

Simultaneous Detection and Segmentation

Bharath Hariharan¹, Pablo Arbeláez^{1,2}, Ross Girshick¹, and Jitendra Malik¹
{bharath2,arbelaez,rbg,malik}@eecs.berkeley.edu

¹University of California, Berkeley
²Universidad de los Andes, Colombia

Abstract. We aim to detect all instances of a category in an image and, for each instance, mark the pixels that belong to it. We call this task Simultaneous Detection and Segmentation (SDS). Unlike classical bounding box detection, SDS requires a segmentation and not just a box. Unlike classical semantic segmentation, we require individual object instances. We build on recent work that uses convolutional neural networks to classify category-independent region proposals (R-CNN [16]), introducing a novel architecture tailored for SDS. We then use category-specific, top-down figure-ground predictions to refine our bottom-up proposals. We show a 7 point boost (16% relative) over our baselines on SDS, a 5 point boost (10% relative) over state-of-the-art on semantic segmentation, and state-of-the-art performance in object detection. Finally, we provide diagnostic tools that unpack performance and provide directions for future work.

Keywords: detection, segmentation, convolutional networks

1 Introduction

Object recognition comes in many flavors, two of the most popular being object detection and semantic segmentation. Starting with face detection, the task in object detection is to mark out bounding boxes around each object of a particular category in an image. In this task, a predicted bounding box is considered a true positive if it overlaps by more than 50% with a ground truth box, and different algorithms are compared based on their precision and recall. Object detection systems strive to find every instance of the category and estimate the spatial extent of each. However, the detected objects are very coarsely localized using just bounding boxes.

In contrast, semantic segmentation requires one to assign a category label to all pixels in an image. The MSRC dataset [30] was one of the first publicly available benchmarks geared towards this task. Later, the standard metric used to evaluate algorithms in this task converged on pixel IU (intersection over union): for each category, this metric computes the intersection over union of the predicted pixels and ground truth pixels over the entire dataset. This task deals with “stuff” categories (such as grass, sky, road) and “thing” categories (such as cow, person, car) interchangeably. For things, this means that there is no notion

of object instances. A typical semantic segmentation algorithm might accurately mark out the dog pixels in the image, but would provide no indication of how many dogs there are, or of the precise spatial extent of any one particular dog.

These two tasks have continued to this day and were part of the PASCAL VOC challenge [11]. Although often treated as separate problems, we believe the distinction between them is artificial. For the “thing” categories, we can think of a unified task: detect all instances of a category in an image and, for each instance, correctly mark the pixels that belong to it. Compared to the bounding boxes output by an object detection system or the pixel-level category labels output by a semantic segmentation system, this task demands a richer, and potentially more useful, output. Our aim in this paper is to improve performance on this task, which we call **Simultaneous Detection and Segmentation** (SDS).

The SDS algorithm we propose has the following steps (Figure 1):

1. **Proposal generation:** We start with category-independent bottom-up object proposals. Because we are interested in producing segmentations and not just bounding boxes, we need *region* proposals. We use MCG [1] to generate 2000 region candidates per image. We consider each region candidate as a putative object hypothesis.
2. **Feature extraction:** We use a convolutional neural network to extract features on each region. We extract features from both the bounding box of the region as well as from the region foreground. This follows work by Girshick *et al.* [16] (R-CNN) who achieved competitive semantic segmentation results and dramatically improved the state-of-the-art in object detection by using CNNs to classify region proposals. We consider several ways of training the CNNs. We find that, compared to using the same CNN for both inputs (image windows and region masks), using separate networks where each network is finetuned for its respective role dramatically improves performance. We improve performance further by training both networks jointly, resulting in a feature extractor that is trained end-to-end for the SDS task.
3. **Region classification:** We train an SVM on top of the CNN features to assign a score for each category to each candidate.
4. **Region refinement:** We do non-maximum suppression (NMS) on the scored candidates. Then we use the features from the CNN to produce category-specific coarse mask predictions to refine the surviving candidates. Combining this mask with the original region candidates provides a further boost.

Since this task is not a standard one, we need to decide on evaluation metrics. The metric we suggest in this paper is an extension to the bounding box detection metric. It has been proposed earlier [31,32]. Given an image, we expect the algorithm to produce a set of object hypotheses, where each hypothesis comes with a predicted *segmentation* and a score. A hypothesis is correct if its *segmentation* overlaps with the *segmentation* of a ground truth instance by more than 50%. As in the classical bounding box task, we penalize duplicates. With this labeling, we compute a precision recall (PR) curve, and the average precision (AP), which is the area under the curve. We call the AP computed in this way

AP^r , to distinguish it from the traditional bounding box AP, which we call AP^b (the superscripts r and b correspond to region and bounding box respectively). AP^r measures the accuracy of segmentation, and also requires the algorithm to get each instance separately and completely. Our pipeline achieves an AP^r of 49.5% while at the same time improving AP^b from 51.0% (R-CNN) to 53.0%.

One can argue that the 50% threshold is itself artificial. For instance if we want to count the number of people in a crowd, we do not need to know their accurate segmentations. On the contrary, in a graphics application that seeks to matte an object into a scene, we might want extremely accurate segmentations. Thus the threshold at which we regard a detection as a true positive depends on the application. In general, we want algorithms that do well under a variety of thresholds. As the threshold varies, the PR curve traces out a PR surface. We can use the volume under this PR surface as a metric. We call this metric AP_{vol}^r and AP_{vol}^b respectively. AP_{vol}^r has the attractive property that an AP_{vol}^r of 1 implies we can perfectly detect and precisely segment all objects. Our pipeline gets an AP_{vol}^r of 41.4%. We improve AP_{vol}^b from 41.9% (R-CNN) to 44.2%.

We also find that our pipeline furthers the state-of-the-art in the classic PASCAL VOC semantic segmentation task, from 47.9% to 52.6%. Last but not the least, following work in object detection [18], we also provide a set of diagnostic tools for analyzing common error modes in the SDS task. Our algorithm, the benchmark and all diagnostic tools are publicly available at <http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/sds>.

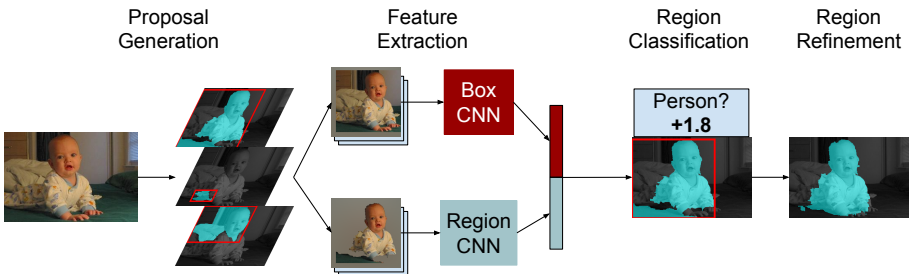


Fig. 1. Overview of our pipeline. Our algorithm is based on classifying region proposals using features extracted from both the bounding box of the region and the region foreground with a jointly trained CNN. A final refinement step improves segmentation.

2 Related work

For semantic segmentation, several researchers have tried to use activations from off-the-shelf object detectors to guide the segmentation process. Yang *et al.* [32]

use object detections from the deformable parts model [13] to segment the image, pasting figure-ground masks and reasoning about their relative depth ordering. Arbeláez *et al.* [2] use poselet detections [4] as features to score region candidates, in addition to appearance-based cues. Ladicky *et al.* [22] use object detections as higher order potentials in a CRF-based segmentation system: all pixels in the foreground of a detected object are encouraged to share the category label of the detection. In addition, their system is allowed to switch off these potentials by assigning a true/false label to each detection. This system was extended by Boix *et al.* [3] who added a global, image-level node in the CRF to reason about the categories present in the image, and by Kim *et al.* [20] who added relationships between objects. In more recent work, Tighe *et al.* [31] use exemplar object detectors to segment out the scene as well as individual instances.

There has also been work on localizing detections better using segmentation. Parkhi *et al.* use color models from predefined rectangles on cat and dog faces to do GrabCut and improve the predicted bounding box [26]. Dai and Hoiem generalize this to all categories and use instance and category appearance models to improve detection [7]. These approaches do well when the objects are coherent in color or texture. This is not true of many categories such as people, where each object can be made of multiple regions of different appearance. An alternative to doing segmentation *post facto* is to use segmentation to generate object proposals which are then classified. The proposals may be used as just bounding boxes [27] or as region proposals [6,1]. These proposals incorporate both the consistency of appearance in an object as well as the possibility of having multiple disparate regions for each object. State-of-the-art detection systems [16] and segmentation systems [5] are now based on these methods.

In many of these approaches, segmentation is used only to localize the detections better. Other authors have explored using segmentation as a stronger cue. Fidler *et al.* [14] use the output of a state-of-the-art semantic segmentation approach [5] to score detections better. Mottaghi [25] uses detectors based on non-rectangular patches to both detect and segment objects.

The approaches above were typically built on features such as SIFT[24] or HOG[8]. Recently the computer vision community has shifted towards using convolutional neural networks (CNNs). CNNs have their roots in the Neocognitron proposed by Fukushima [15]. Trained with the back-propagation algorithm, LeCun [23] showed that they could be used for handwritten zip code recognition. They have since been used in a variety of tasks, including detection [29,28] and semantic segmentation [12]. Krizhevsky *et al.* [21] showed a large increase in performance by using CNNs for classification in the ILSVRC challenge [9]. Donahue *et al.* [10] showed that Krizhevsky’s architecture could be used as a generic feature extractor that did well across a wide variety of tasks. Girshick *et al.* [16] build on this and finetune Krizhevsky’s architecture for detection to nearly double the state-of-the-art performance. They use a simple pipeline, using CNNs to classify bounding box proposals from [27]. Our algorithm builds on this system, and on high quality region proposals from [1].

3 Our approach

3.1 Proposal generation

A large number of methods to generate proposals have been proposed in the literature. The methods differ on the type of outputs they produce (boxes vs segments) and the metrics they do well on. Since we are interested in the AP^r metric, we care about segments, and not just boxes. Keeping our task in mind, we use candidates from MCG [1] for this paper. This approach significantly outperforms all competing approaches on the object level Jaccard index metric, which measures the average best overlap achieved by a candidate for a ground truth object. In our experiments we find that simply switching to MCG from Selective Search [27] improves AP^b slightly (by 0.7 points), justifying this choice.

We use the proposals from MCG as is. MCG starts by computing a segmentation hierarchy at multiple image resolutions, which are then fused into a single multiscale hierarchy at the finest scale. Then candidates are produced by combinatorially grouping regions from all the single scale hierarchies and from the multiscale hierarchy. The candidates are ranked based on simple features such as size and location, shape and contour strength.

3.2 Feature extraction

We start from the R-CNN object detector proposed by Girshick *et al.* [16] and adapt it to the SDS task. Girshick *et al.* train a CNN on ImageNet Classification and then finetune the network on the PASCAL detection set. For finetuning they took bounding boxes from Selective Search, padded them, cropped them and warped them to a square and fed them to the network. Bounding boxes that overlap with the ground truth by more than 50% were taken as positives and other boxes as negatives. The class label for each positive box was taken to be the class of the ground truth box that overlaps the most with the box. The network thus learned to predict if the bounding box overlaps highly with a ground truth bounding box. We are working with MCG instead of Selective Search, so we train a similar object detection network, finetuned using bounding boxes of MCG regions instead of Selective Search boxes.

At test time, to extract features from a bounding box, Girshick *et al.* pad and crop the box, warp it to a square and pass it through the network, and extract features from one of the later layers, which is then fed into an SVM. In this paper we will use the penultimate fully connected layer.

For the SDS task, we can now use this network finetuned for detection to extract feature vectors from MCG bounding boxes. However these feature vectors do not contain any information about the actual region foreground, and so will be ill-equipped to decide if the region overlaps highly with a ground truth segmentation or not. To get around this, we start with the idea used by Girshick *et al.* for their experiment on semantic segmentation: we extract a second set of features from the region by feeding it the cropped, warped box, but with

the background of the region masked out (with the mean image.) Concatenating these two feature vectors together gives us the feature vector we use. (In their experiments Girshick *et al.* found both sets of features to be useful.) This method of extracting features out of the region is the simplest way of extending the object detection system to the SDS task and forms our baseline. We call this feature extractor **A**.

The network we are using above has been finetuned to classify bounding boxes, so its use in extracting features from the region foreground is suboptimal. Several neurons in the network may be focussing on context in the background, which will be unavailable when the network is fed the region foreground. This suggests that we should use a different network to extract the second set of features: one that is finetuned on the kinds of inputs that it is going to see. We therefore finetune another network (starting again from the net trained on ImageNet) which is fed as input cropped, padded bounding boxes of MCG regions with the background masked out. Because this region sees the actual foreground, we can actually train it to predict region overlap instead, which is what we care about. Therefore we change the labeling of the MCG regions to be based on segmentation overlap of the region with a ground truth region (instead of overlap with bounding box). We call this feature extractor **B**.

The previous strategy is still suboptimal, because the two networks have been trained in isolation, while at test time the two feature sets are going to be combined and fed to the classifier. This suggests that one should train the networks jointly. We formalize this intuition as follows. We create a neural network with the architecture shown in Figure 2. This architecture is a single network with two pathways. The first pathway operates on the cropped bounding box of the region (the “box” pathway) while the second pathway operates on the cropped bounding box with the background masked (the “region” pathway). The two pathways are disjoint except at the very final classifier layer, which concatenates the features from both pathways. Both these pathways individually have the same architecture as that of Krizhevsky *et al.* Note that both **A** and **B** can be seen as instantiations of this architecture, but with different sets of weights. **A** uses the same network parameters for both pathways. For **B**, the box pathway gets its weights from a network finetuned separately using bounding box overlap, while the region pathway gets its parameters from a network finetuned separately using region overlap.

Instead of using the same network in both pathways or training the two pathways in isolation, we now propose to train it as a whole directly. We use segmentation overlap as above. We initialize the box pathway with the network finetuned on boxes and the region pathway with the network finetuned on regions, and then finetune the entire network. At test time, we discard the final classification layer and use the output of the penultimate layer, which concatenates the features from the two pathways. We call this feature extractor **C**.

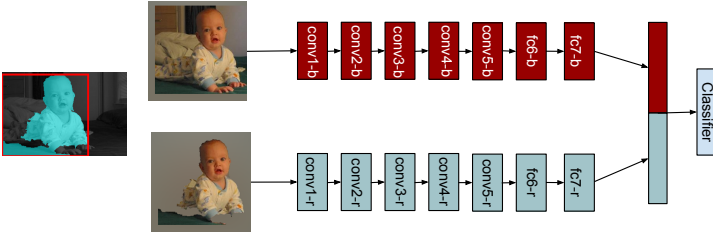


Fig. 2. Left: The region with its bounding box. Right: The architecture that we train for **C**. The top pathway operates on cropped boxes and the bottom pathway operates on region foregrounds.

3.3 Region classification

We use the features from the previous step to train a linear SVM. We first train an initial SVM using ground truth as positives and regions overlapping ground truth by less than 20% as negative. Then we re-estimate the positive set: for each ground truth we pick the highest scoring MCG candidate that overlaps by more than 50%. Ground truth regions for which no such candidate exists (very few in number) are discarded. We then retrain the classifier using this new positive set. This training procedure corresponds to a multiple instance learning problem where each ground truth defines a positive bag of regions that overlap with it by more than 50%, and each negative region is its own bag. We found this training to work better than using just the ground truth as positives.

At test time we use the region classifiers to score each region. Because there may be multiple overlapping regions, we do a strict non-max suppression using a region overlap threshold of 0. This is because while the bounding box of two objects can in fact overlap, their pixel support in the image typically shouldn't. Post NMS, we work with only the top 20,000 detections for each category (over the whole dataset) and discard the rest for computational reasons. We confirmed that this reduction in detections has no effect on the AP^r metric.

3.4 Region refinement

We take each of the remaining regions and refine its support. This is necessary because our region candidates have been created by a purely bottom-up, class agnostic process. Since the candidate generation has not made use of category-specific shape information, it is prone to both undershooting (*i.e.* missing some part of the object) and overshooting (*i.e.* including extraneous stuff).

We first learn to predict a coarse, top-down figure-ground mask for each region. To do this, we take the bounding box of each predicted region, pad it as for feature extraction, and then discretize the resulting box into a 10×10 grid. For each grid cell we train a logistic regression classifier to predict the probability that the grid cell belongs to the foreground. The features we use are the features extracted from the CNN, together with the figure-ground mask of the region

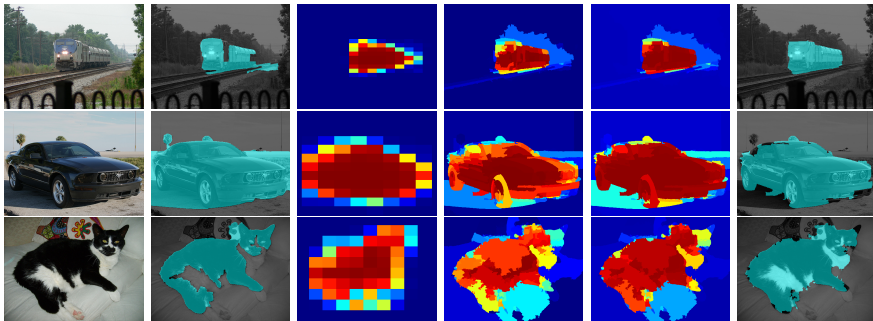


Fig. 3. Some examples of region refinement. We show in order the image, the original region, the coarse 10×10 mask, the coarse mask projected to superpixels, the output of the final classifier on superpixels and the final region after thresholding. Refinement uses top-down category specific information to fill in the body of the train and the cat and remove the road from the car.

discretized to the same 10×10 grid. The classifiers are trained on regions from the training set that overlap by more than 70% with a ground truth region.

This coarse figure-ground mask makes a top-down prediction about the shape of the object but does not necessarily respect the bottom-up contours. In addition, because of its coarse nature it cannot do a good job of modeling thin structures like aircraft wings or structures that move around. This information needs to come from the bottom-up region candidate. Hence we train a second stage to combine this coarse mask with the region candidate. We project the coarse mask to superpixels by assigning to each superpixel the average value of the coarse mask in the superpixel. Then we classify each superpixel, using as features this projected value in the superpixel and a 0 or 1 encoding if the superpixel belongs to the original region candidate. Figure 3 illustrates this refinement.

4 Experiments and results

We use the segmentation annotations from SBD [17] to train and evaluate. We train all systems on PASCAL VOC 2012 train. For all training and finetuning of the network we use the recently released Caffe framework [19].

4.1 Results on AP^r and AP_{vol}^r

Table 1 and Table 2 show results on the AP^r and the AP_{vol}^r metrics respectively on PASCAL VOC 2012 val (ground truth segmentations are not available for test). We compute AP_{vol}^r by averaging the AP^r obtained for 9 thresholds.

1. **O₂P** uses features and regions from Carreira *et al.* [5], which is the state-of-the-art in semantic segmentation. We train region classifiers on these features and do NMS to get detections. This baseline gets a mean AP^r of 25.2% and a mean AP_{vol}^r of 23.4%.

2. **A** is our most naive feature extractor. It uses MCG candidates and features from the bounding box and region foreground, using a single CNN finetuned using box overlaps. It achieves a mean AP^r of 42.9% and a mean AP^r_{vol} of 37.0%, a large jump over O₂P. This mirrors gains in object detection observed by Girshick *et al.* [16], although since O₂P is not designed for this task the comparison is somewhat unfair.
3. **B** is the result of finetuning a separate network exclusively on region foregrounds with labels defined by region overlap. This gives a large jump of the AP^r metric (of about 4 percentage points) and a smaller but significant jump on the AP^r_{vol} metric of about 2.5 percentage points.
4. **C** is the result of training a single large network with two pathways. There is a clear gain over using two isolated networks: on both metrics we gain about 0.7 percentage points.
5. **C+ref** is the result of refining the masks of the regions obtained from **C**. We again gain 2 points in the AP^r metric and 1.2 percentage points in the AP^r_{vol} metric. This large jump indicates that while MCG candidates we start from are very high quality, there is still a lot to be gained from refining the regions in a category specific manner.

A paired sample t-test indicates that each of the above improvements are statistically significant at the 0.05 significance level.

The left part of Figure 5 plots the improvement in mean AP^r over **A** as we vary the threshold at which a detection is considered correct. Each of our improvements increases AP^r across all thresholds, indicating that we haven’t overfit to a particular regime.

Clearly we get significant gains over both our naive baseline as well as O₂P. However, prior approaches that reason about segmentation together with detection might do better on the AP^r metric. To see if this is the case, we compare to the SegDPM work of Fidler *et al.* [14]. SegDPM combined DPMs [13] with O₂P [5] and achieved a 9 point boost over DPMs in classical object detection. For this method, only the bounding boxes are available publicly, and for some boxes the algorithm may choose not to have associated segments. We therefore compute an upper bound of its performance by taking each detection, considering all MCG regions whose bounding box overlaps with the detection by more than 70%, and selecting the region which best overlaps a ground truth.

Since SegDPM detections are only available on PASCAL VOC2010 val, we restrict our evaluations only to this set. Our upper bound on SegDPM has a mean AP^r of **31.3**, whereas **C+ref** achieves a mean AP^r of **50.3**.

4.2 Producing diagnostic information

Inspired by [18], we created tools for figuring out error modes and avenues for improvement for the SDS task. As in [18], we evaluate the impact of error modes by measuring the improvement in AP^r if the error mode was corrected. For localization, we assign labels to detections under two thresholds: the usual strict

Table 1. Results on AP^r on VOC2012 val. All numbers are %.

| | O ₂ P | A | B | C | C+ref |
|-------------|------------------|------|-------------|-------------|-------------|
| aeroplane | 56.5 | 61.8 | 65.7 | 67.4 | 68.4 |
| bicycle | 19.0 | 43.4 | 49.6 | 49.6 | 49.4 |
| bird | 23.0 | 46.6 | 47.2 | 49.1 | 52.1 |
| boat | 12.2 | 27.2 | 30.0 | 29.9 | 32.8 |
| bottle | 11.0 | 28.9 | 31.7 | 32.0 | 33.0 |
| bus | 48.8 | 61.7 | 66.9 | 65.9 | 67.8 |
| car | 26.0 | 46.9 | 50.9 | 51.4 | 53.6 |
| cat | 43.3 | 58.4 | 69.2 | 70.6 | 73.9 |
| chair | 4.7 | 17.8 | 19.6 | 20.2 | 19.9 |
| cow | 15.6 | 38.8 | 42.7 | 42.7 | 43.7 |
| diningtable | 7.8 | 18.6 | 22.8 | 22.9 | 25.7 |
| dog | 24.2 | 52.6 | 56.2 | 58.7 | 60.6 |
| horse | 27.5 | 44.3 | 51.9 | 54.4 | 55.9 |
| motorbike | 32.3 | 50.2 | 52.6 | 53.5 | 58.9 |
| person | 23.5 | 48.2 | 52.6 | 54.4 | 56.7 |
| pottedplant | 4.6 | 23.8 | 25.7 | 24.9 | 28.5 |
| sheep | 32.3 | 54.2 | 54.2 | 54.1 | 55.6 |
| sofa | 20.7 | 26.0 | 32.2 | 31.4 | 32.1 |
| train | 38.8 | 53.2 | 59.2 | 62.2 | 64.7 |
| tvmonitor | 32.3 | 55.3 | 58.7 | 59.3 | 60.0 |
| Mean | 25.2 | 42.9 | 47.0 | 47.7 | 49.7 |

Table 2. Results on AP^r_{vol} on VOC2012 val. All numbers are %.

| | O ₂ P | A | B | C | C+ref |
|-------------|------------------|------|------|-------------|-------------|
| aeroplane | 46.8 | 48.3 | 51.1 | 53.2 | 52.3 |
| bicycle | 21.2 | 39.8 | 42.1 | 42.1 | 42.6 |
| bird | 22.1 | 39.2 | 40.8 | 42.1 | 42.2 |
| boat | 13.0 | 25.1 | 27.5 | 27.1 | 28.6 |
| bottle | 10.1 | 26.0 | 26.8 | 27.6 | 28.6 |
| bus | 41.9 | 49.5 | 53.4 | 53.3 | 58.0 |
| car | 24.0 | 39.5 | 42.6 | 42.7 | 45.4 |
| cat | 39.2 | 50.7 | 56.3 | 57.3 | 58.9 |
| chair | 6.7 | 17.6 | 18.5 | 19.3 | 19.7 |
| cow | 14.6 | 32.5 | 36.0 | 36.3 | 37.1 |
| diningtable | 9.9 | 18.5 | 20.6 | 21.4 | 22.8 |
| dog | 24.0 | 46.8 | 48.9 | 49.0 | 49.5 |
| horse | 24.4 | 37.7 | 41.9 | 43.6 | 42.9 |
| motorbike | 28.6 | 41.1 | 43.2 | 43.5 | 45.9 |
| person | 25.6 | 43.2 | 45.8 | 47.0 | 48.5 |
| pottedplant | 7.0 | 23.4 | 24.8 | 24.4 | 25.5 |
| sheep | 29.0 | 43.0 | 44.2 | 44.0 | 44.5 |
| sofa | 18.8 | 26.2 | 29.7 | 29.9 | 30.2 |
| train | 34.6 | 45.1 | 48.9 | 49.9 | 52.6 |
| tvmonitor | 25.9 | 47.7 | 48.8 | 49.4 | 51.4 |
| Mean | 23.4 | 37.0 | 39.6 | 40.2 | 41.4 |

threshold of 0.5 and a more lenient threshold of 0.1 (note that this is a threshold on region overlap). Detections that count as true positives under the lenient threshold but as false positives under the strict threshold are considered mislocalizations. Duplicate detections are also considered mislocalizations. We then consider the performance if either a) all mislocalized instances were removed, or b) all mislocalized instances were correctly localized and duplicates removed.

Figure 4 shows how the PR curve for the AP^r benchmark changes if mislocalizations are corrected or removed for two categories. For the person category, removing mislocalizations brings precision up to essentially 100%, indicating that mislocalization is the predominant source of false positives. Correcting the mislocalizations provides a huge jump in recall. For the cat category the improvement provided by better localization is much less, indicating that there are still some false positives arising from misclassifications.

We can do this analysis for all categories. The average improvement in AP^r by fixing mislocalization is a measure of the impact of mislocalization on performance. We can also measure impact in this way for other error modes: for instance, false positives on objects of other similar categories, or on background [18]. (For defining similar and non-similar categories, we divide object categories into “animals”, “transport” and “indoor” groups.) The left subfigure in Figure 6 shows the result of such an analysis on our best system (C+ref). The dark blue bar shows the AP^r improvement if we remove mislocalized detections and the light blue bar shows the improvement if we correct them. The other two bars show the improvement from removing confusion with similar categories and background. Mislocalization has a huge impact: it sets us back by about 16 percentage points. Compared to that confusion with similar categories or background is virtually non-existent.

We can measure the impact of mislocalization on the other algorithms in Table 1 as well, as shown in Table 3. It also shows the upper bound AP^r achievable when all mislocalization is fixed. Improvements in the feature extractor improve the upper bound (indicating fewer misclassifications) but also reduce the gap due to mislocalization (indicating better localization). Refinement doesn’t change the upper bound and only improves localization, as expected.

To get a better handle on what one needs to do to improve localization, we considered two statistics. For each detection and a ground truth, instead of just taking the overlap (*i.e.* intersection over union), we can compute the pixel precision (fraction of the region that lies inside the ground truth) and pixel recall (fraction of the ground truth that lies inside the region). It can be shown that having both a pixel precision $> 67\%$ and a pixel recall $> 67\%$ is guaranteed to give an overlap of greater than 50%. We assign detection labels using pixel precision or pixel recall using a threshold of 67% and compute the respective AP. Comparing these two numbers then gives us a window into the kind of localization errors: a low pixel precision AP indicates that the error mode is overshooting the region and predicting extraneous background pixels, while a low pixel recall AP indicates that the error mode is undershooting the region and missing out some ground truth pixels.

The second half of Figure 6 shows the difference between pixel precision AP (AP^{pp}) and pixel recall AP (AP^{pr}). Bars to the left indicate higher pixel recall AP, while bars to the right indicate higher pixel precision AP. For some categories such as person and bird we tend to miss ground truth pixels, whereas for others such as bicycle we tend to leak into the background.

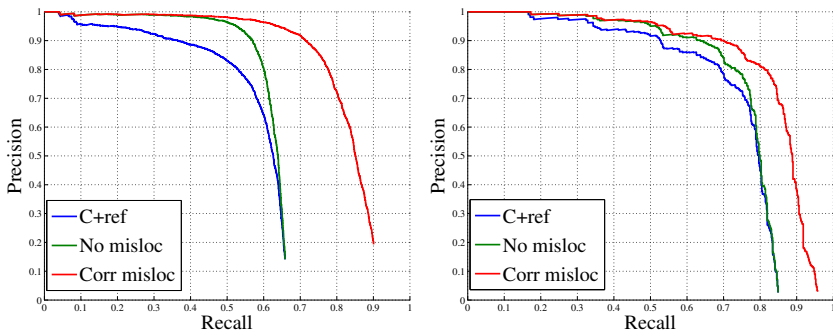


Fig. 4. PR on person(left) and cat(right). Blue is $C+ref$. Green is if an oracle removes mislocalized predictions, and red is if the oracle corrects our mislocalizations.

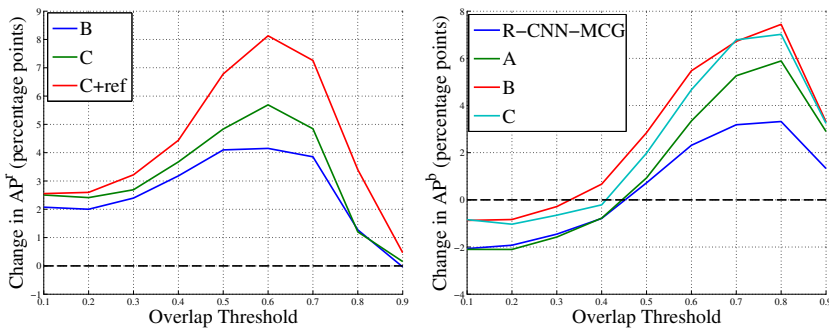


Fig. 5. Left: Improvement in mean AP^r over A due to our 3 variants for a variety of overlap thresholds. We get improvements for all overlap thresholds. Right: A similar plot for AP^b . Improvements are relative to R-CNN with Selective Search proposals [16]. As the threshold becomes stricter, the better localization of our approach is apparent.

4.3 Results on AP^b and AP_{vol}^b

Comparison with prior work is easier on the classical bounding box and segmentation metrics. It also helps us evaluate if handling the SDS task also improves

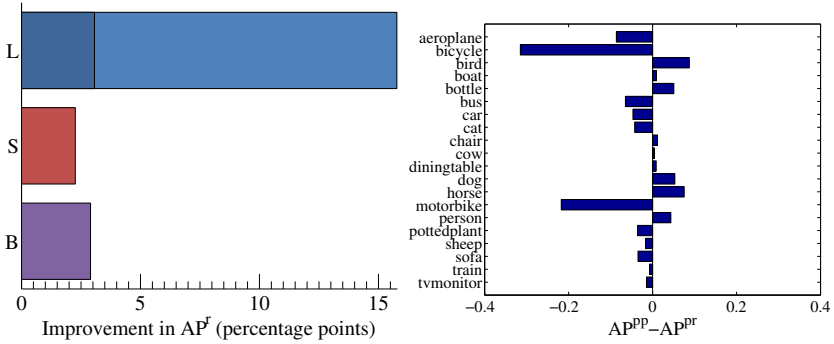


Fig. 6. Left: Impact of the three kinds of false positives on mean AP^r . L : mislocalization, B : detection on background, and S : misfirings on similar categories. Right: Disambiguating between two kinds of mislocalizations. Bars to the left mean that we frequently overshoot the ground truth, while bars to the right mean that we undershoot.

Table 3. Maximum achievable AP^r (assuming perfect localization) and loss in AP^r due to mislocalization for all systems.

| | A | B | C | C+ref |
|-----------------------------|------|------|------|-------|
| AP Upper bound | 63.0 | 65.0 | 65.4 | 65.5 |
| Loss due to mislocalization | 20.1 | 18.0 | 17.7 | 15.8 |

performance on the individual tasks. To compare on AP^b , we retrain our final region classifiers for the bounding box detection task. This is because the ranking of regions based on bounding box overlap is different from that based on segmentation overlap. As in [16], we use ground truth boxes as positive, and MCG boxes overlapping by less than 50% as negative. At test time we do not do any region refinement.

We add two baselines: R-CNN is the system of Girshick *et al.* taken as is, and R-CNN-MCG is R-CNN on boxes from MCG instead of Selective Search. Note that neither of these baselines uses features from the region foreground.

Table 4 shows the mean AP^b and AP_{vol}^b . We get improvements over R-CNN on both AP^b and AP_{vol}^b , with improvements on the latter metric being somewhat larger. The right half of Figure 5 shows the variation in AP^b as we vary the overlap threshold for counting something as correct. We plot the improvement in AP^b over vanilla R-CNN. We do worse than R-CNN for low thresholds, but are much better for higher thresholds. This is also true to some extent for R-CNN-MCG, so this is partly a property of MCG, and partly a consequence of our algorithm’s improved localization. Interestingly, **C** does worse than **B**. We posit that this is because now the entire network has been finetuned for SDS.

Finally we evaluated **C** on PASCAL VOC 2012 test. Our mean AP^b of **50.7** is an improvement over the R-CNN mean AP^b of **49.6** (both without bounding box regression), and much better than other systems, such as SegDPM [14] (**40.7**).

Table 4. Results on AP^b and AP_{vol}^b on VOC12 val. All numbers are %.

| | R-CNN[16] | R-CNN-MCG | A | B | C |
|-------------------|-----------|-----------|------|-------------|------|
| Mean AP^b | 51.0 | 51.7 | 51.9 | 53.9 | 53.0 |
| Mean AP_{vol}^b | 41.9 | 42.4 | 43.2 | 44.6 | 44.2 |

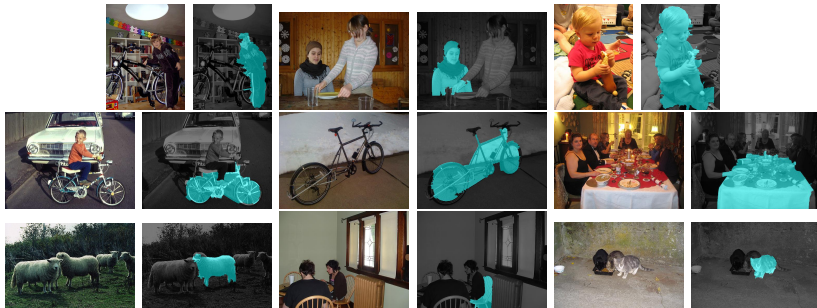
4.4 Results on pixel IU

For the semantic segmentation task, we convert the output of our final system (C+ref) into a pixel-level category labeling using the simple pasting scheme proposed by Carreira *et al.* [5]. We cross validate the hyperparameters of this pasting step on the VOC11 segmentation Val set. The results are in Table 5. We compare to O₂P [5] and R-CNN which are the current state-of-the-art on this task. We advance the state-of-the-art by about 5 points, or 10% relative.

To conclude, our pipeline achieves good results on the SDS task while improving state-of-the-art in object detection and semantic segmentation. Figure 7 shows examples of the output of our system.

Table 5. Results on Pixel IU. All numbers are %.

| | O ₂ P [5] | R-CNN [16] | C+ref |
|------------------------------|----------------------|------------|-------------|
| Mean Pixel IU (VOC2011 Test) | 47.6 | 47.9 | 52.6 |
| Mean Pixel IU (VOC2012 Test) | 47.8 | - | 51.6 |

**Fig. 7.** Top detections: 3 persons, 2 bikes, diningtable, sheep, chair, cat. We can handle uncommon pose and clutter and are able to resolve individual instances.

Acknowledgments. This work was supported by ONR MURI N000141010933, a Google Research Grant and a Microsoft Research fellowship. We thank the NVIDIA Corporation for providing GPUs through their academic program.

References

1. Arbeláez, P., Pont-Tuset, J., Barron, J., Marques, F., Malik, J.: Multiscale combinatorial grouping. In: CVPR (2014)
2. Arbeláez, P., Hariharan, B., Gu, C., Gupta, S., Malik, J.: Semantic segmentation using regions and parts. In: CVPR (2012)
3. Boix, X., Gonfalus, J.M., van de Weijer, J., Bagdanov, A.D., Serrat, J., González, J.: Harmony potentials. IJCV 96(1) (2012)
4. Bourdev, L., Maji, S., Brox, T., Malik, J.: Detecting people using mutually consistent poselet activations. In: ECCV (2010)
5. Carreira, J., Caseiro, R., Batista, J., Sminchisescu, C.: Semantic segmentation with second-order pooling. In: ECCV (2012)
6. Carreira, J., Sminchisescu, C.: Constrained parametric min-cuts for automatic object segmentation. In: CVPR (2010)
7. Dai, Q., Hoiem, D.: Learning to localize detected objects. In: CVPR (2012)
8. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)
9. Deng, J., Berg, A., Satheesh, S., Su, H., Khosla, A., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012). <http://www.image-net.org/challenges/LSVRC/2012/>
10. Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., Darrell, T.: Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531 (2013)
11. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal Visual Object Classes (VOC) Challenge. IJCV 88(2) (2010)
12. Farabet, C., Couprie, C., Najman, L., LeCun, Y.: Learning hierarchical features for scene labeling. TPAMI 35(8) (2013)
13. Felzenszwalb, P.F., Girshick, R.B., McAllester, D., Ramanan, D.: Object detection with discriminatively trained part-based models. TPAMI 32(9) (2010)
14. Fidler, S., Mottaghi, R., Yuille, A., Urtasun, R.: Bottom-up segmentation for top-down detection. In: CVPR (2013)
15. Fukushima, K.: Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. Biological cybernetics 36(4) (1980)
16. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: CVPR (2014)
17. Hariharan, B., Arbelaez, P., Bourdev, L., Maji, S., Malik, J.: Semantic contours from inverse detectors. In: ICCV (2011)
18. Hoiem, D., Chodpathumwan, Y., Dai, Q.: Diagnosing error in object detectors. In: ECCV (2012)
19. Jia, Y.: Caffe: An open source convolutional architecture for fast feature embedding. <http://caffe.berkeleyvision.org/> (2013)
20. Kim, B., Sun, M., Kohli, P., Savarese, S.: Relating things and stuff by high-order potential modeling. In: ECCV Workshop on Higher-Order Models and Global Constraints in Computer Vision (2012)
21. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS (2012)
22. Ladický, L., Sturgess, P., Alahari, K., Russell, C., Torr, P.H.: What, where and how many? combining object detectors and crfs. In: ECCV (2010)

23. LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D.: Backpropagation applied to handwritten zip code recognition. *Neural computation* 1(4) (1989)
24. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. *IJCV* 60(2) (2004)
25. Mottaghi, R.: Augmenting deformable part models with irregular-shaped object patches. In: *CVPR* (2012)
26. Parkhi, O.M., Vedaldi, A., Jawahar, C., Zisserman, A.: The truth about cats and dogs. In: *ICCV* (2011)
27. van de Sande, K.E., Uijlings, J.R., Gevers, T., Smeulders, A.W.: Segmentation as selective search for object recognition. In: *ICCV* (2011)
28. Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., LeCun, Y.: Overfeat: Integrated recognition, localization and detection using convolutional networks. In: *ICLR* (2014)
29. Sermanet, P., Kavukcuoglu, K., Chintala, S., LeCun, Y.: Pedestrian detection with unsupervised multi-stage feature learning. In: *CVPR* (2013)
30. Shotton, J., Winn, J., Rother, C., Criminisi, A.: Textonboost: Joint appearance, shape and context modeling for multi-class object recognition and segmentation. In: *ECCV* (2006)
31. Tighe, J., Niethammer, M., Lazebnik, S.: Scene parsing with object instances and occlusion handling. In: *ECCV* (2010)
32. Yang, Y., Hallman, S., Ramanan, D., Fowlkes, C.C.: Layered object models for image segmentation. *TPAMI* 34(9) (2012)