# Convolutional Feature Masking for Joint Object and Stuff Segmentation

Jifeng Dai        Kaiming He        Jian Sun

Microsoft Research

{jifdai,kahe,jiansun}@microsoft.com

## Abstract

*The topic of semantic segmentation has witnessed considerable progress due to the powerful features learned by convolutional neural networks (CNNs) [13]. The current leading approaches for semantic segmentation exploit shape information by extracting CNN features from masked image regions. This strategy introduces artificial boundaries on the images and may impact the quality of the extracted features. Besides, the operations on the raw image domain require to compute thousands of networks on a single image, which is time-consuming.*

*In this paper, we propose to exploit shape information via masking convolutional features. The proposal segments (e.g., super-pixels) are treated as masks on the convolutional feature maps. The CNN features of segments are directly masked out from these maps and used to train classifiers for recognition. We further propose a joint method to handle objects and "stuff" (e.g., grass, sky, water) in the same framework. State-of-the-art results are demonstrated on benchmarks of PASCAL VOC and new PASCAL-CONTEXT, with a compelling computational speed.*

## 1. Introduction

Semantic segmentation [14, 19, 24, 2] aims to label each image pixel to a semantic category. With the recent breakthroughs [13] by convolutional neural networks (CNNs) [15], R-CNN based methods [8, 10] for semantic segmentation have substantially advanced the state of the art.

The R-CNN methods [8, 10] for semantic segmentation extract two types of CNN features - one is *region features* [8] extracted from proposal bounding boxes [22]; the other is *segment features* extracted from the raw image content masked by the segments [10]. The concatenation of these features are used to train classifiers [10]. These methods have demonstrated compelling results on this long-standing challenging task.

However, the raw-image-based R-CNN methods [8, 10] have two issues. First, the masks on the image content can

lead to artificial boundaries. These boundaries do not exhibit on the samples during the network pre-training (*e.g.*, in the 1000-category ImageNet [5]). This issue may degrade the quality of the extracted segment features. Second, similar to the R-CNN method for object detection [8], these methods need to apply the network on thousands of raw image regions with/without the masks. This is very time-consuming even on high-end GPUs.

The second issue also exists in R-CNN based object detection. Fortunately, this issue can be largely addressed by a recent method called SPP-Net [11], which computes convolutional feature maps on the entire image only once and applies a spatial pyramid pooling (SPP) strategy to form cropped features for classification. The detection results via these cropped features have shown competitive detection accuracy [11], and the speed can be ∼50× faster. Therefore, in this paper, we raise a question: *for semantic segmentation, can we use the convolutional feature maps only?*

The first part of this work says *yes* to this question. We design a *convolutional feature masking* (CFM) method to extract *segment features* directly from feature maps instead of raw images. With the segments given by the region proposal methods (*e.g.*, selective search [22]), we project them to the domain of the last convolutional feature maps. The projected segments play as binary functions for masking the convolutional features. The masked features are then fed into the fully-connected layers for recognition. Because the convolutional features are computed from the unmasked image, their quality is not impacted. Besides, this method is efficient as the convolutional feature maps only need to be computed once. The aforementioned two issues involving semantic segmentation are thus both addressed. Figure 1 compares the raw-image-based pipeline and our feature-map-based pipeline.

The second part of this paper further generalizes our method for joint object and stuff segmentation [18]. Different from objects, "stuff" [18] (*e.g.*, sky, grass, water) is usually treated as the context in the image. Stuff mostly exhibits as colors or textures and has less well-defined shapes. It is thus inappropriate to use a single rectangular box or a single segment to represent stuff. Based on our masked
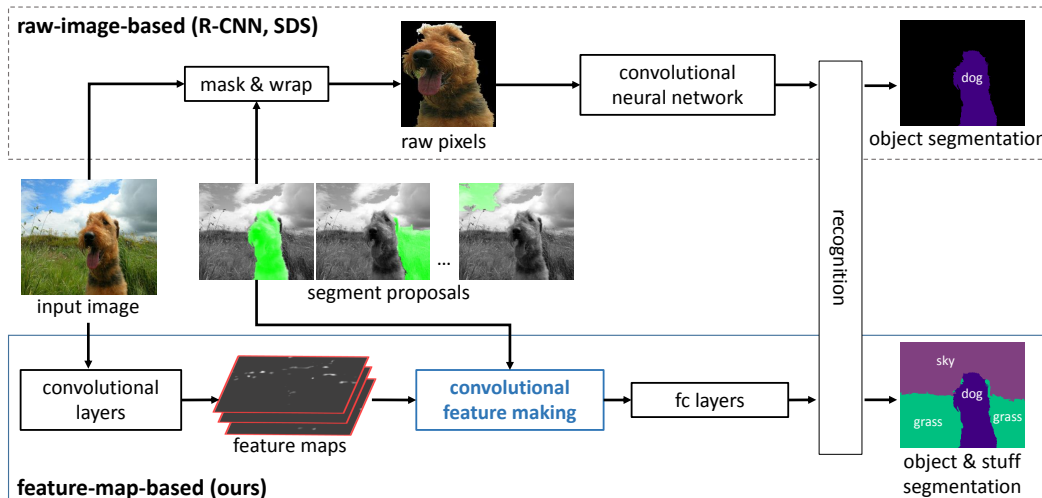
Figure 1: System pipeline. **Top**: the methods of "Regions with CNN features" (R-CNN) [8] and "Simultaneous Detection and Segmentation" (SDS) [10] that operate on the raw image domain. **Bottom**: our method that masks the convolutional feature maps.

convolutional features, we propose a training procedure that treats a stuff as a compact combination of multiple segment features. This allows us to address the object and stuff in the same framework.

Based on the above methods, we show state-of-the-art results on the PASCAL VOC 2012 benchmark [7] for object segmentation. Our method can process an image in a fraction of a second, which is ∼150× faster than the R-CNN-based SDS method [10]. Further, our method is also the *first* deep-learning-based method ever applied to the newly labeled PASCAL-CONTEXT benchmark [18] for both object and stuff segmentation, where our result substantially outperforms previous states of the art.

## 2. Convolutional Feature Masking

### 2.1. Convolutional Feature Masking Layer

The power of CNNs as a generic feature extractor has been gradually revealed in the computer vision area [13, 6, 25, 8, 11]. In Krizhevsky *et al.*'s work [13], they suggest that the features of the fully-connected layers can be used as *holistic* image features, *e.g.*, for image retrieval. In [6, 25], these holistic features are used as generic features for full-image classification tasks in other datasets via transfer learning. In the breakthrough object detection paper of R-CNN [8], the CNN features are also used like holistic features, but are extracted from sub-images which are the crops of raw images. In the CNN-based semantic segmentation paper [10], the R-CNN idea is generalized to masked raw image regions. For all these methods, the entire network is treated as a *holistic* feature extractor, either on the entire image or on sub-images.

In the recent work of SPP-Net [11], it shows that the convolutional feature maps can be used as *localized* features. On a full-image convolutional feature map, the local rectangular regions encode both the semantic information (by strengths of activations) and spatial information (by positions). The features from these local regions can be pooled [11] directly for recognition.

The spatial pyramid pooling (SPP) in [11] actually plays two roles: 1) masking the feature maps by a rectangular region, outside which the activations are removed; 2) generating a fixed-length feature from this arbitrary sized region. So, if masking by rectangles can be effective, *what if we mask the feature maps by a fine segment with an irregular shape*?

The Convolutional Feature Masking (CFM) layer is thus developed. We first obtain the candidate segments (like super-pixels) on the raw image. Many regional proposal methods (*e.g.*, [22, 1]) are based on super-pixels. Each proposal box is given by grouping a few super-pixels. We call such a group as a segment proposal. So we can obtain the candidate segments together with their proposal boxes (referred to as "regions" in this paper) without extra effort. These segments are binary masks on the raw images.

Next we project these binary masks to the domain of the last convolutional feature maps. Because each activation in the convolutional feature maps is contributed by a receptive field in the image domain, we first project each activation onto the image domain as the center of its receptive field (following the details in [11]). Each pixel in the binary masks on the image is assigned to its nearest center of the receptive fields. Then these pixels are projected back onto
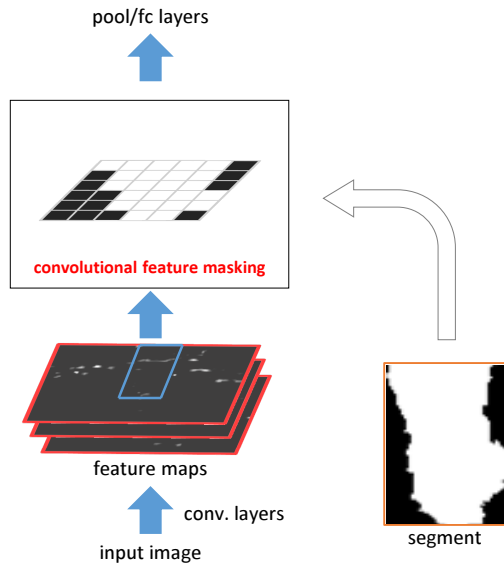
Figure 2: An illustration of the CFM layer.

the convolutional feature map domain based on this center and its activation's position. On the feature map, each position will collect multiple pixels projected from a binary mask. These binary values are then averaged and thresholded (by 0.5). This gives us a mask on the feature maps (Figure 2). This mask is then applied on the convolutional feature maps. Actually, we only need to multiply this binary mask on each channel of the feature maps. We call the resulting features as *segment features* in our method.

## 2.2. Network Designs

In [10], it has been shown that the segment features alone are insufficient. These segment features should be used together with the regional features (from bounding boxes) generated in a way like R-CNN [8]. Based on our CFM layer, we can have two possible ways of doing this.

**Design A: on the last convolutional layer.** As shown in Figure 3 (left part), after the last convolutional layer, we generate two sources of features. One is the regional feature produced by the SPP layer as in [11]. The other is the segment feature produced in the following way. The CFM layer is applied on the full-image convolutional feature map. This gives us an arbitrary-sized (in terms of its bounding box) segment feature. Then we use another SPP layer on this feature to produce a fixed-length output. The two pooled features are fed into two separate fc layers. The features of the last fc layers are concatenated to train a classifier, as is the classifier in [10].

In this design, we have two pathways of the fc layers in both training and testing.

**Design B: on the spatial pyramid pooling layer.** We first

adopt the SPP layer [11] to pool the features. We use a 4-level pyramid of $\{6 \times 6, 3 \times 3, 2 \times 2, 1 \times 1\}$ as in [11]. The $6 \times 6$ level is actually a $6 \times 6$ tiny feature map that still has plenty spatial information. We apply the CFM layer on this tiny feature map to produce the segment feature. This feature is then concatenated with the other three levels and fed onto the fc layers, as shown in Figure 3 (right).

In this design, we keep one pathway of the fc layers to reduce the computational cost and over-fitting risk.

## 2.3. Training and Inference

Based on these two designs and the CFM layer, the training and inference stages can be easily conducted following the common practices in [8, 11, 10]. In both stages, we use the region proposal algorithm (*e.g.*, selective search [22]) to generate about 2,000 region proposals and associated segments. The input image is resized to multiple scales (the shorter edge $s \in \{480, 576, 688, 864, 1200\}$) [11], and the convolutional feature maps are extracted from full images and then fixed (not further tuned).

**Training.** We first apply the SPP method [11][1] to finetune a network for object detection. Then we replace the fine-tuned network with the architecture as in Design A or B, and further finetune the network for segmentation. In the second fine-tuning step, the segment proposal overlapping a ground-truth foreground segment by $[0.5, 1]$ is considered as positive, and $[0.1, 0.3]$ as negative. The overlap is measured by intersection-over-union (IoU) score based on the two segments' areas (rather than their bounding boxes). After fine-tuning, we train a linear SVM classifier on the network output, for each category. In the SVM training, only the ground-truth segments are used as positive samples.

**Inference.** Each region proposal is assigned to a proper scale as in [11]. The features of each region and its associated segment are extracted as in Design A or B. The SVM classifier is used to score each region.

Given all the scored region proposals, we obtain the pixel-level category labeling by the pasting scheme in SDS [10]. This pasting scheme sequentially selects the region proposal with the highest score, performs region refinement, inhibits overlapping proposals, and pastes the pixel labels onto the labeling result. Region refinement improves the accuracy by about 1% on PASCAL VOC 2012 for both SDS and our method.

## 2.4. Results on Object Segmentation

We evaluate our method on the PASCAL VOC 2012 semantic segmentation benchmark [7] that has 20 object categories. We follow the "comp6" evaluation protocol, which is also used in [4, 8, 10]. The training set of PASCAL VOC

---

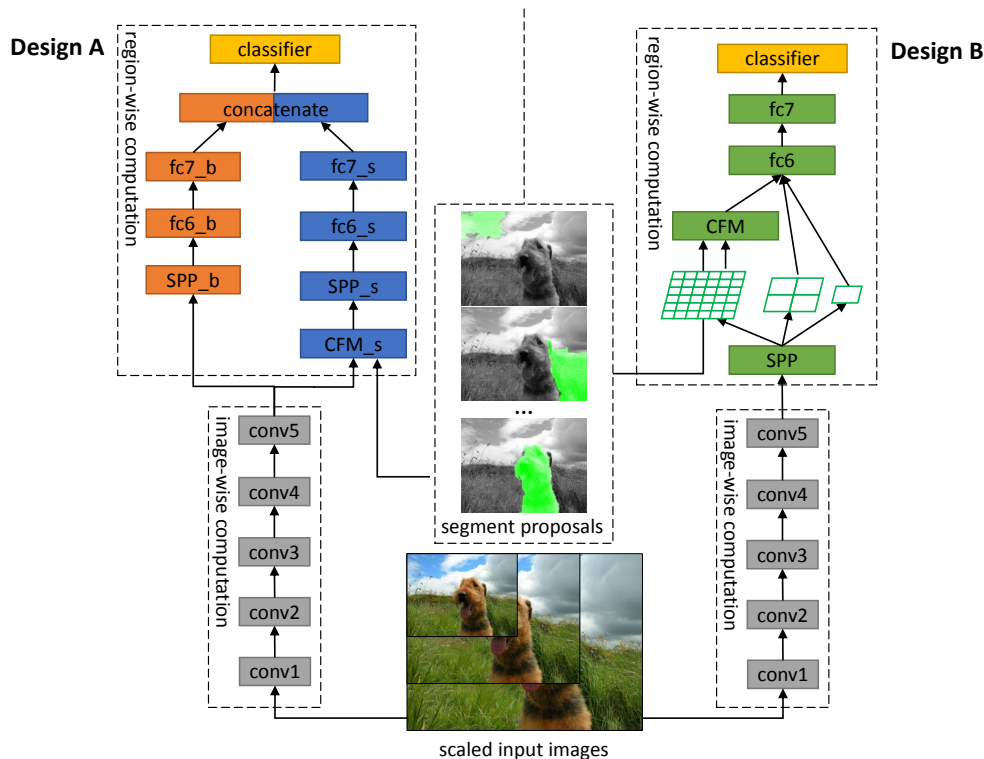[1] https://github.com/ShaoqingRen/SPP_net

Figure 3: Two network designs in this paper. The input image is processed as a whole at the convolutional layers from conv1 to conv5. Segments are exploited at a deeper hierarchy by: (Left) applying CFM on the feature map of conv5, where "_b" means for "bounding boxes" and "_s" means for segments; (Right) applying CFM on the finest feature map of the spatial pyramid pooling layer.

2012 and the additional segmentation annotations from [9] are used for training and evaluation as in [4, 8, 10]. Two scenarios are studied: semantic segmentation and simultaneous detection and segmentation.

### Scenario I: Semantic Segmentation

In the experiments of semantic segmentation, category labels are assigned to all the pixels in the image, and the accuracy is measured by region IoU scores [7].

We first study using the "ZF SPPnet" model [11] as our feature extractor. This model is based on Zeiler and Fergus's fast model [25] but with the SPP layer [11]. It has five convolutional layers and three fc layers. This model is released with the code of [11]. We note that the results in R-CNN [8] and SDS [10] use the "AlexNet" [13] instead. To understand the impacts of the pre-trained models, we report their *object detection* mAP on the *val* set of PASCAL VOC 2012: SPP-Net (ZF) is 51.3%, R-CNN (AlexNet) is 51.0%, and SDS (AlexNet) is 51.9%. This means that both pre-trained models are comparable as generic feature extractors. So the following gains of CFM are not simply due to pre-trained models.

To show the effect of the CFM layer, we present a baseline with no CFM - in our Design B, we remove the CFM layer but still use the same entire pipeline. We term this baseline as the "no-CFM" version of our method. Actually, this baseline degrades to the original SPP-net usage [11], except that the definitions of positive/negative samples are for segmentation. Table 1 compares the results of no-CFM and the two designs of CFM. We find that the CFM has obvious advantages over the no-CFM baseline. This is as expected, because the no-CFM baseline has not any segment-based feature. Further, we find that the designs A and B perform just comparably, while A needs to compute two pathways of the fc layers. So in the rest of this paper, we adopt Design B for ZF SPPnet.

In Table 2 we evaluate our method using different region proposal algorithms. We adopt two proposal algorithms: Selective Search (SS) [22], and Multiscale Combinatorial Grouping (MCG) [1]. Following the protocol in [10], the "fast" mode is used for SS, and the "accurate" mode is used for MCG. Table 2 shows that our method achieves higher accuracy on the MCG proposals. This indicates that our feature masking method can exploit the information generated by more accurate segmentation proposals.

| no-CFM | CFM (A) | CFM (B) |
|--------|---------|---------|
| 43.4 | 51.0 | 50.9 |

Table 1: Mean IoU on PASCAL VOC 2012 *validation* set using our various designs. Here we use ZF SPPnet and Selective Search.

| | ZF SPPnet | VGG net |
|-----|-----------|---------|
| SS | 50.9 | 56.3 |
| MCG | 53.0 | 60.9 |

Table 2: Mean IoU on PASCAL VOC 2012 *validation* set using different pre-trained networks and proposal methods. SS denotes Selective Search [22], and MCG denotes Multiscale Combinatorial Grouping [1].

| | ZF SPPnet | VGG net |
|---------|-----------|---------|
| 5-scale | 53.0 | 60.9 |
| 1-scale | 52.9 | 60.5 |

Table 3: Mean IoU on PASCAL VOC 2012 *validation* set using different scales. Here we use MCG for proposals.

| | conv time | fc time | total time |
|------------------|-----------|---------|------------|
| SDS (AlexNet) [10] | 17.8s | 0.14s | 17.9s |
| CFM, (ZF, 5 scales) | 0.29s | 0.09s | 0.38s |
| CFM, (ZF, 1 scale) | 0.04s | 0.09s | 0.12s |
| CFM, (VGG, 5 scales) | 1.74s | 0.36s | 2.10s |
| CFM, (VGG, 1 scale) | 0.21s | 0.36s | 0.57s |

Table 4: Feature extraction time per image on GPU.

In Table 2 we also evaluate the impact of pre-trained networks. We compare the ZF SPPnet with the public VGG-16 model [20][2]. Recent advances in image classification have shown that very deep networks [20] can significantly improve the classification accuracy. The VGG-16 model has 13 convolutional and 3 fc layers. Because this model has no SPP layer, we consider its last pooling layer ($7 \times 7$) as a special SPP layer which has a single-level pyramid of $\{7 \times 7\}$. In this case, our Design B does not apply because there is no coarser level. So we apply our Design A instead. Table 2 shows that our results improve substantially when using the VGG net. This indicates that our method benefits from the more representative features learned by deeper models.

In Table 3 we evaluate the impact of image scales. Instead of using the 5 scales, we simply extract features from single-scale images whose shorter side is $s = 576$. Table 3 shows that our single-scale variant has negligible degradation. But the single-scale variant has a faster computational speed as in Table 4.

Next we compare with the state-of-the-art results on the PASCAL VOC 2012 test set in Table 5. Here SDS [10] is the previous state-of-the-art method on this task, and $O_2P$ [4] is a leading non-CNN-based method. Our method with ZF SPPnet and MCG achieves a score of 55.4. This is 3.8% higher than the SDS result reported in [10] which uses AlexNet and MCG. This demonstrates that our CFM method can produce effective features without masking raw-pixel images. With the VGG net, our method has a score of **61.8** on the test set.

Besides the high accuracy, our method is much faster than SDS. The running time of the feature extraction steps in SDS and our method is shown in Table 4. Both approaches are run on an Nvidia GTX Titan GPU based on the Caffe library [12]. The time is averaged over 100 random images from PASCAL VOC. Using 5 scales, our method with ZF SPPnet is $\sim 47\times$ faster than SDS; using 1 scale, our method with ZF SPPnet is $\sim 150\times$ faster than SDS and is more accurate. The speed gain is because our method only needs to compute the feature maps once. Table 4 also shows that our method is still feasible using the VGG net.

Concurrent with our work, a Fully Convolutional Network (FCN) method [16] is proposed for semantic segmentation. It has a score (62.2 on test set) comparable with our method, and has a fast speed as it also performs convolutions once on the entire image. But FCN is not able to generate instance-wise results, which is another metric evaluated in [10]. Our method is also applicable in this case, as evaluated below.

**Scenario II: Simultaneous Detection and Segmentation**

In the evaluation protocol of simultaneous detection and segmentation [10], all the object instances and their segmentation masks are labeled. In contrast to semantic segmentation, this scenario further requires to identify different object instances in addition to labeling pixel-wise semantic categories. The accuracy is measured by mean AP[r] score defined in [10].

We report the mean AP[r] results on VOC 2012 validation set following [10], as the ground-truth labels for the test set are not available. As shown in Table 6, our method has a mean AP[r] of 53.2 when using ZF SPPnet and MCG. This is better than the SDS result (49.7) reported in [10]. With the VGG net, our mean AP[r] is **60.7**, which is the state-of-the-art result reported in this task. Note that the FCN method [16] is not applicable when evaluating the mean AP[r] metric,

| | mean | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow | table | dog | horse | mbike | person | plant | sheep | sofa | train | tv |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| O$_2$P [4] | 47.8 | 64.0 | **27.3** | 54.1 | 39.2 | 48.7 | 56.6 | 57.7 | 52.5 | 14.2 | 54.8 | 29.6 | 42.2 | 58.0 | 54.8 | 50.2 | 36.6 | 58.6 | 31.6 | 48.4 | 38.6 |
| SDS (AlexNet + MCG) [10] | 51.6 | 63.3 | 25.7 | 63.0 | 39.8 | 59.2 | 70.9 | 61.4 | 54.9 | 16.8 | 45.0 | 48.2 | 50.5 | 51.0 | 57.7 | 63.3 | 31.8 | 58.7 | 31.2 | 55.7 | 48.5 |
| CFM (ZF + SS) | 53.5 | 63.3 | 21.5 | 59.1 | 40.3 | 52.4 | 68.6 | 55.4 | 66.6 | 25.4 | 60.5 | 48.5 | 60.0 | 53.6 | 58.6 | 59.8 | 40.5 | **68.6** | 31.7 | 49.3 | 53.6 |
| CFM (ZF + MCG) | 55.4 | 65.2 | 23.5 | 59.0 | 40.4 | 61.1 | 68.9 | 57.9 | 70.8 | 23.9 | 59.4 | 44.7 | 66.2 | 57.5 | 62.1 | 57.6 | 44.1 | 64.5 | 42.5 | 52.9 | **55.7** |
| CFM (VGG + MCG) | **61.8** | **75.7** | 26.7 | **69.5** | **48.8** | **65.6** | **81.0** | **69.2** | **73.3** | **30.0** | **68.7** | **51.5** | **69.1** | **68.1** | **71.7** | **67.5** | **50.4** | 66.5 | **44.4** | **58.9** | 53.5 |

Table 5: Mean IoU scores on the PASCAL VOC 2012 *test* set.

| method | mean AP$^r$ |
|---|---|
| SDS (AlexNet + MCG) [10] | 49.7 |
| CFM (ZF + SS) | 51.0 |
| CFM (ZF + MCG) | 53.2 |
| CFM (VGG + MCG) | **60.7** |

Table 6: Instance-wise semantic segmentation evaluated by mean AP$^r$ [10] on PASCAL VOC 2012 *validation* set.

because it cannot produce object instances.

## 3. Joint Object and Stuff Segmentation

The semantic categories in natural images can be roughly divided into *objects* and *stuff*. Objects have consistent shapes and each instance is countable, while stuff has consistent colors or textures and exhibits as arbitrary shapes, *e.g.*, grass, sky, and water. So unlike an object, a stuff region is not appropriate to be represented as a rectangular region or a bounding box. While our method can generate segment features, each segment is still associated with a bounding box due to its way of generation. When the region/segment proposals are provided, it is rarely that the stuff can be fully covered by a single segment. Even if the stuff is covered by a single rectangular region, it is almost certain that there are many pixels in this region that do not belong to the stuff. So stuff segmentation has issues different from object segmentation.

Next we show a generalization of our framework to address this issue involving stuff. We can simultaneously handle objects and stuff by a single solution. Especially, the convolutional feature maps need only to be computed once. So there will be little extra cost if the algorithm is required to further handle stuff.

Our generalization is to modify the underlying probabilistic distributions of the samples during training. Instead of treating the samples equally, our training will bias toward the proposals that can cover the stuff as compact as possible (discussed below). A Segment Pursuit procedure is proposed to find the compact proposals.

### 3.1. Stuff Representation by Segment Combination

We treat stuff as a combination of multiple segment proposals. We expect that each segment proposal can cover a stuff portion as much as possible, and a stuff can be fully covered by several segment proposals. At the same time, we hope the combination of these segment proposals is compact - the fewer the segments, the better.

We first define a candidate set of segment proposals (in a single image) for stuff segmentation. We define a "purity score" as the IoU ratio between a segment proposal and the stuff portion that is within the bounding box of this segment. Among all the segment proposals in a single image, those having high purity scores ($> 0.6$) with stuff consist of the candidate set for potential combinations.

To generate one compact combination from this candidate set, we adopt a procedure similar to the matching pursuit [23, 17]. We sequentially pick segments from the candidate set without replacement. At each step, the largest segment proposal is selected. This selected proposal then inhibits its highly overlapped proposals in the candidate set (they will not be selected afterward). In this paper, the inhibition overlap threshold is set as IoU=0.2. The process is repeated till the remaining segments all have areas smaller than a threshold, which is the average of the segment areas in the initial candidate set (of that image). We call this procedure segment pursuit.

Figure 4 (b) shows an example if segment proposals are randomly sampled from the candidate set. We see that there are many small segments. It is harmful to define these small, less discriminative segments as either positive or negative samples (*e.g.*, by IoU) - if they are positive, they are just a very small part of the stuff; if they are negative, they share the same textures/colors as a larger portion of the stuff. So we prefer to ignore these samples in the training, so the classifier will not bias toward any side about these small samples. Figure 4 (c) shows the segment proposals selected by segment pursuit. We see that they can cover the stuff (grass here) by only a few but large segments. We expect the solver to rely more on such a compact combination of proposals.

However, the above process is *deterministic* and can only give a small set of samples from each image. For example, in Figure 4 (c) it only provides 5 segment proposals. In
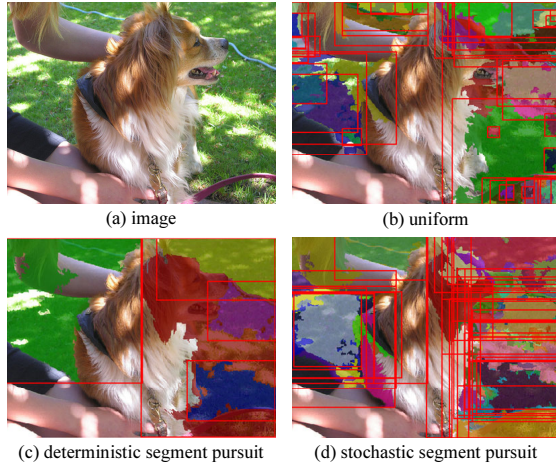
Figure 4: Stuff segment proposals sampled by different methods. (a) input image; (b) 43 regions uniformly sampled; (c) 5 regions sampled by deterministic segment pursuit; (d) 43 regions sampled by stochastic segment pursuit for finetuning.

the fine-tuning process, we need a large number of *stochastic* samples for the training. So we inject randomness into the above segment pursuit procedure. In each step, we randomly sample a segment proposal from the candidate set, rather than using the largest. The picking probability is proportional to the area size of a segment (so a larger one is still preferred). This can give us another compact combination in a stochastic way. Figure 4 (d) shows an example of the segment proposals generated in a few trials.

All the segment proposals given by this way are considered as the positive samples of a category of stuff. The negative samples are the segment proposals whose purity scores are below 0.3. These samples can then be used for fine-tuning and SVM training as detailed below.

During the fine-tuning stage, in each epoch each image generates a stochastic compact combination. All the segment proposals in this combination for all images consist of the samples of this epoch. These samples are randomly permuted and fed into the SGD solver. Although now the samples appear mutually independent to the SGD solver, they are actually sampled jointly by the rule of segment pursuit. Their underlying probabilistic distributions will impact the SGD solver. This process is repeated for each epoch. For the SGD solver, we halt the training process after 200k mini-batches. For SVM training, we only use the single combination given by the deterministic segment pursuit.

Using this way, we can treat object+stuff segmentation in the same framework as for object-only. The only difference is that the stuff samples are provided in a way given by segment pursuit, rather than purely randomly. To balance different categories, the portions of objects, stuff, and

| | Super Parsing | O$_2$P | ZF+SS | | | VGG+SS | VGG+MCG |
| | | | no-CFM | CFM w/o SP | CFM | CFM | CFM |
|---|---|---|---|---|---|---|---|
| mean | - | 18.1 | 20.7 | 24.0 | 26.6 | 31.5 | **34.4** |
| mean on † | 15.2 | 29.2 | 32.4 | 37.2 | 40.4 | 46.1 | **49.5** |
| aeroplane† | 19.5 | 36.4 | 20.5 | 37.6 | 42.9 | **48.9** | 47.5 |
| bicycle† | 11.3 | 23.5 | 32.2 | 39.1 | 40.3 | 41.2 | **48.0** |
| bird† | 4.1 | 24.6 | 27.9 | 40.5 | 46.6 | 52.9 | **59.0** |
| boat† | 0.0 | 22.3 | 16.6 | 29.9 | 34.0 | 33.6 | **37.7** |
| bottle† | 1.2 | 15.0 | 40.0 | 39.0 | 39.8 | 41.5 | **51.6** |
| bus† | 14.0 | 43.2 | 50.0 | 52.4 | 53.5 | 61.0 | **65.2** |
| car† | 15.0 | 33.5 | 41.0 | 44.0 | 47.1 | 53.7 | **57.2** |
| cat† | 20.1 | 36.7 | 45.1 | 54.1 | 56.1 | 60.0 | **67.4** |
| chair† | 2.9 | 6.8 | 13.6 | 15.7 | 17.7 | 22.9 | **24.6** |
| cow† | 0.1 | 16.2 | 28.5 | 34.8 | 39.8 | 52.4 | **58.9** |
| table† | 6.4 | 7.0 | 12.1 | 12.3 | 13.9 | 11.5 | **16.7** |
| dog† | 11.5 | 26.9 | 39.5 | 48.3 | 51.4 | 57.6 | **63.7** |
| horse† | 2.0 | 26.4 | 33.0 | 40.2 | 43.1 | **50.5** | 58.0 |
| motorbike† | 14.3 | 32.8 | 40.4 | 45.1 | 47.9 | 54.8 | 55.0 |
| person† | 30.1 | 44.5 | 47.4 | 51.0 | 54.5 | 59.9 | **65.0** |
| pottedplant† | 1.1 | 15.9 | 31.4 | 31.5 | 34.9 | 34.1 | **41.1** |
| sheep† | 4.2 | 23.7 | 29.3 | 45.5 | 56.3 | 59.6 | **60.7** |
| sofa† | 3.6 | 16.1 | 15.2 | 19.1 | 22.0 | 22.1 | **31.8** |
| train† | 10.4 | 26.7 | 33.6 | 39.1 | 43.0 | 49.0 | **56.1** |
| tvmonitor† | 9.0 | 24.3 | 40.3 | 41.0 | 40.7 | **50.4** | 50.3 |
| sky† | 65.6 | 75.6 | 64.9 | 70.3 | 76.8 | **80.6** | 76.8 |
| grass† | 45.3 | 56.0 | 51.9 | 56.9 | 60.7 | **66.1** | 66.1 |
| ground† | 24.0 | 27.6 | 22.0 | 20.5 | 22.8 | 38.3 | **39.4** |
| road† | 15.8 | 31.2 | 25.3 | 30.6 | 34.0 | 36.2 | **37.8** |
| building† | 19.8 | 24.3 | 25.0 | 28.2 | 32.4 | 37.9 | **39.5** |
| tree† | 37.8 | 44.3 | 44.2 | 50.9 | 53.4 | **59.8** | 58.0 |
| water† | 34.5 | 54.8 | 51.4 | 54.1 | 59.7 | 65.3 | **69.1** |
| mountain† | 8.8 | 19.2 | 14.9 | 20.8 | 18.4 | **36.7** | 35.6 |
| wall† | 30.8 | 40.5 | 28.7 | 36.4 | 40.4 | 42.5 | **43.8** |
| floor† | 14.4 | 25.7 | 23.0 | 28.1 | 31.7 | 35.9 | **38.9** |
| track† | 17.5 | 29.5 | 35.7 | 27.3 | 31.9 | **40.1** | 38.2 |
| keyboard† | 0.1 | 18.2 | 26.1 | 30.2 | 25.1 | 36.7 | **39.8** |
| ceiling† | 6.4 | 12.7 | 19.3 | 13.4 | 20.3 | **27.9** | 23.8 |
| bag | - | 1.2 | 0.5 | 2.1 | 2.8 | 2.1 | **9.0** |
| bed | - | 0.7 | 0.0 | 1.2 | **3.0** | 1.1 | 2.9 |
| bedclothes | - | 0.0 | 4.0 | 9.9 | 11.9 | 13.7 | **16.6** |
| bench | - | 0.1 | 0.0 | 0.0 | 0.1 | 0.0 | **0.2** |
| book | - | 5.0 | 15.0 | 8.9 | 14.8 | **20.5** | 20.1 |
| cabinet | - | 4.4 | 4.2 | 5.0 | 8.7 | 11.4 | **18.4** |
| cloth | - | 1.8 | 0.2 | 2.8 | **3.4** | 2.7 | 2.7 |
| computer | - | 0.0 | 0.0 | 3.8 | 5.0 | 4.5 | **8.6** |
| cup | - | 1.4 | 7.4 | 8.1 | 12.4 | 21.2 | **25.5** |
| curtain | - | 11.6 | 12.5 | 9.2 | 15.5 | 21.4 | **25.1** |
| door | - | 2.3 | 3.9 | 5.0 | 6.2 | **12.7** | 4.9 |
| fence | - | 6.6 | 8.4 | 4.9 | 7.3 | 20.7 | **23.3** |
| flower | - | 6.8 | 4.1 | 9.0 | 8.4 | 10.7 | **28.0** |
| food | - | 10.7 | 22.7 | 23.3 | 29.8 | 35.2 | **38.1** |
| mouse | - | 0.9 | 1.4 | 4.4 | 9.9 | **12.4** | 12.3 |
| plate | - | 5.6 | 7.5 | 7.0 | 10.7 | 19.3 | **21.8** |
| platform | - | 7.5 | 14.7 | 16.2 | 18.4 | 18.7 | **26.7** |
| rock | - | 6.7 | 8.1 | 13.5 | 15.0 | 24.2 | **26.4** |
| shelves | - | 3.7 | 1.8 | 1.5 | 3.6 | 3.1 | **10.4** |
| sidewalk | - | 0.5 | 0.0 | 1.7 | 2.5 | **7.9** | 6.8 |
| sign | - | 7.0 | 1.6 | 4.1 | 8.7 | **18.7** | 17.2 |
| snow | - | 16.4 | 19.0 | 23.9 | 28.9 | 28.3 | **40.5** |
| truck | - | 0.2 | 0.3 | 0.4 | 4.5 | 3.4 | **11.0** |
| window | - | 14.6 | 10.7 | 11.9 | 12.4 | **21.7** | 19.2 |
| wood | - | 0.8 | 0.5 | 2.4 | 2.8 | 1.9 | **5.8** |
| light | - | 8.5 | 11.5 | 6.2 | 10.9 | 15.3 | **20.5** |

Table 7: Segmentation accuracy measured by IoU scores on the new PASCAL-CONTEXT validation set [18]. The categories marked by † are the 33 easier categories identified in [18]. The results of SuperParsing [21] and O$_2$P [4] are from the *errata* of [18].

background samples in each mini-batch are set to be approximately 30%, 30%, and 40%. The testing stage is the same as in the object-only case. While the testing stage is unchanged, the classifiers learned are biased toward those compact proposals.
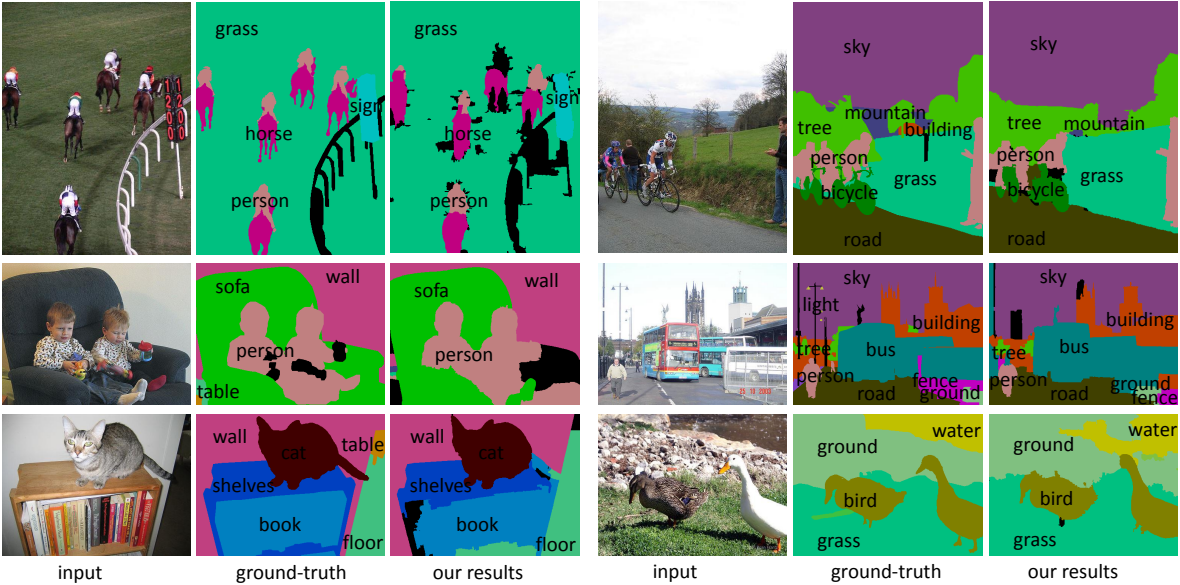
Figure 5: Some example results of our CFM method (with VGG and MCG) for **joint object and stuff segmentation**. The images are from the PASCAL-CONTEXT validation set [18].

## 3.2. Results on Joint Object and Stuff Segmentation

We conduct experiments on the newly labeled PASCAL-CONTEXT dataset [18] for joint object and stuff segmentation. In this enriched dataset, every pixel is labeled with a semantic category. It is a challenging dataset with various images, diverse semantic categories, and balanced ratios of object/stuff pixels. Following the protocol in [18], the semantic segmentation is performed on the most frequent 59 categories and one background category (Table 7). The segmentation accuracy is measured by mean IoU scores over the 60 categories. Following [18], the mean of the scores over a subset of 33 easier categories (identified by [18]) is reported in this 60-way segmentation task as well. The training and evaluation are performed on the train and val sets respectively. We compare with two leading methods - SuperParsing [21] and $O_2P$ [4], whose results are reported in [18]. For fair comparisons, the region refinement [10] is not used in all methods. The pasting scheme is the same as in $O_2P$ [4]. In this comparison, we ignore R-CNN [8] and SDS [10] because they have not been developed for stuff.

Table 7 shows the mean IoU scores. Here "no-CFM" is our baseline (no CFM, no segment pursuit); "CFM w/o SP" is our CFM method but without segment pursuit; and "CFM" is our CFM method with segment pursuit. When segment pursuit is not used, the positive stuff samples are uniformly sampled from the candidate set (in which the segments have purity scores > 0.6).

SuperParsing [21] gets a mean score of 15.2 on the easier 33 categories, and the overall score is unavailable in [18]. The $O_2P$ method [4] results in 29.2 on the easier 33 cate-

gories and 18.1 overall, as reported in [18]. Both methods are not based on CNN features.

For the CNN-based results, the no-CFM baseline (20.7, with ZF and SS) is already better than $O_2P$ (18.1). This is mainly due to the generic features learned by deep networks. Our CFM method without segment pursuit improves the overall score to 24.0. This shows the effects of the masked convolutional features. With our segment pursuit, the CFM method further improves the overall score to 26.6. This justifies the impact of the samples generated by segment pursuit. When replacing the ZF SPPnet by the VGG net, and the SS proposals by MCG, our method yields an over score of **34.4**. So our method benefits from deeper models and more accurate segment proposals. Some of our results are shown in Figure 5.

It is worth noticing that although only mean IoU scores are evaluated in this dataset, our method is also able to generate instance-wise results for objects.

## 4. Conclusion

We have presented convolutional feature masking, which exploits the shape information at a late stage in the network. We have further shown that convolutional feature masking is applicable for joint object and stuff segmentation.

We plan to further study improving object detection by convolutional feature masking. Exploiting the context information provided by joint object and stuff segmentation would also be interesting.

# References

[1] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. CVPR, 2014.

[2] T. Brox, L. Bourdev, S. Maji, and J. Malik. Object segmentation by alignment of poselet activations to image contours. In *CVPR*, 2011.

[3] V. Bychkovsky, S. Paris, E. Chan, and F. Durand. Learning photographic global tonal adjustment with a database of input / output image pairs. In *CVPR*, 2011.

[4] J. Carreira, R. Caseiro, J. Batista, and C. Sminchisescu. Semantic segmentation with second-order pooling. In *ECCV*. 2012.

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

[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. *arXiv:1310.1531*, 2013.

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

[8] 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.

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

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

[11] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *arXiv preprint arXiv:1406.4729*, 2014.

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012.

[14] M. P. Kumar, P. Ton, and A. Zisserman. Obj cut. In *CVPR*, 2005.

[15] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1989.

[16] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *arXiv:1411.4038*, 2014.

[17] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 1993.

[18] 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*. 2014.

[19] J. Shotton, J. Winn, C. Rother, and A. Criminisi. Textonboost: Joint appearance, shape and context modeling for mulit-class object recognition and segmentation. In *ECCV*, 2006.

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

[21] J. Tighe and S. Lazebnik. Superparsing: scalable nonparametric image parsing with superpixels. In *ECCV*. 2010.

[22] J. R. Uijlings, K. E. van de Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *IJCV*, 2013.

[23] Y. N. Wu, Z. Si, H. Gong, and S.-C. Zhu. Learning active basis model for object detection and recognition. *IJCV*, 2010.

[24] Y. Yang, S. Hallman, D. Ramanan, and C. Fowlkes. Layered object detection for multi-class segmentation. In *CVPR*, 2010.

[25] M. D. Zeiler and R. Fergus. Visualizing and understanding convolutional neural networks. *arXiv preprint arXiv:1311.2901*, 2013.