Multiple Instance Curriculum Learning for Weakly Supervised Object Detection
Supplementary Materials

Siyang Li
siyangl@usc.edu

Xiangxin Zhu
xiangxin@google.com

Qin Huang
qinhuang@usc.edu

Hao Xu
hxu1@usc.edu

C.-C. Jay Kuo
ckuo@sipi.usc.edu

1 Department of Electrical Engineering
University of Southern California
Los Angeles, CA, USA

2 Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA, USA

1 The Relationship between Multi-region and Single-region Saliency Maps

In the Segmentation-based Seed Growing Section (Sec. 3.2), we generalize the saliency map definition to a detector with the RoI pooling operator:

\[ A(x, y; c, r) = \sum_k \frac{\partial p(c; r)}{\partial f(x, y, k; r)} \otimes f(x, y, k; r). \]  

\[ (1) \]

The image classification network can be viewed as a detector with only one region of interest (RoI) and the RoI is the whole image, denoted as \( R \). The RoI feature map \( f(x, y, k; R) \) is then equivalent to the image feature map \( f(x, y, k) \). Also, the RoI classification score \( p(c; R) \) is simply the image classification score \( p(c) \). Thus, we have

\[ A(x, y; c, R) = \sum_k \frac{\partial p(c)}{\partial f(x, y, k)} \otimes f(x, y, k). \]  

\[ (2) \]

Since there is only one RoI, the aggregation reduce to

\[ A(x, y; c) = \hat{A}(x, y; c, R)p(c), \]  

\[ (3) \]

where \( \hat{A}(x, y; c, R) \) is obtained by resizing \( A(x, y; c, R) \) to the image size via bilinear interpolation. Padding is no longer needed, as the RoI is the whole image. If we consider the saliency

© 2017. The copyright of this document resides with its authors.
It may be distributed unchanged freely in print or electronic forms.
map before resizing (i.e., the saliency map that matches the size of the image feature map instead of the image size), which is denoted by $A^{feat}(x, y; c)$, we have

$$A^{feat}(x, y; c) = A(x, y, c, R)p(c) = p(c)\sum_k \frac{\partial p(c)}{\partial f(x, y, k)} \otimes f(x, y, k). \quad (4)$$

If the image classification network is a GAP network, then $p(c)$ is obtained via

$$p(c) = \sum_k w(k; c)\bar{f}(k) + b(c), \quad (5)$$

where $w(k; c)$ is the weight parameters of the fully connected (FC) layer corresponding to category $c$ and $b(c)$ is the bias parameter. $\bar{f}(k)$ denotes the feature vector obtained by the global average pooling operation on the feature map $f(x, y, k)$. Namely,

$$\bar{f}(k) = \frac{1}{HW} \sum_{x, y} f(x, y, k), \quad (6)$$

where $H$ and $W$ are the height and width of $f(x, y, k)$, respectively. By substituting Eq. (6) to Eq. (5), we have

$$p(c) = \frac{1}{HW} \sum_k w(k; c) \left[ \sum_{x, y} f(x, y, k) \right] + b(c). \quad (7)$$

Thus, the partial derivative in Eq. (4) is simplified as

$$\frac{\partial p(c)}{\partial f(x, y, k)} = \frac{1}{HW} w(k; c), \quad (8)$$

and Eq. (4) is reduced to

$$A^{feat}(x, y; c) = \frac{p(c)}{HW} \sum_k w(k; c) f(x, y, k). \quad (9)$$

As the saliency map is normalized so that the maximum saliency is one, the factor $\frac{p(c)}{HW}$ can be ignored and Eq.(9) is equivalent to

$$A^{CAM}(x, y; c) = \sum_k f(x, y, k)w(k; c), \quad (10)$$

which is the original CAM saliency map derived in [16].

## 2 Implementation Details

The backbone architecture for all modules is the VGG16 [11] network, for fair benchmarking with previous methods [2, 5, 7, 9, 10, 13]. The network is pretrained on the ImageNet dataset [4]. The whole system is implemented with the TensorFlow [1] library. The ImageNet pretrained model is converted from the public released Caffe model in the model zoo [1]. To train each module, we use the SGD optimizer with momentum equal to 0.9. The weight decay is set to 0.0005.

---

2.1 Detector Initialization

Classification fine-tuning. We first fine-tune the VGG16 network to perform image classification task on the PASCAL VOC dataset by removing the 1000-way out FC8 layer and adding a 20-way out FC layer as the new FC8. This network is trained on a subset of ImageNet where images from the PASCAL classes are selected. We argue this fine-tuning process does not require additional supervision as the pretrained model from the model zoo already sees all the images in the training set. We do not use images from PASCAL dataset for classification fine-tuning because the detector assumes each bounding box contains only one object but PASCAL images are multi-labeled. The fine-tuning process takes 10000 iterations with a constant learning rate of $10^{-4}$ and a batch size of 16.

Category-specific object proposals. In order to find the most salient candidate, we first find a small set of category-specific proposals for each existing category, due to the limited GPU memory. We extract 1000 category-independent proposals for each image via Selective Search (SS) [14]. To assign a classification confidence score to each proposal, we adopt the Fast R-CNN [6] architecture but drop the bounding box regression layers. This network is initialized from the fine-tuned model for whole image classification task mentioned above. The classification head composed by FC layers (FC6-8) followed by a softmax operation is applied to each RoI feature obtained through RoI pooling layer, so every proposal receives a classification score vector. Then the 1000 RoI score vectors are aggregated to image-level score vector by global maximum pooling,

$$p(c) = \max_r p(c; r),$$

where $p(c; r)$ denotes the probability of the $r$-th RoI being an instance of category $c$ and $p(c)$ denotes the probability that the input image contains instances of category $c$. Then the network can be trained by minimizing the multi-label cross entropy loss. The training process takes 10000 iterations with a constant learning rate of $10^{-4}$ and a batch size of 4.

The trained network is applied to all training images. We select the top 100 proposals for each existing category and the most salient object will be identified from them. The category-specific proposals are also useful in curriculum learning.

The most salient candidate detector. As mentioned in the paper, we add a saliency branch in parallel with the classification branch. More specifically, saliency branch is added in parallel with FC8 layer and takes the feature from the FC7 layer. The RoI feature for each category-specific proposal is computed by feeding the correspondingly cropped image region into the network. RoI features are computed on the fly and the network is end-to-end trained. We use the network fine-tuned for whole image classification to initialize the detector. The newly added saliency branch is randomly initialized. The training process takes 10000 iterations with a constant learning rate of $10^{-4}$ and a batch size of 4.

2.2 Segmentation-based Seed Growing (SSG)

We use the DeepLab-LargeFOV [3] segmentation network in this module. The backbone is the VGG16 network. The training parameters are set identical with SEC[8]: the batch size is 15; the base learning rate is $10^{-3}$ with a decay rate equal to 0.1 and decay step equal to 2000; the total training iteration number is 8000. The network is initialized from the model pretrained on ImageNet.

\[^{2}\text{http://www.cs.jhu.edu/~alanlab/ccvl/init_models}\]
Segmentation seeds. The seeds used for the first round of the segmentation training are from a classification network which is applied to the entire image. Since CAM \[16\] does not apply to the standard VGG-16 network \[11\], we follow the modification in SEC \[8\] to make CAM compatible with VGG-16. The network is finetuned by training images from PASCAL VOC datasets with image-level labels. To generate seeds for existing categories, the saliency maps are computed via Eq. \((10)\) and the maximum saliency is normalized to 1. The threshold is set to 0.2. The background saliency maps are computed via Grad \[12\]. The detailed equation is given in the paper. Then they are normalized such that the maximum and the minimum background saliency is 1 and 0, respectively. The threshold is set to 0.9 to generate seeds or we use the top 10\% most background-salient regions. We take the one with more background seeds.

In the later training rounds on the segmentation network, the background seeds are kept identical while the foreground seeds are updated based on the trained detector, following Eq. \((1)\). The thresholding parameter is kept identical.

Post-processing of segmentation results. Since the trained segmentation network is applied to training images, where the existing and absent object categories are known, the category label assigned to each pixel is obtained by

\[
M(x, y) = \arg \max_{c \in C^+} p(c; x, y),
\]

where \(p(c; x, y)\) is the probability that pixel \((x, y)\) belongs to category \(c\) and \(C^+\) is the existing categories including background.

2.3 Curriculum learning

Example selection. The threshold \(T\) for intersection over union (IoU) is set to 0.5 in our experiments except the final training round, where all examples are selected.

Training. The Fast RCNN network is initialized from the MSC detector with the saliency branch dropped. The bounding box regression layers are initialized randomly. Because we use a subset of training examples, we do not take all proposals from SS \[14\]. Instead, when only a part of examples are selected from the entire training set, we use the category-specific proposals mentioned in Sec. 2.1, as one image containing both “person” and “chair” may be selected as easy example for “person” but not for “chair” by the easiness measurement. By taking category-specific proposals only, we avoid the detector seeing proposals that overlap largely with existing but not selected categories. Using category-specific proposals also reduces the training time. Only in the last training round where all examples are involved, the category-independent proposals are fed into the detector during training, so that the trained detector can be directly applied to test images, where the existing categories are unknown.

The multiple instance curriculum learning (MICL) takes two rounds. In the first round, the simple examples are selected by measuring the consistency between the results from MSC detector and the segmenter. About 2300 examples are selected for the first MICL training round. Segmentation seeds are updated based on the detector from training round 1 and the segmenter is retrained (initialized from the ImageNet-pretrained model). Then, objects are re-localized by the detector and the segmenter. We take the average of the bounding boxes from those two and use those boxes as the pseudo location labels. In the second training round, all examples are used and the detector is initialized by the detector from training round one. For all training rounds, we use the SGD optimizer with a momentum of 0.9 for 40000 iterations. The learning rate is set to \(10^{-4}\) and the batch size is 2. Proposals with
IoU larger than 0.5 are treated as positive samples; those with IoU between 0.1 and 0.5 are negative samples, by following the setting in [6].

3 Supplementary Experiments

3.1 Segmentation-based Seed Growing

Saliency maps. To evaluate the saliency maps, we adopt the “pointing game” metric proposed in ExciteBack [15]. If the location of the maximum saliency for an existing category $c$, which is a point on the maps, fall in one of the ground truth object boxes for $c^3$, it is counted as a “hit”; otherwise it is a “miss”. To evaluate the precision of the maximum saliency point, the percentage of hits is computed. Differently with ExciteBack, the precision evaluation is conducted on the training set, as they are used to grow segmentation masks on the training images. The precision is calculated for each category separately and the mean of those precision values are shown Tab. 1. The saliency maps from the detector trained on the easy images achieves 2.5% higher in precision compared with the maps from the classification network.

<table>
<thead>
<tr>
<th>Saliency maps</th>
<th>Precision (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification network (single-region)</td>
<td>88.6</td>
</tr>
<tr>
<td>Detection network (multi-region)</td>
<td>91.1</td>
</tr>
</tbody>
</table>

Table 1: The precision of the maximum saliency point on VOC07 trainval set.

Similarly, the precision of segmentation seeds generated by thresholding the saliency maps is evaluated by checking if the seeds fall into object bounding boxes. The same thresholding parameters are applied to the saliency maps from the classifier and the detector. We also evaluate the recall of the segmentation seeds by the ratio between the number of hit seeds and the area of the object bounding boxes. The results are shown in Tab. 2. By generating saliency maps from the detector trained on easy examples, we are able to improve the precision and recall by 1.7% and 14.5%, respectively.

<table>
<thead>
<tr>
<th>Seeds</th>
<th>Precision (%)</th>
<th>Recall (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification network (single-region)</td>
<td>82.6</td>
<td>25.3</td>
</tr>
<tr>
<td>Detection network (multi-region)</td>
<td>84.3</td>
<td>39.8</td>
</tr>
</tbody>
</table>

Table 2: The precision and recall of segmentation seeds on VOC07 trainval set.

Segmentation Results. The CorLoc metric is applied to the bounding boxes generated from the trained segmentation network based on classification or detection saliency maps. Results are given in Tab. 3. After updating the segmentation seeds in the re-localization step, the re-trained segmenter improves the CorLoc by 1.3% on the VOC07 trainval set.

---

$^3$Even if the point falls in object boxes, it may still fall on background. However, since not all images from VOC07 trainval set have segmentation mask labels, checking the point against bounding boxes is the best estimate on the point precision of saliency maps, same for segmentation seed evaluation.
<table>
<thead>
<tr>
<th>Seeds for training</th>
<th>CorLoc (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification network (single-region)</td>
<td>53.5</td>
</tr>
<tr>
<td>Detection network (multi-region)</td>
<td>54.8</td>
</tr>
</tbody>
</table>

Table 3: The CorLoc of the bounding boxes from the segmenter on VOC07 trainval set.

### 3.2 Error Analysis on the Detector and the Segmenter

In the paper, we conducted error analysis on the MSC detector and the SSG segmenter (Sec 4.3). The mis-localized boxes are categorized into three types: too small, too large and others. A box is *too small* if the following conditions hold:

\[
\frac{\text{Area}(B^* \cap B^{GT})}{\text{Area}(B^*)} \geq T, \\
\text{Area}(B^*) < \text{Area}(B^{GT}),
\]

where \( B^* \) stands for the bounding box from the detector \( B^{DET} \) or the segmenter \( B^{SSG} \) and \( B^{GT} \) denotes the ground truth box. Similarly, a box is *too large* if

\[
\frac{\text{Area}(B^* \cap B^{GT})}{\text{Area}(B^{GT})} \geq T, \\
\text{Area}(B^*) > \text{Area}(B^{GT}).
\]

If a box fits neither of the two groups of conditions, it is categorized as *others*. The threshold \( T \) is set to 0.5 in our experiments. The error percentage is plotted in Fig. 6 in the paper.

### 3.3 Visualized Easy and Hard Examples

The easiness is measured by the consistency between the detector and the segmenter. Some visualized easy and hard examples are shown in Figs. 1 and 2.
Figure 1: Visualized easy examples selected by measuring the consistency between boxes from the segmenter (in red) and the detector (in green). Those examples are used for the next round of detector training.
Figure 2: Visualized hard examples selected by measuring the consistency between boxes from the segmenter (in red) and the detector (in green). Those examples are NOT used for the next round of detector training.
3.4 Visualized Detection Results

We provide more visualized detection results in Fig. 3, to support Fig. 4 in the paper.

Figure 3: Qualitative detection results, where the correctly detected ground truth objects are in green boxes and blue boxes represent correspondingly predicted locations. Objects that we fail to detect are in yellow boxes and false positive detections are in red.
References


