Webly Supervised Semantic Segmentation

Bin Jin
IC, EPFL
bin.jin@epfl.ch

Maria V. Ortiz Segovia
Océ Print Logic Technologies
Maria.Ortiz@oce.com

Sabine Süsstrunk
IC, EPFL
sabine.susstrunk@epfl.ch

Abstract

We propose a weakly supervised semantic segmentation algorithm that uses image tags for supervision. We apply the tags in queries to collect three sets of web images, which encode the clean foregrounds, the common backgrounds, and realistic scenes of the classes. We introduce a novel three-stage training pipeline to progressively learn semantic segmentation models. We first train and refine a class-specific shallow neural network to obtain segmentation masks for each class. The shallow neural networks of all classes are then assembled into one deep convolutional neural network for end-to-end training and testing. Experiments show that our method notably outperforms previous state-of-the-art weakly supervised semantic segmentation approaches on the PASCAL VOC 2012 segmentation benchmark. We further apply the class-specific shallow neural networks to object segmentation and obtain excellent results.

1. Introduction

Semantic segmentation, which refers to accurately assigning semantic labels to the corresponding pixels in an image, is a challenging task actively studied in computer vision. Recent breakthroughs [1, 2, 3, 4, 5, 6] in semantic segmentation are mainly due to the *fully supervised* algorithms that apply Convolutional Neural Networks (CNNs) on datasets that contain images and their pixel-wise annotations, *e.g.*, PASCAL VOC [7] and Microsoft COCO [8]. These algorithms report excellent performance on the limited amount of classes covered by these datasets. The PASCAL VOC segmentation set contains 20 object classes with 500 images per class, the Microsoft COCO 91 object classes with 3.5K images per class, respectively. Extending fully supervised algorithms to more object classes, however, requires collecting massive amount of pixel-wise annotations, which is both time-consuming and expensive. As reported in [9], the average annotation time was 239.7 seconds per image in the PASCAL VOC 2012 dataset. Thus, other annotations that are less precise but faster to collect, such as points, scribbles, or bounding boxes, have also been employed to supervise semantic segmentation [9, 10, 11, 12].

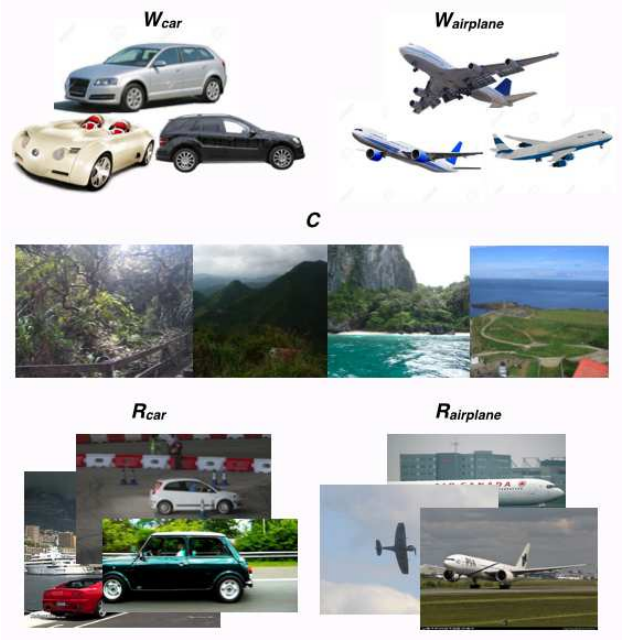


Figure 1: To supervise semantic segmentation, we extract three sets of web images: images with white background $\{\mathbf{W_k}\}$, images that contain common background scenes $\mathbf{C}$, and realistic images $\{\mathbf{R_k}\}$. $\{\mathbf{W_k}\}$ and $\mathbf{C}$ sets are used for initial training of the segmentation models. Later, these models are iteratively refined on the realistic image set $\{\mathbf{R_k}\}$. Finally $\{\mathbf{R_k}\}$ is adopted again to train an end-to-end semantic segmentation network.

Our proposed semantic segmentation algorithm only uses *image tags* as supervision. Image tags represent which object class(es) are present in the image. They are usually much easier and faster to obtain than the other human annotations described above, and have thus been used in many *weakly supervised* semantic segmentation methods [13, 14, 15, 16, 17, 18, 19, 20, 21, 22]. However, as opposed to the segmentation masks obtained by the pixel-wise annotations, image tags do not indicate the location of the object(s) in the image, therefore making semantic segmen-

tation much more challenging.

In our framework, we take advantage of the enormous amount of images and their rich context on the Internet. Through cleverly querying and exploiting web images, we can build a pipeline to automatically generate segmentation masks for each class and then train a deep convolutional neural network for segmenting multiple classes using the generated masks. We only use image tags to query the web images and to train the network. No additional human annotations or interactions are required.

We extract three sets of images to supervise semantic segmentation, as shown in Figure 1. These three sets of images include the objects on white background, the common backgrounds, and the realistic scenes of the classes. We propose a novel three-stage procedure to progressively train our semantic segmentation model, as illustrated in Figure 2. In the first stage, a Shallow Neural Network (SNN) is initially trained to predict class-specific segmentation masks, using the hypercolumn features [23] from images with white background and images with common backgrounds. We then iteratively refine the SNN of each class on a set of realistic images of that class to generate better segmentation masks. In the last stage, the SNNs of all classes are assembled into one Deep Convolutional Neural Network (DCNN) by training the DCNN with the predicted multi-label segmentation masks from the SNNs. The DCNN, after the last training stage, outperforms current state-of-the-art weakly supervised semantic segmentation algorithms on the PASCAL VOC 2012 segmentation benchmark [7] by a notable margin.

In summary, our **main contributions** are:

- We propose to collect three sets of useful web images for supervising segmentation. The first set contains images with white background, the second set contains images with common background scenes, and the third set contains realistic images of each class.
- We present a novel three-stage pipeline to train semantic segmentation models using the three collected web image sets. The segmentation performance progressively improves following the training pipeline.
- Our DCNN, after the three training stages, achieves state-of-the-art performance on the PASCAL VOC 2012 segmentation benchmark, outperforming the previous weakly supervised semantic segmentation algorithms by more than 3 percent.
- The SNNs from the first two training stages produce state-of-the-art results in an object segmentation application.

2. Related Work

Weakly Supervised Semantic Segmentation Our method belongs to the family of *weakly supervised semantic segmentation* algorithms that require only image tag annotation. Here we review the CNN-based approaches as these methods [13, 14, 15, 16, 17, 18, 19, 20, 21, 22] provide good segmentation quality on the challenging PASCAL VOC benchmark. Early works [13, 14, 16] extend the Multiple-Instance Learning (MIL) [24] framework for weakly supervised semantic segmentation, where the loss functions are built on the image tags level. They adopt different approaches, *e.g.*, maximum pool [13] or Log-Sum-Exp [14], to pool pixel-level probability predictions into image-level losses. No object location information is considered in these frameworks, thus resulting in coarse segmentation masks. Recent methods [18, 19, 20, 21, 22] investigate how to automatically infer the location of each class without pixel-wise annotations. [20] and [19] both build their location cues by inversing the pre-trained classification network [25] on ImageNet [26]. The difference is that [20] builds general objectness measure for all classes while [19] focuses on class-specific saliency maps. Similar location cues are used in [18] in the form of the seed loss. They further integrate the seed loss with another two losses that encode more location information. The bottom up segment proposals [27] are used as another approach to obtain location information of each class in [21, 22]. Instead of inferring rough location cues for each class using objectness or saliency maps, we train a shallow segmentation network from web images to automatically generate segmentation masks for each class. Using these segmentation masks, we further train a DCNN that achieves the state-of-the-art results for semantic segmentation.

Webly Supervised Computer Vision The idea of utilizing web images for supervising computer vision algorithms has been explored in several tasks, such as object classification [28], object detection [29], object parts localization [30] and object segmentation [17,31,32,33]. Recently, Wei *et al.* [17] also propose to use web images to train CNNs for semantic segmentation. While we extract two sets of images that distinctly separate foreground and background, [17] employs Flickr images that may have cluttered backgrounds. In addition, we apply an iterative refinement step that significantly boosts the performance. Our method achieves better accuracy (4.2% mIoU better on the PASCAL VOC 2012 test set) than their approach while using significantly less web images for supervision.

Fully Supervised Semantic Segmentation For completeness sake, we also quickly review some fully supervised semantic segmentation methods [1, 2, 3, 4, 5, 6] that rely on pixel-wise annotations for training. No surprise, they report excellent performance on the standard segmentation benchmark. These methods build upon the Fully Convolutional Network (FCN) architecture [1] to perform end-to-end training. Techniques like Conditional Random

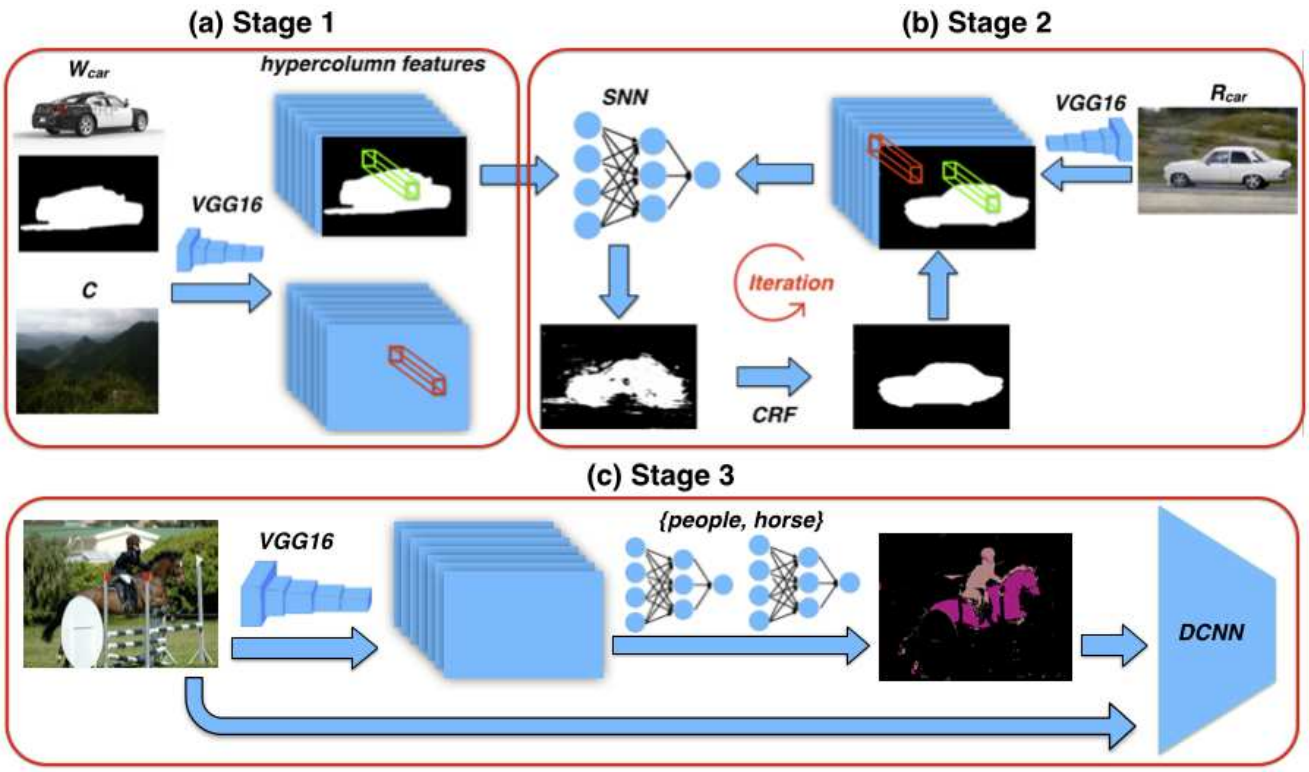


Figure 2: The proposed three-stage training pipeline. (a) Stage one: initial training of the SNN using hypercolumn features from $\mathbf{W_k}$ and $\mathbf{C}$. (b) Stage two: iterative refinement of the SNN on realistic images $\mathbf{R_k}$. (c) Stage three: all SNNs are assembled into one DCNN for end-to-end training and testing.

Field (CRF) [2, 3, 4], deconvolution [5] and Boundary Neural Fields (BNF) [6] are further combined with the FCN. However, since pixel-wise annotations are costly and time-consuming to obtain, the scalability of these methods is limited.

3. Web image sets

Billions of images have been published online with rich context information. By cleverly querying, analyzing, and extracting images from this giant collection, we propose a novel pipeline that learns semantic segmentation models with only image tag supervision. In this section, we describe how we query the web image collection and what types of web images we retrieve to supervise semantic segmentation.

For a class k, three sets of web images are collected that cover the visual appearance of the objects of the class and the backgrounds. A white background set (denoted as $\mathbf{W_k}$) is built by querying the text-based image search engine, *e.g.*, Google or Microsoft Bing, with the query "<class> on white background". Images retrieved with this query mostly have salient objects in front of a clean background and are thus easy to segment. We segment these images with a dense conditional random field (CRF) [34], using saliency maps from [35] as the unary term. Sample images from the $\mathbf{W_k}$ set and the corresponding segmentation masks are shown in Figure 3. Since images from $\mathbf{W_k}$ are relatively easy to segment, the quality of the segmentation masks, which are generated by saliency and CRF algorithms without using human annotations, is thus acceptable. Experiments in Sec. 5 show that using these masks, our segmentation networks achieve reliable segmentation performance.

Figure 3: Sample images from $\mathbf{W_{car}}$ and the corresponding segmentation masks generated from saliency combined with a dense CRF.

Images in the $\mathbf{W_k}$ set encode the foreground information of class k while the backgrounds are missing. We thus collect another set of images $\mathbf{C}$ that is unlikely to contain the classes of interest but contains common background scenes,

like sky, sea, grass, *etc*. $\mathbf{C}$ can be built by retrieving images from image sharing websites, *e.g*., Flickr or Imgur[1], with common background keywords. Another approach is to use existing online datasets that mostly contain common background scenes, such as the Holiday dataset [36].

The third set $\mathbf{R_k}$ contains images of class k depicting realistic scenes. $\mathbf{R_k}$ can be constructed by crawling image sharing websites with the given class name or using existing datasets that already cover the class. Example images for these three sets are shown in Figure 1. Note that each class has a separate $\mathbf{W_k}$ and $\mathbf{R_k}$ set while sharing the same $\mathbf{C}$ set since the $\mathbf{C}$ set contains the common backgrounds for most classes.

4. Training the network

Based on the three sets of web images, we propose a novel three-stage training pipeline to learn the semantic segmentation models, as shown in Figure 2. For each class k, a Shallow Neural Network (SNN) is initially trained to output class-specific segmentation masks, using the hypercolumn features from $\mathbf{W_k}$ (images with clean foreground) and $\mathbf{C}$ (images with common background information). The SNN then gets iteratively refined on the realistic images in $\mathbf{R_k}$. In the last stage, a DCNN is trained using the multi-label segmentation masks generated by the SNNs of all the classes.

4.1. Stage 1: Initial training

We denote $\Omega = \{1, 2, ..., N\}$ as the set of class names where N is the number of classes. For each class $k \in \Omega$, a SNN, whose parameters is denoted as Θ_k, is trained using the images in $\mathbf{W_k}$ and $\mathbf{C}$. Since objects in $\mathbf{W_k}$ are surrounded by white background, the foreground pixels from $\mathbf{W_k}$, denoted as $\mathbf{W_k^f}$, thus represent the visual appearance of class k. Correspondingly, the pixels in $\mathbf{C}$ represent the visual appearance of the common backgrounds. We use hypercolumn features [23] extracted from the pre-trained VGG16 network [25] to encode the visual appearance. For each pixel x_i^k from $\mathbf{W_k^f}$ or $\mathbf{C}$, we compute its hypercolumn feature:

$$h_i^k = \mathbf{H}(x_i^k) \tag{1}$$

Here $\mathbf{H}$ represents the operation to compute hypercolumn features. We then use the hypercolumn features from $\mathbf{C}$ and $\mathbf{W_k^f}$ to train the SNN, as shown in Figure 2(a).

Θ_k is trained to minimize the binary crossentropy loss:

$$\min_{\theta_k} \sum_i -\Big(t_i^k log\big(f(h_i^k, \theta_k)\big) + (1-t_i^k) log\big(1-f(h_i^k, \theta_k)\big)\Big) \tag{2}$$

Here $f(h_i^k, \theta_k)$ represents the SNN output. The SNN takes in the hypercolumn feature h_i^k and outputs the probability of x_i^k belongs to class k.

$$t_i^k = \begin{cases} 0, & \text{if } x_i^k \text{ is from } \mathbf{C} \\ 1, & \text{if } x_i^k \text{ is from } \mathbf{W_k^f} \end{cases} \tag{3}$$

An equal number of hypercolumn features are randomly extracted from $\mathbf{C}$ and $\mathbf{W_k^f}$, forming a balanced set for training the SNN. After initial training, the SNN of class k can predict the probability of a pixel belonging to class k, effectively outputting a class-specific mask for an image.

The combination of hypercolumn features and the SNN is effectively similar to the functionality of the Fully Convolutional Network (FCN) [1]. We separate these two steps to easily balance the class distribution and to parallelize the training for different classes. Moreover, since the network to extract hypercolumn features is pre-trained and shared, we only need to store Θ_k for a new class. Θ_k is relatively small for a shallow network (around 3.6 MB per class in our experiment), enabling efficient storage and retrieval for a large number of classes.

4.2. Stage 2: Refinement

Θ_k is initially trained to separate the objects of class k from the common backgrounds. However, the realistic background of class k may be different from the common backgrounds. We thus further iteratively refine Θ_k on realistic images, as shown in Figure 2(b). We make use of multiple CRF iterations to improve our SNNs. CRF has been shown to be helpful for semantic segmentation as it recovers missing parts and refines the boundaries in the segmentation masks [2,18,20]. Unlike most methods that apply one time CRF as post-processing, we apply CRF in each refinement iteration and learn to update the SNNs according to the CRF-refined masks. Consequently, the SNNs are forced to gradually learn to generate more complete segmentation masks with better boundaries.

Assume x_i^k represents the ith pixel in the collection of pixels from the realistic image set $\mathbf{R_k}$. h_i^k and y_i^k are its corresponding hypercolumn feature and label (0 for background and 1 for foreground). We refine Θ_k by minimizing the loss:

$$\min_{\theta_k} \sum_i -\Big(y_i^k log\big(f(h_i^k, \theta_k)\big) + (1-y_i^k) log\big(1-f(h_i^k, \theta_k)\big)\Big) \tag{4}$$

Given the parameters Θ_k of the SNN, the labels $\{y_i^k\}$ are predicted by minimizing the dense CRF energy function:

$$\min_{\{y_i^k\}} \sum_i \phi_i^k(y_i^k) + \sum_{i,j} \phi_{i,j}^k(y_i^k, y_j^k) \tag{5}$$

where the SNN's output is adopted for the unary term:

$$\phi_i^k(y_i^k) = -log\Big(y_i^k f(h_i^k, \Theta_k) + (1-y_i^k)\big(1-f(h_i^k, \Theta_k)\big)\Big) \tag{6}$$

[1] www.flickr.com and www.imgur.com

The pairwise term is set to be the standard color and spatial distance as in [34]. An *Expectation-Maximization* (EM) algorithm is adopted to iteratively optimize Equation 4 and Equation 5. In each iteration, we first update the labels of all pixels $\{y_i^k\}$ by minimizing Equation 5 using the method in [34]. Then the parameters Θ_k get updated by minimizing Equation 4 using back propagation. The new parameters Θ_k are used in Equation 5 to obtain the new $\{y_i^k\}$ as the beginning of a new iteration. Figure 4 shows the evolving of the labels $\{y_i^k\}$ through iterations. One can clearly see that the segmentation masks predicted by the SNNs progressively improve over iterations, recovering missing parts and producing better boundaries. We found that 2 refinement iterations already significantly boost the performance while still being efficient in training time. More iterations result in minor performance gain but longer training time.

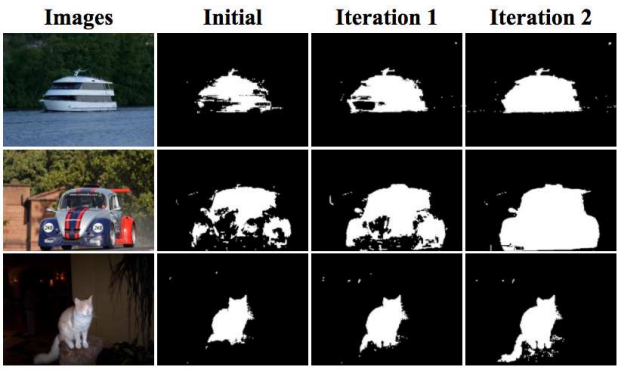


Figure 4: Improved segmentation masks by evolving the labels $\{y_i^k\}$ through iterations.

4.3. Stage 3: Assembling of classes

The SNNs are trained independently for each class. To perform semantic segmentation for multiple classes, in this stage we assemble all SNNs into one deep convolution neural network (DCNN), as shown in Figure 2(c).

We use the DCNN architecture proposed in [2] due to its outstanding performance and good documentation. The DCNN is a fully convolutional neural network that takes in an image and directly predicts a multi-label segmentation mask. We train the DCNN on all realistic images in $\{\mathbf{R_k}\}$. Since no pixel-wise human annotations are provided for training, we use the SNNs to automatically generate multi-label segmentation masks as supervision. Specifically, if one image in $\{\mathbf{R_k}\}$ is labeled with tags $C \subseteq \Omega$, the predicted label y_i for the ith pixel is:

$$y_i = \underset{k \in \{0\} \cup C}{\operatorname{argmax}} f(h_i^k, \Theta_k) \tag{7}$$

$$f(h_i^0, \Theta_0) = 1 - \max_{k \in C} f(h_i^k, \Theta_k) \tag{8}$$

Effectively, the combined multi-label segmentation mask is produced by taking the pixel-wise maximum of the probability maps across all labels, including background (represented as label 0). The background probability is set as one minus the maximum foreground probability.

After generating the multi-label segmentation masks using the SNNs, we treat them as the groundtruth segmentation masks and perform end-to-end training of the DCNN. These multi-label segmentation masks are not human annotations but automatically generated masks from our SNNs, which only require image tags during training. Therefore, the fully supervised DCNN training in [2] is transformed to weakly supervised in our framework.

5. Experiments

5.1. Setup

Datasets We validate our algorithm on the standard PASCAL VOC 2012 segmentation benchmark [7]. Following [13, 14, 16, 18, 19, 20, 21, 22], we augment it with the extra annotations from [37], resulting in an augmented training (trainaug) set of 10,582 images, a validation set of 1,449 images, and a test set of 1,456 images, covering 20 classes. We report the standard *Intersection over Union* (IoU) value on both validation and test set.

For the three-stage training, we build $\{\mathbf{W_k}\}$ by querying Google with the 20 classes (using the strategy explained in Section 3) and obtain on average 340 images per class (6807 images in total). The Holiday dataset [36], which contains 1491 holiday images, serves as the $\mathbf{C}$ set since these images cover some common background scenes, *e.g.*, sky, mountains, grass. We use the trainaug set of PASCAL VOC 2012 as $\{\mathbf{R_k}\}$ since they are all realistic images of the 20 classes. The pixel-wise annotations of the trainaug set are not used. During the first two stages of training, all images are resized such that the larger dimension equals 340. In stage three, images are used in their original size according to [2].

Training and Testing The hypercolumn features during training are extracted from the *conv1_2* (64 channels), *conv2_2* (128 channels), *conv3_3* (256 channels), *conv4_3* (512 channels) and *conv5_3* (512 channels) layers of the pre-trained VGG16 model, resulting in a 1472 dimensional vector. In the first training stage, each image of $\mathbf{W_k}$ contributes 1000 randomly selected hypercolumn features. An equal number of hypercolumn features are randomly selected from $\mathbf{C}$, forming a balanced set for initial training. In the refinement stage, 1000 hypercolumn features are randomly selected from both the foreground and the background regions of each image in $\mathbf{R_k}$.

The SNN is set as a fully connected network with 4 layers ($1472 \rightarrow 512 \rightarrow 256 \rightarrow 64 \rightarrow 1$).We set *Relu* activation in between hidden layers and *Sigmoid* activation after the last layer. The network is trained with the Adam optimizer ($\text{lr} = 0.0002$) for 50 epochs in the initial training stage and

20 epochs in each refinement iteration. The CRF parameters are set through cross-validation on 100 separate validation images [2]. The segmentation performance is robust to the CRF parameters. All the experiments are conducted on a NVIDIA TITAN X GPU with 12GB momory. Training each SNN takes approximately 10 Min for initial training and 20 Min per refinement iteration. This part is implemented in the Theano framework [38]. The third training stage is performed using Deeplab code [2], which is based on the Caffe framework [39]. The parameters of Deeplab training are set according to [2].

5.2. Performance of each stage

Since we use **Web** images to supervise semantic **S**egmentation, we call our method **WebS**. Table 1 illustrates the performance of our method at different training stages on the validation set of the PASCAL VOC 2012 dataset.

Table 1: Performance on the PASCAL VOC 2012 validation set of our method at different stages.

Method	mIoU
WebS-i	46.4
WebS-i1	51.6
WebS-i2	53.4

To properly evaluate the semantic segmentation performance for all classes, the third stage of assembling all SNNs into a DCNN is always applied. **WebS-i** represents the model that directly combines stage one with stage three, bypassing the refinement stage. **WebS-i1** and **WebS-i2** are the models trained through all three stages, with one refinement iteration and two refinement iterations, respectively. The **WebS-i** model's mIoU of 46.4% is already comparable with several state-of-the-art weakly supervised semantic segmentation methods [11, 13, 14, 16, 19], see Table 4. This demonstrates the effectiveness of our collected web images ($\{\mathbf{W_k}\}$ and $\mathbf{C}$) for semantic segmentation. The performance progressively improves after each refinement iteration. The first iteration increases the performance by 5.2% to 51.6%, already producing the best performance for weakly supervised semantic segmentation. The **WebS-i2** model further improves the performance by 1.8% for a mIoU value of 53.4%. This fact clearly demonstrates the benefits of adopting CRF in our refinement stage.

Qualitative results of the three models are shown in Figure 5 and supplementary material. We can observe that the **WebS-i** model produces rough locations of the objects while missing some parts. Each refinement iteration reveals more details of the objects. The **WebS-i2** model produces accurate segmentation masks with fine-grained boundaries, even for some non-trivial objects like the partially occluded dog in the 3rd row and the person riding on the horse in the 5th row. This is attributed to the fact that during each iteration, the CRF recovers missing parts of the objects based on the low-level statistics, like color, and produces better boundaries. Consequently, the SNNs learned from the CRF-refined masks also produce more accurate predictions. In the last image, our method fails to segment the train from the rail since these two objects often occur together. This is a typical failure case for most weakly-supervised methods, as also discussed in [18].

5.3. Number of images in $\mathbf{W_k}$ and $\mathbf{R_k}$

We further evaluate the impact of the number of images in $\{\mathbf{W_k}\}$ and $\{\mathbf{R_k}\}$ on the segmentation performance. In Table 2, we present the performance of the **WebS-i2** model with different number of training images. With only 2,000 images for the $\{\mathbf{W_k}\}$ and 2,000 images for the $\{\mathbf{R_k}\}$ (100 images per class), our method achieves a mIoU of 45.9%, already better than [11, 13, 16, 19], which are supervised with the whole trainaug set of the PASCAL VOC (10,582 images). It is even 9.3% better than [14], refer to Table 4, which uses an additional 700K images from ILSVRC [26]. Clearly by adding more images for supervision, the performance of our method improves. We project that our performance can be further improved by collecting more web images. Using only 6,807 Google images[2], 1,491 Holiday images and the 10,582 PASCAL VOC images, our method outperforms all state-of-the-art methods by a notable margin. Compared to [17] which also uses web images to supervise semantic segmentation, our method requires substantially less images ([17] uses 41K Flickr images as well as the trainaug set of PASCAL VOC), but still produces better results on both the validation set ($+3.6\%$ mIoU) and the test set ($+4.1\%$ mIoU).

Table 2: Performance on the PASCAL VOC 2012 validation set of the **WebS-i2** model using different numbers of training images.

#images in $\{\mathbf{W_k}\}$	#images in $\{\mathbf{R_k}\}$	mIoU
2,000	2,000	45.9
2,000	10,582	48.9
4,000	10,582	51.5
6,807	2,000	50.6
6,807	6,000	51.0
6,807	10,582	53.4

5.4. Comparison with weakly supervised methods

In this subsection, we compare our method with other CNN based weakly supervised semantic segmentation algorithms. Table 4 and Table 5 show the performance on the PASCAL VOC 2012 validation and test set, respectively. Here we only compare with methods that require no additional human annotations except image tags

[2]due to Google search limit and copyright protection.

Figure 5: Qualitative results on the PASCAL VOC 2012 validation set of our method before and after refinement iterations. Better viewed on screen.

[11,13,14,16,17,18,19,20]. For completeness, the comparison with methods that require additional human annotations can be found in the supplementary material. Our method produces excellent results on both the validation and test set of the PASCAL VOC, outperforming all state-of-the-art weakly-supervised semantic segmentation methods. On the validation set, we improve over the previous state-of-the-art SEC [18] method by 2.7% and achieve best scores on 12 out of 21 classes (including background) among all methods. Similar results are observed on the test set, where the evaluation is performed by the PASCAL VOC evaluation server. Our method achieves 3.6% higher mIoU than the state-of-the-art approaches, producing best scores on 14 out of 21 classes.

5.5. Object Segmentation using SNNs

We also investigated the performance of our class-specific SNNs in an object segmentation task, where the

Table 3: Comparison between our SNNs and other object segmentation methods on the OD dataset.

Method	Car	Horse	Airplane	mIoU
Joulin *et al.* [40]	37.2	30.2	15.4	27.6
Joulin *et al.* [41]	35.2	29.5	11.7	25.5
Rubinstein *et al.* [31]	64.4	51.7	55.8	57.3
Chen *et al.* [33]	64.9	33.4	40.3	46.2
SNN-i	67.7	52.4	53.8	58.0
SNN-i1	74.5	59.6	55.4	63.2
SNN-i2	**76.1**	**61.7**	**56.5**	**64.8**

Table 4: Performance on the PASCAL VOC 2012 **validation set** of semantic segmentation methods using only image tag supervision. * represents applying the dense CRF [34] as post-processing.

Method	bg	aero	bic	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
MIL-FCN [13]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	24.9
EM-Adapt [11]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	38.2
MIL-sppxl* [14]	77.2	37.3	18.4	25.4	28.2	31.9	41.6	48.1	50.7	12.7	45.7	14.6	50.9	44.1	39.2	37.9	28.3	44.0	19.6	37.6	35.0	36.6
CCCN* [16]	68.5	25.5	18.0	25.4	20.2	36.3	46.8	47.1	48.0	15.8	37.9	21.0	44.5	34.5	46.2	40.7	30.4	36.3	22.2	38.8	36.9	35.3
DCSM* [19]	76.7	45.1	24.6	40.8	23.0	34.8	61.0	51.9	52.4	15.5	45.9	32.7	54.9	48.6	57.4	51.8	38.2	55.4	**32.2**	42.6	39.6	44.1
BFBP* [20]	79.2	60.1	20.4	50.7	41.2	**46.3**	62.6	49.2	62.3	13.3	49.7	**38.1**	58.4	49.0	57.0	48.2	27.8	55.1	29.6	54.6	26.6	46.6
STC* [17]	**84.5**	**68.0**	19.5	60.5	42.5	44.8	68.4	64.0	64.8	14.5	52.0	22.8	58.0	55.3	57.8	**60.5**	**40.6**	56.7	23.0	**57.1**	31.2	49.8
SEC* [18]	82.4	62.9	26.4	61.6	27.6	38.1	66.6	62.7	75.2	**22.1**	53.5	28.3	65.8	57.8	**62.3**	52.5	32.5	62.6	32.1	45.4	45.3	50.7
ours:																						
WebS-i2*	84.3	65.3	**27.4**	**65.4**	**53.9**	**46.3**	**70.1**	**69.8**	**79.4**	13.8	**61.1**	17.4	**73.8**	**58.1**	57.8	56.2	35.7	**66.5**	22.0	50.1	**46.2**	**53.4**

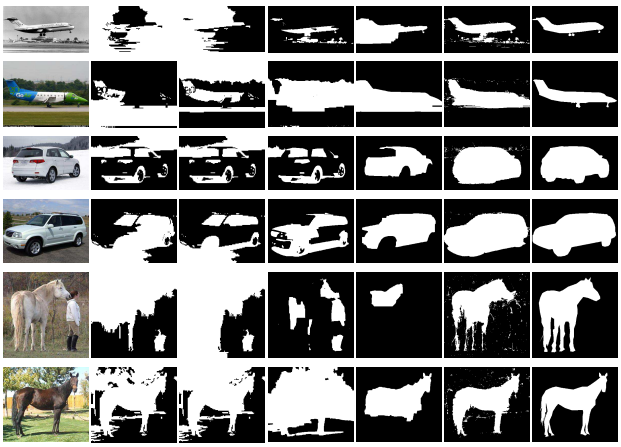
Table 5: Performance on the PASCAL VOC 2012 **test set** of semantic segmentation methods using only image tag supervision. * represents applying the dense CRF [34] as post-processing.

Method	bg	aero	bic	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mIoU
MIL-FCN [13]	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	25.7
EM-Adapt [11]	76.3	37.1	21.9	41.6	26.1	38.5	50.8	44.9	48.9	16.7	40.8	29.4	47.1	45.8	54.8	28.2	30.0	44.0	29.2	34.3	46.0	39.6
MIL-sppxl* [14]	74.7	38.8	19.8	27.5	21.7	32.8	40.0	50.1	47.1	7.2	44.8	15.8	49.4	47.3	36.6	36.4	24.3	44.5	21.0	31.5	41.3	35.8
CCCN* [16]	-	24.2	19.9	26.3	18.6	38.1	51.7	42.9	48.2	15.6	37.2	18.3	43.0	38.2	52.2	40.0	33.8	36.0	21.6	33.4	38.3	35.6
DCSM* [19]	78.1	43.8	26.3	49.8	19.5	40.3	61.6	53.9	52.7	13.7	47.3	34.8	50.3	48.9	69.0	49.7	38.4	57.1	34.0	38.0	40.0	45.1
BFBP* [20]	80.3	57.5	24.1	**66.9**	31.7	43.0	67.5	48.6	56.7	12.6	50.9	**42.6**	59.4	52.9	65.0	44.8	41.3	51.1	33.7	44.4	33.2	48.0
STC* [17]	85.2	62.7	21.1	58.0	31.4	55.0	68.8	63.9	63.7	14.2	57.6	28.3	63.0	59.8	67.6	**61.7**	42.9	61.0	23.2	**52.4**	33.1	51.2
SEC* [18]	83.5	56.4	28.5	64.1	23.6	46.5	70.6	58.5	71.3	**23.2**	54.0	28.0	68.1	62.1	70.0	55.0	38.4	58.0	**39.9**	38.4	**48.3**	51.7
ours:																						
WebS-i2*	**85.8**	**66.1**	**30.0**	64.1	**47.9**	**58.6**	**70.7**	**68.5**	**75.2**	11.3	**62.6**	19.0	**75.6**	**67.2**	**72.8**	61.4	**44.7**	**71.5**	23.1	42.3	43.6	**55.3**

user provides the labels of the classes they want to segment in an image. Note that this task is different from typical semantic segmentation where the labels of the images are not given during testing. We evaluate the performance on the Object Discovery (OD) dataset [31], a dataset containing three classes (airplane, car, horse) with 100 images per class. The SNNs are applied to generate segmentation masks using the strategy explained in Section 4.3. The labels of the images are used to retrieve the correct SNN while generating the masks, as shown in Figure 2(c).

We compare the mIoU values with state-of-the-art object segmentation methods [31, 33, 40, 41] in Table 3. **SNN-i**, **SNN-i1** and **SSN-i2** are our SNNs after the initial training, the first refinement iteration, and the second refinement iteration, respectively. Our **SNN-i2** method improves over the previous state-of-the-art method by a large margin (7.5%). Even the **SNN-i** model already achieves state-of-the-art performance, again demonstrating the effectiveness of our web image sets for supervising segmentation models.

We show sample results of our method compared with previous approaches in Figure 6 and the supplementary material. While previous approaches either miss parts of the objects or segment some background regions as the objects, our SNNs successfully segment out the whole objects with accurate boundaries.



Inputs [40] [41] [31] [33] **SNN-i2** Groundtruth

Figure 6: Sample results of our method and other state-of-the-art approaches on the OD dataset.

6. Conclusion

We propose a novel three-stage training pipeline to progressively learn the semantic segmentation model from three sets of web images. We demonstrate that our method outperforms the previous state-of-the-art weakly supervised semantic segmentation algorithms on the PASCAL VOC 2012 benchemark. The class-specific shallow neural networks (SNNs) learned in the first two training stages also produce excellent results when used in object segmentation. Note that when learning the SNNs, no pixel-wise human annotations are used. Adopting these SNNs, many fully supervised computer vision methods, such as semantic segmentation [1, 2, 3, 6] and object detection [42], can be easily transformed into a weakly supervised framework, which is in line with our future plan.

References

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3431–3440, 2015. 1, 2, 4, 8

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Semantic image segmentation with deep convolutional nets and fully connected crfs," in *International Conference on Learning Representations*, 2015. 1, 2, 3, 4, 5, 6, 8

[3] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *IEEE International Conference on Computer Vision*, pp. 1529–1537, 2015. 1, 2, 3, 8

[4] G. Lin, C. Shen, I. Reid, *et al.*, "Efficient piecewise training of deep structured models for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 3

[5] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *IEEE International Conference on Computer Vision*, pp. 1520–1528, 2015. 1, 2, 3

[6] G. Bertasius, J. Shi, and L. Torresani, "Semantic segmentation with boundary neural fields," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1, 2, 3, 8

[7] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (VOC) challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, 2010. 1, 2, 5

[8] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision*, pp. 740–755, 2014. 1

[9] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *European Conference on Computer Vision*, 2016. 1

[10] D. Lin, J. Dai, J. Jia, K. He, and J. Sun, "Scribblesup: Scribble-supervised convolutional networks for semantic segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[11] G. Papandreou, L.-C. Chen, K. P. Murphy, and A. L. Yuille, "Weakly-and semi-supervised learning of a deep convolutional network for semantic image segmentation," in *IEEE International Conference on Computer Vision*, pp. 1742–1750, 2015. 1, 6, 7, 8

[12] J. Dai, K. He, and J. Sun, "Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation," in *IEEE International Conference on Computer Vision*, pp. 1635–1643, 2015. 1

[13] D. Pathak, E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional multi-class multiple instance learning," in *International Conference on Learning Representations*, 2015. 1, 2, 5, 6, 7, 8

[14] P. O. Pinheiro and R. Collobert, "From image-level to pixel-level labeling with convolutional networks," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1713–1721, 2015. 1, 2, 5, 6, 7, 8

[15] N. Pourian, S. Karthikeyan, and B. Manjunath, "Weakly supervised graph based semantic segmentation by learning communities of image-parts," in *IEEE International Conference on Computer Vision*, pp. 1359–1367, 2015. 1, 2

[16] D. Pathak, P. Krahenbuhl, and T. Darrell, "Constrained convolutional neural networks for weakly supervised segmentation," in *IEEE International Conference on Computer Vision*, pp. 1796–1804, 2015. 1, 2, 5, 6, 7, 8

[17] Y. Wei, X. Liang, Y. Chen, X. Shen, M.-M. Cheng, Y. Zhao, and S. Yan, "STC: A simple to complex framework for weakly-supervised semantic segmentation," *arXiv preprint arXiv:1509.03150*, 2015. 1, 2, 6, 7, 8

[18] A. Kolesnikov and C. H. Lampert, "Seed, expand and constrain: Three principles for weakly-supervised image segmentation," in *European Conference on Computer Vision*, 2016. 1, 2, 4, 5, 6, 7, 8

[19] W. Shimoda and K. Yanai, "Distinct class-specific saliency maps for weakly supervised semantic segmentation," in *European Conference on Computer Vision*, pp. 218–234, 2016. 1, 2, 5, 6, 7, 8

[20] F. Saleh, M. S. A. Akbarian, M. Salzmann, L. Petersson, S. Gould, and J. M. Alvarez, "Built-in foreground/background prior for weakly-supervised semantic segmentation," in *European Conference on Computer Vision*, pp. 413–432, 2016. 1, 2, 4, 5, 7, 8

[21] X. Qi, Z. Liu, J. Shi, H. Zhao, and J. Jia, "Augmented feedback in semantic segmentation under image level supervision," in *European Conference on Computer Vision*, pp. 90–105, 2016. 1, 2, 5

[22] Y. Wei, X. Liang, Y. Chen, Z. Jie, Y. Xiao, Y. Zhao, and S. Yan, "Learning to segment with image-level annotations," *Pattern Recognition*, 2016. 1, 2, 5

[23] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik, "Hypercolumns for object segmentation and fine-grained localization," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 447–456, 2015. 2, 4

[24] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *Advances in Neural Information Processing Systems*, pp. 561–568, 2002. 2

[25] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015. 2, 4

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015. 2, 6

[27] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik, "Multiscale combinatorial grouping," in *IEEE*

Conference on Computer Vision and Pattern Recognition, pp. 328–335, 2014. 2

[28] D. Tsai, Y. Jing, Y. Liu, H. A. Rowley, S. Ioffe, and J. M. Rehg, "Large-scale image annotation using visual synset," in *IEEE International Conference on Computer Vision*, pp. 611–618, 2011. 2

[29] X. Chen and A. Gupta, "Webly supervised learning of convolutional networks," in *IEEE International Conference on Computer Vision*, pp. 1431–1439, 2015. 2

[30] D. Novotny, D. Larlus, and A. Vedaldi, "Learning the semantic structure of objects from web supervision," in *European Conference on Computer Vision*, 2016. 2

[31] M. Rubinstein, A. Joulin, J. Kopf, and C. Liu, "Unsupervised joint object discovery and segmentation in internet images," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1939–1946, 2013. 2, 7, 8

[32] E. Ahmed, S. Cohen, and B. Price, "Semantic object selection," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3157, 2014. 2

[33] X. Chen, A. Shrivastava, and A. Gupta, "Enriching visual knowledge bases via object discovery and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2027–2034, 2014. 2, 7, 8

[34] P. Krähenbühl and V. Koltun, "Efficient inference in fully connected crfs with gaussian edge potentials," in *Advances in Neural Information Processing Systems*, pp. 109–117, 2011. 3, 5, 8

[35] C. Yang, L. Zhang, H. Lu, X. Ruan, and M.-H. Yang, "Saliency detection via graph-based manifold ranking," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3166–3173, 2013. 3

[36] H. Jégou, M. Douze, and C. Schmid, "Hamming embedding and weak geometry consistency for large scale image search-extended version," in *European Conference on Computer Vision*, 2008. 4, 5

[37] B. Hariharan, P. Arbeláez, L. Bourdev, S. Maji, and J. Malik, "Semantic contours from inverse detectors," in *IEEE International Conference on Computer Vision*, pp. 991–998, 2011. 5

[38] Theano Development Team, "Theano: A Python framework for fast computation of mathematical expressions," *arXiv e-prints*, vol. abs/1605.02688, May 2016. 6

[39] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *The 22nd ACM International Conference on Multimedia*, 2014. 6

[40] A. Joulin, F. Bach, and J. Ponce, "Discriminative clustering for image co-segmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1943–1950, 2010. 7, 8

[41] A. Joulin, F. Bach, and J. Ponce, "Multi-class cosegmentation," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 542–549, 2012. 7, 8

[42] M. Najibi, M. Rastegari, and L. S. Davis, "G-cnn: an iterative grid based object detector," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 8