance, segmentation, and contextual potential functions (features), respectively. We describe the potentials in detail below.

3.2. Details of Potential Functions

Appearance: To extract the appearance features we follow [9]. The image in each candidate detection's box is warped to a fixed size $227 \times 227 \times 3$. We run the image through the CNN [15] trained on the ImageNet dataset and fine-tuned on PASCAL's data [9]. As our appearance feature $\phi_{app}(p; \mathbf{x})$ we use the 4096-dimensional feature extracted from the $fc7$ layer.

Segmentation: Similar to [8], our segmentation features attempt to capture the agreement between the candidate's bounding box and a particular segment. The features are complementary in nature, and, when combined within the model, aim at placing the box tightly around each segment. We emphasize that the weights for each feature will be learned, thus allowing the model to adjust the importance of each feature's contribution to the joint energy.

We use slightly more complex features tailored to exploit a much larger set of segments than [8]. In particular, we use a grid feature that aims to capture a loose geometric arrangement of the segment inside the candidate's box. We also incorporate class information, where the model is allowed to choose a different segment for each class, depending on the contextual information contained in a segment with respect to the class of the detector.

We use multiple segmentation features, one for each class, thus our segmentation term decomposes:

$$\omega_{seg}^T \cdot \phi_{seg}(p, \mathbf{h}; \mathbf{x}) = \sum_{c \in \{1,...,C\}} \sum_{type} \omega_{type}^T \cdot \phi_{type}(p, h_c; \mathbf{x}).$$

Specifically, we consider the following features:

SegmentGrid-In: Let $S(h_c)$ denote the binary mask of the segment chosen by h_c. For a particular candidate box p, we crop the segment's mask via the bounding box of p and compute the SegmentGrid-in feature on a $K \times K$ grid $\mathcal{G}$ placed over the cropped mask. The kth dimension represents the percentage of segment's pixels inside the kth block, relative to the number of all pixels in $S(h_c)$.

$$\phi_{seggrid-in}(x, p, h_c, k) = \frac{1}{|S(h_c)|} \sum_{i \in \mathcal{G}(p,k)} S(h_c, i), \quad (2)$$

where $\mathcal{G}(p, k)$ is the kth block of pixels in grid $\mathcal{G}$, and $S(h_c, i)$ indexes the segment's mask in pixel i. That is, $S(h_c, i) = 1$ when pixel i is part of the segment and $S(h_c, i) = 0$ otherwise. For c matching the detector's class, this feature will attempt to place a box slightly bigger than the segment while at the same time trying to localize it such that the spatial distribution of pixels within each grid

matches the class' expected shape. For c other than the detector's class, this feature will try to place the box such that it intersects as little as possible with the segments of other classes. The dimensionality of this feature is $K \times K \times C$.

Segment-Out: This feature follows [8], and computes the percentage of segment pixels outside the candidate box. Unlike the SegmentGrid-In, this feature computes a single value for each segment/bounding box pair.

$$\phi_{seg-out}(p, h) = \frac{1}{|S(h)|} \sum_{i \notin \mathcal{B}(p)} S(h, i), \quad (3)$$

where $\mathcal{B}(p)$ is the bounding box corresponding to p. The aim of this feature is to place boxes that are smaller compared to the segments, which, in combination with SegmentGrid-In, achieves a tight fit around the segments.

BackgroundGrid-In: This feature is also computed with a $K \times K$ grid $\mathcal{G}$ for each bounding box p. We compute the percentage of pixels in each grid cell that are **not** part of the segment:

$$\phi_{back-in}(p, h, k) = \frac{1}{M - |S(h)|} \sum_{i \in \mathcal{G}(p,k)} (1 - S(h, i)),$$
(4)

with M the area of the largest segment for the image.

Background-Out: This scalar feature measures the % of segment's background outside of the candidate's box:

$$\phi_{back-out}(p, h) = \frac{1}{M - |S(h)|} \sum_{i \notin \mathcal{B}(p)} (1 - S(h, i)). \quad (5)$$

Overlap: Similarly to [8], we use another feature to measure the alignment of the candidate's box and the segment $S(h)$. It is computed as the intersection-over-union (IOU) between the box or p and a tightly fit bounding box around the segment $S(h)$.

$$\phi_{overlap}(x, p, h) = \frac{\mathcal{B}(p) \cap \mathcal{B}(S(x, h))}{\mathcal{B}(p) \cup \mathcal{B}(S(x, h))} - \lambda, \quad (6)$$

where $\mathcal{B}(S(x, h))$ is tight box around $S(x, h)$, and λ a bias term which we set to $\lambda = -0.7$ in our experiments.

SegmentClass: Since we are dealing with many segments per image, we add an additional feature to our model. We train the O2P [2] rankers for each class which uses several region-aware features as input into our segmentation features. Each ranker is trained to predict the IOU overlap of the given segment with the ground-truth object's segment. The output of all the class-specific rankers defines the following feature:

$$\phi_{potential}(h, c) = \frac{1}{1 + e^{-s(h,c)}}, \quad (7)$$

where $s(h, c)$ is the score of class c for segment $S(h)$.

SegmentGrid-in, segment-out, backgroundGrid-in, and background-out can be efficiently computed via integral images [8]. Note that [8]'s features are a special case of these features with a grid size $K = 1$. Overlap and segment features can also be quickly computed using matrix operations.

Context: CNNs are typically trained for the task of image classification where in most cases an input image is much larger than the object. This means that part of their success may be due to learning complex dependencies between the objects and their contextual information (*e.g.* sky for aeroplane, road for car and bus). However, the appearance features that we use are only computed based on the candidate's box, thus hardly capturing useful information from the scene. We thus add an additional feature that looks at a bigger scope than the candidate's box.

In particular, we enlarge each input candidate box by a fixed percentage ρ along its horizontal and vertical direction. For big boxes, or those close to the image boundary, we clip the enlarged region to be fully inside the image. We keep the object labels for each expanded box the same as that for the original boxes, even if the expanded box now encloses objects of other classes. We then warp the image in each enlarged box to 227×227 and fine-tune the original ImageNet-trained CNN using these images and labels. We call the fine-tuned network the *expanded CNN*. For our contextual features $\phi_{ctx}(x, p)$ we extract the $fc7$ layer features of the expanded CNN by running the warped image in the enlarged window through the network.

3.3. Inference

In the inference stage of our model, we score each candidate box p as follows:

$$f_{\boldsymbol{w}}(x, p) = \max_{\mathbf{h}} \left(\omega_{app}^T \cdot \phi_{app}(x, p) + \omega_{ctx}^T \cdot \phi_{ctx}(x, p) \right.$$
$$\left. + \omega_{seg}^T \cdot \phi_{seg}(x, p, \mathbf{h}) \right). \tag{8}$$

Observe that the first two terms in Eq. 8 can be computed efficiently by matrix multiplication, and the only part that depends on $\mathbf{h}$ is its last term. Although there could be exponential number of candidates for $\mathbf{h}$, we can greedily search each dimension of $\mathbf{h}$ and find the best segment $S(h_c)$ w.r.t. model parameters ω_{seg} for each class c. Since our segmentation features do not depend on the pairwise relationships in $\mathbf{h}$, this greedy approach is guaranteed to find the global maximum of $\omega_{seg}^T \cdot \phi_{seg}(x, p, \mathbf{h})$. Finally, we sum the three terms to obtain the score of each bounding box location p.

3.4. Learning

Given a set of images with N candidate boxes $\{p_n\}$ and their annotations $\{y(x_n, p_n)\}$, together with a collection of segments for each image $\{S(x_n, h_n)\}$ and associated potentials $\{\phi(x_n, h_n)\}$ with $n = 1, .., N$, training our model

can be written as follows:

$$\min_{\boldsymbol{\omega}} \|\boldsymbol{\omega}\|^2 + C \sum_{n=1}^{N} \max\left(0, 1 - y(x_n, p_n) f_{\boldsymbol{w}}(x_n, p_n)\right), \tag{9}$$

where $\boldsymbol{\omega}$ is a vector of all the weights in our model.

The learning problem in (9) is a latent SVM [7] where we treat the assignment variable $\mathbf{h}$ as a latent variable for each training instance. To optimize Equation (9), we iterate two steps following [7]:

1. *label each positive example:* for each (x, p) with $y(x, p) = 1$, we compute $\mathbf{h}^* = \arg\max_{\mathbf{h}} f_{\boldsymbol{w}}(x, p)$ with current the model parameters $\boldsymbol{\omega}$;

2. *update the weights:* we do hard-negative mining over a set of negative instances until reaching a certain memory limit. We then use stochastic gradient descent to optimize the weights $\boldsymbol{\omega}$.

Latent SVM is guaranteed to converge to a local minimum only, thus we need to carefully pick a good initialization for positive examples at the first iteration. We use the overlap feature $\phi_{overlap}$ as the indicator and set each dimension of $\mathbf{h}^*$ as $\mathbf{h}^* = \arg\max_{\mathbf{h}} \phi_{overlap}(p, \mathbf{h})$. This encourages the method to pick segments that best overlaps with the candidate's box.

Although our segmentation features are efficient to compute, we need to recompute them for all positive examples during the first step and for all hard negative examples during the second step of training. In our implementation, we cache a pool of segmentation features $\phi_{seg}(x, p, \mathbf{h})$ for all training instances to avoid computing them in every iteration. With the compact segmentation feature, our method achieves similar running speed with that of R-CNN [9].

3.5. Iterative Bounding Box Prediction

As a typical **postprocessing** step, object detection approaches usually perform bounding box prediction on their final candidate set [7, 9]. This typically results in a few percent improvement in accuracy, since the approach is able to make an informative re-localization based on complex features. Following [9], we also use $pool5$ layer features in order to do bounding box prediction.

In this paper we take this idea one step further by doing bounding box prediction and scoring the model in an iterative fashion. Our motivation is that better localization can lead to improved predictions. In particular, we first extract the CNN features and regress to a corrected set of boxes. We then re-extract the features on the new boxes and score our model. We only re-extract the features on the boxes which have changed by more than 20% percent from the original set. We can then repeat this process, by doing bounding box prediction again, re-extracting the features, and re-scoring.

	seg	exp	ibr	br	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
RCNN					69.9	64.2	48.0	30.2	26.9	63.3	56.0	67.6	26.8	44.7	29.6	61.7	55.7	69.8	56.4	26.6	56.7	35.6	54.4	57.7	50.1
RCNN+CPMC					71.5	65.3	48.6	31.5	27.9	64.3	57.2	67.6	26.7	46.2	33.6	62.8	57.8	70.7	57.9	26.6	54.0	37.8	57.0	57.6	51.1
segDPM+CNN	√				72.8	64.1	50.7	32.1	28.2	64.9	55.9	72.4	27.7	50.6	31.7	65.9	59.3	71.1	57.1	26.5	59.4	38.8	57.1	57.6	52.2
segDeepM	√				73.8	64.0	52.4	32.7	28.2	66.4	56.7	73.1	28.1	51.4	34.0	66.1	59.9	71.0	56.6	29.5	59.5	43.9	61.6	58.0	53.3
segDeepM		√			72.2	65.2	52.4	36.3	29.4	67.3	59.0	71.0	28.9	49.1	30.6	67.6	59.3	72.6	59.1	28.7	60.6	38.6	58.2	60.3	53.3
segDeepM			√		71.4	64.3	50.2	31.8	30.6	66.0	57.5	68.7	25.6	49.7	30.5	64.7	58.3	69.9	60.7	26.9	54.4	35.0	57.1	55.5	51.4
segDeepM	√	√			74.5	64.8	55.3	36.3	31.2	69.0	59.0	73.8	29.7	53.3	33.7	68.8	62.3	73.1	59.3	29.8	63.1	41.3	63.4	60.0	55.1
segDeepM	√	√	√		77.1	67.4	58.2	36.9	37.4	71.3	61.1	74.4	29.3	56.3	34.8	69.8	64.1	72.7	64.1	31.5	60.0	39.7	64.8	58.2	56.4
RCNN				√	72.9	65.8	54.0	34.5	31.2	68.0	59.8	72.3	26.6	51.3	35.0	65.7	59.7	71.7	60.7	28.0	60.6	37.1	60.3	59.9	53.7
segDeepM	√			√	76.9	66.8	57.8	36.2	32.2	71.4	60.0	75.4	27.7	53.8	38.6	68.6	64.4	72.6	61.1	30.4	61.5	43.2	64.1	60.9	56.2
segDeepM		√		√	76.7	68.7	58.0	39.9	34.6	71.1	62.1	**75.9**	30.3	54.6	36.2	69.6	63.3	74.0	63.5	31.3	62.5	37.9	66.3	61.0	56.9
segDeepM			√	√	77.2	66.6	55.2	34.5	34.6	67.5	60.0	70.4	27.1	53.4	35.9	66.4	63.4	71.9	63.0	32.0	55.7	38.5	62.0	58.0	54.7
segDeepM	√	√		√	77.2	67.6	59.8	40.2	35.7	**72.0**	62.1	75.7	30.4	**58.1**	37.2	69.9	64.8	**73.9**	63.4	32.4	**63.9**	43.1	**68.4**	61.6	57.9
segDeepM	√	√	√	√	**79.0**	**70.6**	**61.9**	40.4	**39.0**	71.6	61.9	74.7	**31.3**	56.6	**39.2**	70.4	66.5	73.5	65.6	35.3	60.7	44.3	68.0	58.7	**58.5**

Table 1: Detection results (in % AP) on PASCAL VOC 2010 *val* for R-CNN and segDeepM detectors.

.

	seg	exp	ibr	br	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
RCNN					74.4	69.0	55.6	34.5	35.2	70.8	63.0	81.4	35.0	57.9	39.3	77.7	70.2	75.8	61.7	29.1	66.9	56.7	63.8	58.7	58.8
segDeepM	√				77.5	67.7	59.0	33.4	35.9	71.3	62.4	82.8	35.6	61.4	42.7	79.1	70.7	76.7	62.3	31.3	67.1	54.0	65.5	60.0	59.8
segDeepM		√			77.4	72.9	62.6	36.8	39.4	71.5	64.9	83.7	37.1	62.0	40.4	81.0	73.1	77.9	65.7	34.7	68.0	59.1	70.0	58.7	61.8
segDeepM	√	√			79.2	69.8	63.6	36.4	39.5	72.9	65.2	83.5	38.4	63.6	43.8	80.8	75.1	78.3	66.2	33.3	68.6	56.0	70.7	60.2	62.2
RCNN				√	78.6	72.1	62.1	40.4	40.0	71.2	65.0	84.2	36.7	59.5	41.8	80.3	74.5	78.0	65.8	33.0	67.3	59.9	68.7	61.3	62.0
segDeepM	√			√	79.0	71.3	63.0	38.9	40.0	72.8	63.4	84.9	36.7	62.1	44.3	80.2	76.0	78.7	66.0	35.8	68.3	57.8	67.5	62.0	62.4
segDeepM		√		√	**81.6**	**74.6**	**66.6**	41.6	44.6	70.7	**68.0**	84.9	39.7	62.5	44.2	**84.1**	**77.1**	79.2	69.9	35.7	67.6	**60.9**	**72.7**	**61.3**	64.4
segDeepM	√	√		√	81.5	73.4	**66.6**	40.4	**44.7**	71.3	67.5	**85.1**	**40.9**	**62.9**	**46.0**	83.5	76.9	**80.0**	**70.0**	**37.1**	**68.9**	60.4	72.2	61.1	**64.5**

Table 2: Detection results (in % AP) on PASCAL VOC 2010 *val* for RCNN and segDeepM using 16 layer OxfordNet CNN.

.

	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
segDeepM-16 layers	**82.3**	**75.2**	**67.1**	**50.7**	**49.8**	**71.1**	**69.5**	**88.2**	**42.5**	**71.2**	**50.0**	**85.7**	**76.6**	**81.8**	**69.3**	**41.5**	**71.9**	**62.2**	**73.2**	**64.6**	**67.2**
segDeepM-8 layers	75.3	69.7	57.6	44.2	42.1	62.2	64.7	74.8	30.1	55.6	43.1	70.7	66.4	72.6	63.5	31.9	61.9	46.1	64.4	58.1	57.8
BabyLearning	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN (breg)-16 ly.	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
R-CNN-16 layers	76.5	70.4	58.0	40.2	39.6	61.8	63.7	81.0	36.2	64.5	45.7	80.5	71.9	74.3	60.6	31.5	64.7	52.5	64.6	57.2	59.8
Feature Edit	74.8	69.2	55.7	41.9	36.1	64.7	62.3	69.5	31.3	53.3	43.7	69.9	64.0	71.8	60.5	32.7	63.0	44.1	63.6	56.6	56.4
R-CNN (breg)	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	53.7
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2

Table 3: State-of-the-art detection results (in %AP) on PASCAL VOC 2010 *test*. The 16 layer models adopt OxforNet, the rest use 8-layer AlexNet.

.

	seg	exp	ibr	br	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	mAP
RCNN					68.9	63.5	45.6	29.1	26.7	64.4	55.6	69.5	26.3	50.3	36.1	62.2	55.6	68.7	56.0	27.6	54.8	40.2	54.0	60.5	50.8
segDeepM	√	√	√		75.1	67.6	56.6	37.9	34.6	73.8	60.8	76.1	27.6	55.4	39.1	68.9	63.7	**72.4**	63.8	33.5	60.8	45.9	65.7	60.9	57.0
RCNN				√	74.0	66.7	50.4	32.9	31.5	68.0	58.4	74.7	26.9	52.9	39.3	67.0	59.2	71.5	59.1	29.9	56.8	44.2	63.1	63.7	54.5
segDeepM	√	√	√	√	**77.3**	**70.2**	**60.2**	**39.8**	**38.3**	**75.2**	**62.3**	**76.1**	**29.4**	**55.7**	**40.8**	**70.5**	**66.4**	**72.4**	**65.8**	**35.6**	**60.9**	**46.1**	**66.7**	**63.5**	**58.7**

Table 4: Detection results (in % AP) on PASCAL VOC 2012 *val* for RCNN and segDeepM detectors.

.

Our experiments show that this procedure converges to a set of stable boxes after two iterations.

4. Experimental Evaluation

We evaluate our method on the main object detection benchmark PASCAL VOC. We provide a details ablative study of different potentials and choices in our model in Subsec. 4.1. In Subsec. 4.2 we test our method on PASCAL's held-out test set and compare it to the current state-of-the-art methods.

4.1. A Detailed Analysis of Our Model on Val

We first evaluate our detection performance on *Val* set of the PASCAL VOC 2010 detection dataset. We train all methods on the *train* subset and evaluate the detection performance using the standard PASCAL criterion. We provide a detailed performance analysis of each proposed potential function, which we denote with *seg* (segmentation) and expanded network *exp* (the contextual network) in Table 1. We also compare our iterative bounding box regression approach, referred to as *ibr*, to the standard bounding box regression, referred to as *br*, [9].

R-CNN [9] serves as our main baseline. To better justify our model, we provide an additional baseline, where we simply augment the set of Selective Search boxes used originally by the R-CNN with the CPMC proposal set. We call this approach RCNN+CPMC in the Table (second row). To contrast our model with segDPM, which originally uses segmentation features in a DPM-style formulation, we simplify our model to use their exact features. Instead of HOG, however, we use CNNs for a fair comparison. We also use their approach to generate segments, by finding connected

(a) PR curve for plane (b) PR curve for bicycle (c) PR curve for bird (d) PR curve for boat (e) PR curve for bottle

(f) PR curve for bus (g) PR curve for car (h) PR curve for cat (i) PR curve for chair (j) PR curve for cow
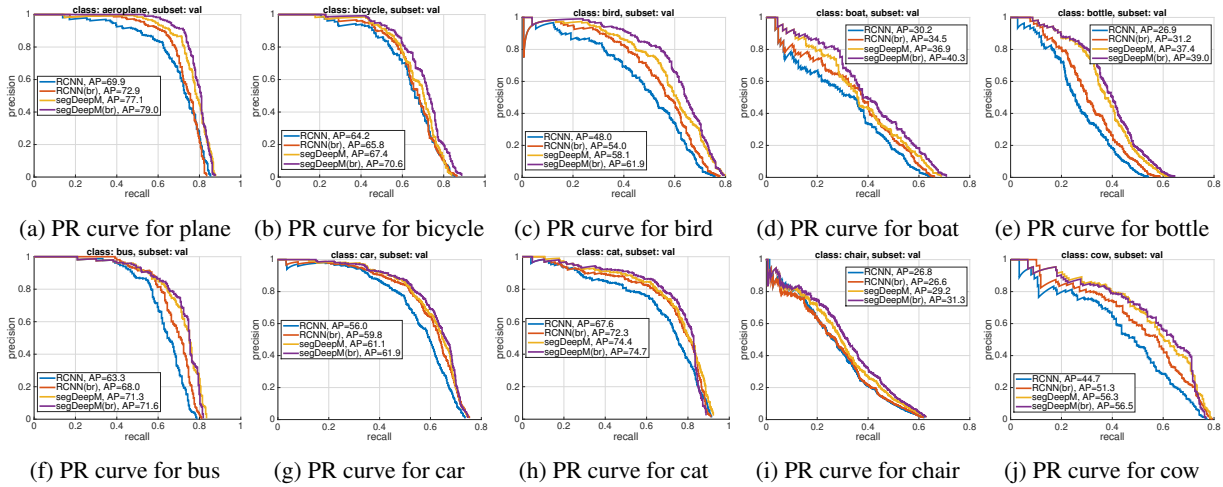
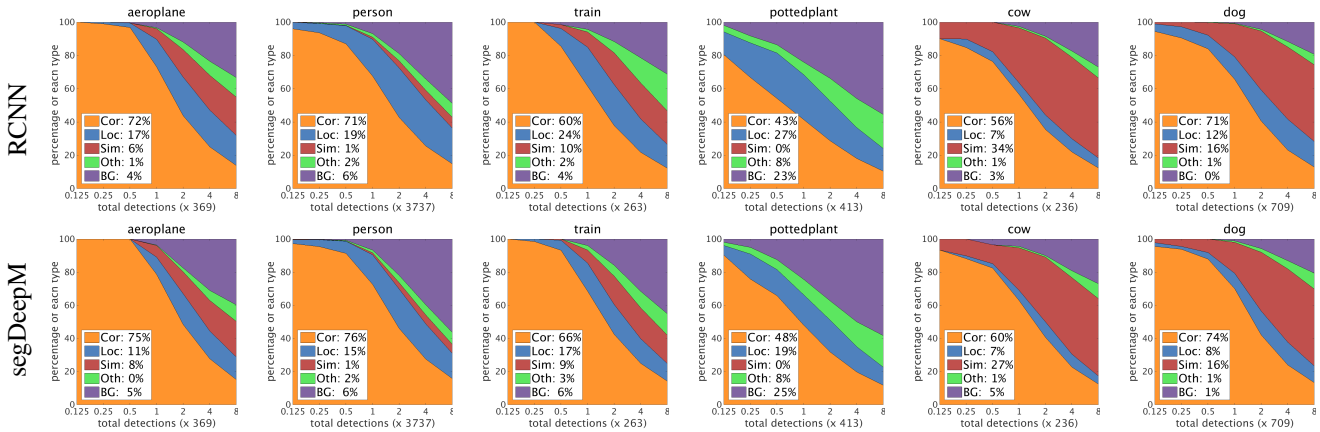Figure 2: PR curves on PASCAL VOC 2010 *train/val* split for 10 classes. All plots are in Suppl. material.

Figure 3: Error analysis in style of [12] of our detector compared to RCNN.

components in the final output of CPMC-O2P segmentation [8]. This approach is referred to as segDPM+CNN (third row in Table 1).

Observe that using a small set of additional segments brings a 1% improvement for RCNN+CPMC over the R-CNN baseline. Using a segDPM+CNN approach yields a more significant improvement of 2.1%. With our segmentation features we get an additional 1.1% increase over segDPM+CNN, thus justifying our feature set. Interestingly, this 3% boost over R-CNN is achieved by our simple segmentation features which require only 223 additional parameters. The Table also shows a steady improvement of each additional added potential/step, with the highest contribution achieved by the expanded contextual network.

Our full approach, in the setting without any post-processing, outperforms the strong baseline detector [9] by 6.3%, a significant improvement. After post-processing, the improvement is slightly lower, achieving a 4.8% performance gain. We note that we improve over the baseline in 19 out of 20 object classes. The PR curves for the first 10 classes are shown in Figure 2 and the qualitative results are shown in Figure 6. A detailed error analysis as proposed
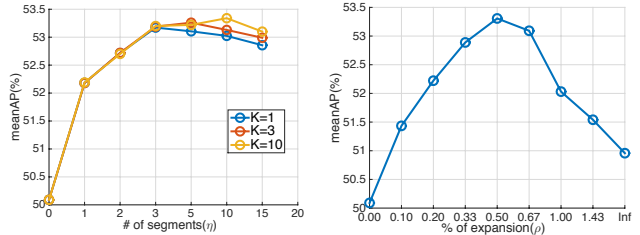
Figure 4: mAP for segDeepM w.r.t.**(left)** # of segments per image (η). $\eta = 0$ indicates no segments are used. **(right)** box expansion ratio ρ. $\rho = 0$ disables context and $\rho = \infty$ indicates full image context. Only contextual features used in this experiment. Both plots for PASCAL VOC 2010 *val*.

in [12] of R-CNN and our detector is shown in Figure 3.

Performance vs. grid size and # of segments. We evaluate the influence of different grid sizes and different number of CPMC segments per image. For each CPMC segment we compute the best O2P ranking score across all classes, and choose the top η segments according to these scores. Figure 4, left panel, shows that the highest performance gain

is due to the few best scoring segments. The differences are minor across different values of K and η. Interestingly, the model performs worse with more segments and a coarse grid, as additional low-quality segments add noise and make L-SVM training more difficult. When using a finer grid, the performance peaks when more segments are use, and achieves an overall improvement over a single-cell grid.

Performance w.r.t. expansion ratio. We evaluate the influence of the box expansion ratio ρ used in our contextual model. The results for varying values of ρ are illustrated in Figure 4, right panel. Note that even a small expansion ratio (10% in each direction) can boost the detection performance by a significant 1.5%, and the performance reaches its peak at $\rho = 0.5$. This indicates that richer contextual information leads to a better object recognition. Notice also that the detection performance decreases beyond $\rho = 0.5$. This is most likely due to the fact that most contextual boxes obtained this way will cover most or even the full image, and thus the positive and negative training instances in the same image will share the identical contextual features. This confuses our classifier and results in a performance loss. If we take the full image as context, the gain is less than 1%.

Iterative bounding box prediction. We next study the effect of iterative bounding box prediction. We report a 1.4% gain over the original R-CNN by starting with our set of re-localized boxes (one iteration). Note that re-localization in the first iteration only affects 52% of boxes (only 52% of boxes change more than 20% from the original set, thus feature re-computation only affects half of the boxes). This performance gain persists when combined with our full model. If we apply another bounding box prediction as a post-processing step, this approach still obtains a 0.6% improvement over R-CNN with bounding box prediction. In this iteration, re-localization affects 42% of boxes. We have noticed that the performance saturates after two iterations. The second iteration improves mAP by only a small margin (about 0.1%). The interesting side result is that, the mean Average Best Overlap (mABO) measure used by bottom-up proposal generation techniques [23] to benchmark their proposals, remains exactly the same (85.6%) with or without our bounding box prediction, but has a significant impact on the detection performance. This may indicate that mABO is not the best or at least not the only indicator of a good bottom-up grouping technique.

Missing annotations. An interesting issue arises when analyzing the top false-positives of our segDeepM. We have noticed that a non-neglible number of false-positives are due to missing annotations in PASCAL's ground-truth. Some examples are shown in Figure 5. These missed annotations are mostly due to small objects (Figure 5a, 5c),

(a) 3rd FP for "car" (b) 10th FP "car" (c) 16th FP "car" (d) 3rd FP "person"

Figure 5: Missing annotations detected by segDeepM. The red solid rectangle indicates segDeepM detection and dashed rectangle represents GT. We show all GT labels, included those marked as 'difficult'.
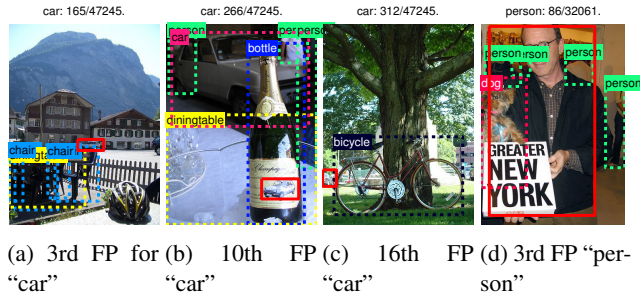
ambiguous definition of an "object" (Figure 5b), and labelers' mistakes (Figure 5d). While missing annotations were not an issue a few years ago when performance was at 30%, it is becoming a problem now, indicating that perhaps a re-annotation is needed.

4.2. Comparison with State-of-The-Art on Test

We evaluate our approach on the PASCAL VOC 2010 $test$ subset in Table 3. For this experiment we trained our segDeepM model, as well as its potentials (the CPMC class regressor) on the PASCAL VOC $trainval$ subset using the best parameters tuned on the $train/val$ split. We only submitted one result to the evaluation server, thus no tuning on the test set was involved. Table 3 shows results of our full segDeepM (including all post-processing steps). We achieve a 4.1% improvement over R-CNN with a 7-layer network, and a 1.4% over the best reported method using a 7-layer network. Notice that the best results on the current leader board are achieved by the recently released 16-layer network [21]. This network has 160 million parameters, compared to 60 million parameters used in our network. Our approach, with only a few additional parameters, scores rather high relative to the much larger network. Our result is "only" 2% lower than the very deep state-of-the-art.

We also run our method using a recently released 16-layer OxfordNet [21]. The results on $train/val$ and $trainval/test$ are shown in Table 2 and Table 3 respectively. On the $test$ set, our segDeepM achieves 67.2% mean AP and outperforms others in 20 out of 20 object classes.

Performance on PASCAL VOC 2012. We also test our full segDeepM model on PASCAL VOC 2012. We use the parameters tuned on the PASCAL VOC 2010 $train/val$ split. The result are reported and compared to the current state-of-the-art in Table 4.

5. Conclusion

We proposed a MRF model that scores appearance as well as context for each detection, and allows each can-
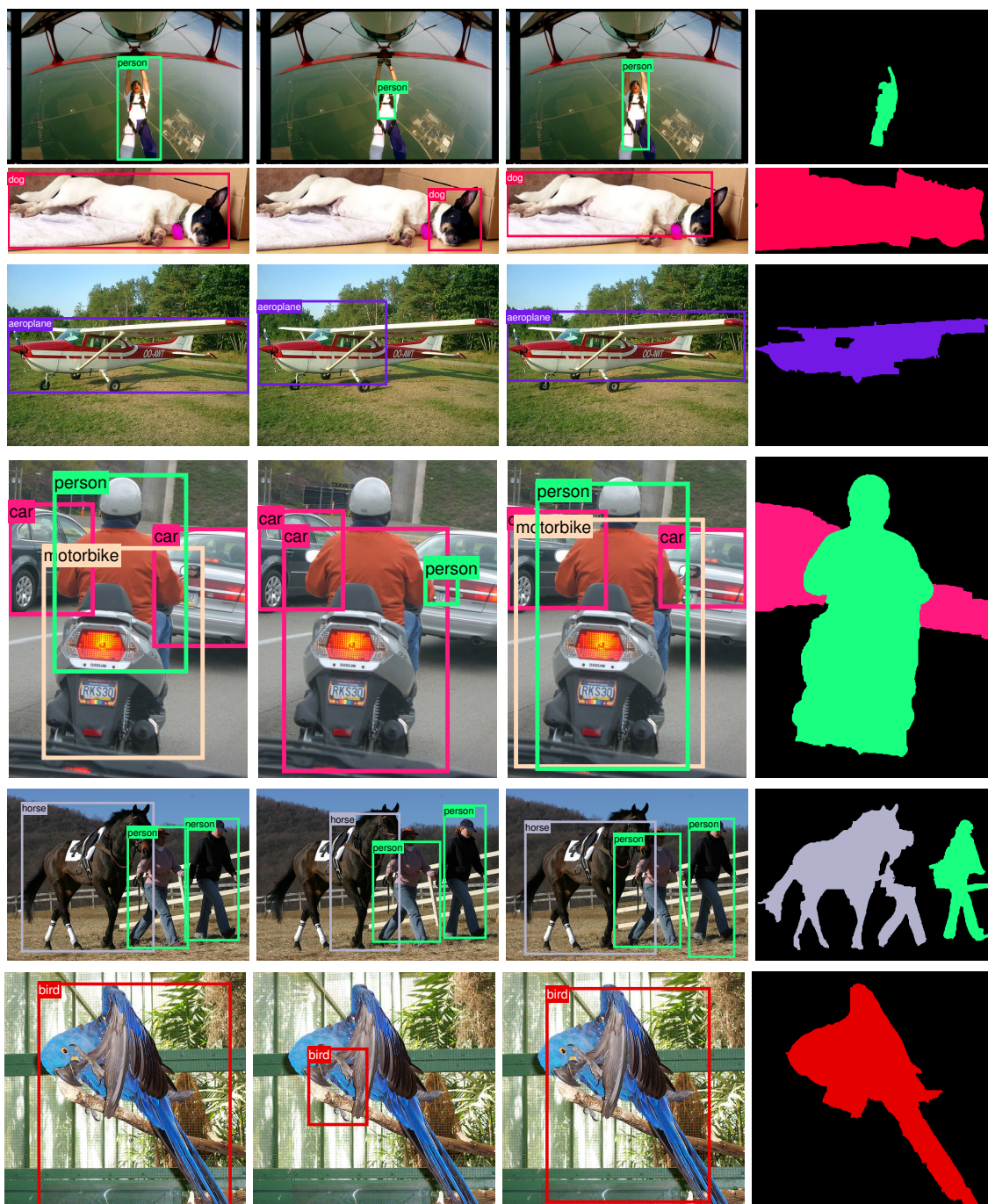
(a) GroundTruth (b) RCNN detection (c) segDeepM detection (d) best segments selected

Figure 6: Qualitative results. We show the top scoring detections for each ground-truth class. For our method, we also show the segment chosen by our model.

didate box to select a segment and score the agreement between them. We additionally proposed a sequential localization scheme, where we iterate between scoring our model and re-positioning the box (changing the spatial scope of the input to the model). We demonstrated that our approach achieves a significant boost over the RCNN base-line, 4.1% on PASCAL VOC 2010 test in the 7-layer setting and 4.3% in the 16-layer setting. The final result places segDeepM at the top of the current PASCAL's leaderboard.

References

[1] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting people using mutually consistent poselet activations. In *ECCV*, 2010. 2

[2] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*, pages 430–443. Springer, 2012. 1, 2, 3

[3] J. Carreira and C. Sminchisescu. Constrained parametric min-cuts for automatic object segmentation. In *CVPR*, pages 3241–3248. IEEE, 2010. 2

[4] X. Chen, R. Mottaghi, X. Liu, N.-G. Cho, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 2

[5] Q. Dai and D. Hoiem. Learning to localize detected objects. In *CVPR*, 2012. 2

[6] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In *ECCV*, pages 299–314. Springer, 2014. 2

[7] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 2, 4

[8] S. Fidler, R. Mottaghi, A. Yuille, and R. Urtasun. Bottom-up segmentation for top-down detection. In *CVPR*, 2013. 2, 3, 4, 6

[9] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *arXiv preprint arXiv:1311.2524*, 2013. 1, 2, 3, 4, 5, 6

[10] C. Gu, J. Lim, P. Arbelaez, and J. Malik. Recognition using regions. In *CVPR*, 2009. 2

[11] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. In *ECCV*, 2014. 1, 2

[12] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In *ECCV*, 2014. 6

[13] R. Kiros, R. Salakhutdinov, and R. Zemel. Multimodal neural language models. In *ICML*, 2014. 1

[14] P. Krähenbühl and V. Koltun. Efficient inference in fully connected crfs with gaussian edge potentials. In *NIPS*, 2011. 2

[15] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, pages 1097–1105, 2012. 1, 2, 3

[16] L. Ladicky, P. Sturgess, K. Alahari, C. Russell, and P. H. Torr. What, where and how many? combining object detectors and crfs. In *ECCV*, 2010. 2

[17] M. Maire, S. X. Yu, and P. Perona. Object detection and segmentation from joint embedding of parts and pixels. In *ICCV*, 2011. 2

[18] R. Memisevic and C. Conrad. Stereopsis via deep learning. In *NIPS*, 2011. 1

[19] 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. *CVPR*, 2014. 2

[20] O. Parkhi, A. Vedaldi, C. V. Jawahar, and A. Zisserman. The truth about cats and dogs. In *ICCV*, 2011. 2

[21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *arXiv:1409.1556*, 2014. 1, 7

[22] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D.Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *arXiv preprint arXiv:1409.4842*, 2014. 1

[23] K. E. Van de Sande, J. R. Uijlings, T. Gevers, and A. W. Smeulders. Segmentation as selective search for object recognition. In *ICCV*, pages 1879–1886. IEEE, 2011. 1, 2, 7

[24] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid. DeepFlow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 1

[25] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object models for image segmentation. *PAMI*, 2011. 2

[26] J. Yao, S. Fidler, and R. Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 2