# Weakly Supervised Instance Segmentation by Deep Community Learning
## *Supplementary Materials*

Jaedong Hwang[1*] Seohyun Kim[1*] Jeany Son[2] Bohyung Han[1]
[1]ECE & ASRI, Seoul National University, Seoul, Korea
[2]ETRI, Daejeon, Korea
[1]{jd730, goodbye61, bhhan}@snu.ac.kr, [2]jeany@etri.re.kr

## 1. Details of Our Framework

This section discusses more details regarding our feature extractor, object detection and instance mask generation modules, which are described in our main paper.

### 1.1. Feature Extractor

We use ResNet50 [50] as a backbone network, which is pretrained on ImageNet. For object detection, one SPP layer is attached after res4, followed by res5. The output of the last residual block is shared with IMG and segmentation modules through upsampling. The IMG module employs multiple level of $28 \times 28$ features from outputs of SPP layers attached to res3 and res4, and upsampled res5 output. These features are given to the weighted GAP and the classification layers following one convolution layer for each level of the CAM subnetwork. For instance segmentation, the upsampled output of res5 is used. On our implementation, batch normalization is replaced to group normalization [53] due to the small batch size.

### 1.2. Object Detection Module

Object detection module is composed of detector and regressor parts. Note that any weakly supervised object detection algorithm can be used as the detector in the proposed framework.

#### 1.1.1 Detector

We adopt OICR [36] for the detector. OICR is one of the most commonly used algorithm for weakly supervised object detection relying on multiple instance learning [35, 36, 42]. The model has two parts, multiple instance detection network (MIDN) and refinement layers.

[*] Equal contribution.

#### 1.2.1 MIDN

MIDN is based on the Weakly Supervised Deep Detection Network (WSDDN) [4], which has two parallel fully connected layers for classification and detection, respectively and they are followed by two separate softmax layers. For classification, the softmax layer is given by

$$[\sigma_{\text{cls}}(\mathbf{x}^c)]_{ij} = \frac{e^{x^c_{ij}}}{\sum_{k=1}^{C} e^{x^c_{kj}}}, \tag{1}$$

where $x^c_{ij}$ denotes the classification score for the $i^{\text{th}}$ class of the $j^{\text{th}}$ proposal and $C$ denotes the number of classes. On the other hand, the softmax layer for detection branch is given by

$$[\sigma_{\text{det}}(\mathbf{x}^d)]_{ij} = \frac{e^{x^d_{ij}}}{\sum_{k=1}^{|R|} e^{x^d_{ik}}}, \tag{2}$$

where $x^d_{ij}$ denotes the detection score for the $i^{\text{th}}$ class of the $j^{\text{th}}$ proposal and $|R|$ is the number of proposals.

The final score, $\mathbf{z} \in \mathbb{R}^{C \times |R|}$ is defined as

$$\mathbf{z} = \sigma_{\text{cls}}(\mathbf{x}^c) \odot \sigma_{\text{det}}(\mathbf{x}^d), \tag{3}$$

where $\odot$ is the Hadamard product. The image-level classification score $\boldsymbol{\phi}$ is given by the sum of $\mathbf{z}$ over all proposals. By using the image-level score, the loss from MIDN $\mathcal{L}_{\text{cls}}$ is defined as an image-level cross-entropy, which is described in Eq. 3 in our main paper.

#### 1.2.2 Refinement Layer

Once MIDN predicts a class of each proposal, a refinement layer revises the labels by leveraging object classification scores from the previous stage. The refinement layer finds the proposal with the highest rank in each class, which is considered as a seed. Each proposal is given a label from the highest overlapping seed if its IoU (Intersection over Union) with the seed is higher than a threshold, 0.5; otherwise, it is labeled as a background class. The weight of the proposal

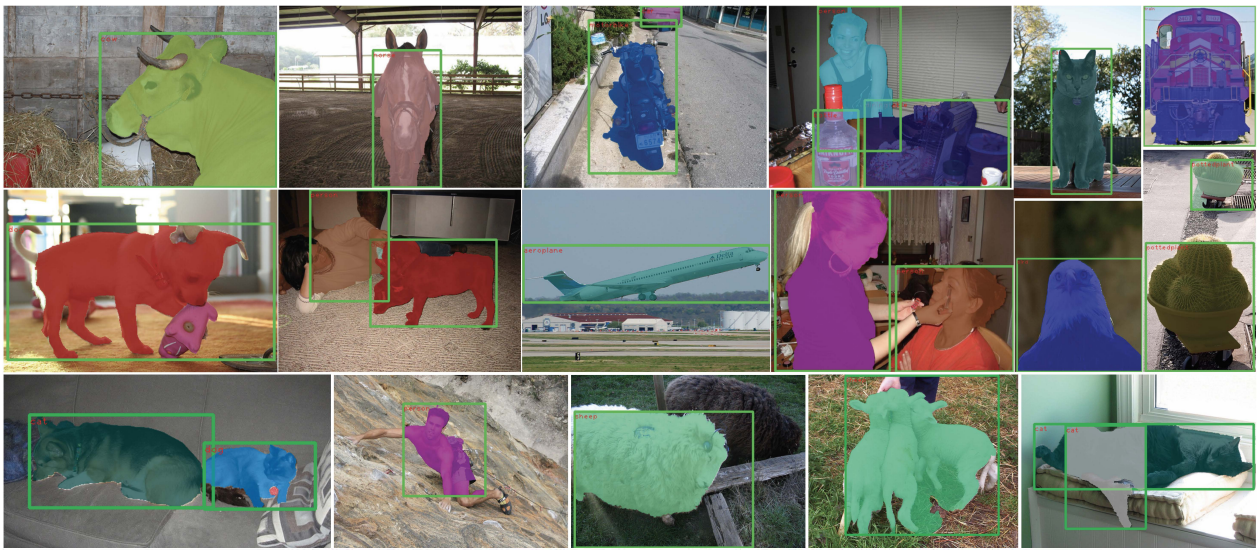

Figure 1. Qualitative results of instance segmentation on PASCAL VOC 2012 segmentation *val* set. Results in the first two rows are the success cases and those in the last row are failure cases

$w_r$ is given by the class score of the seed. Hence, the loss of the $k^{\text{th}}$ refinement layer, $\mathcal{L}^k_{\text{refine}}$ is defined as a weighted cross-entropy loss as described in Eq. 4 in our main paper.

#### 1.1.2 Regressor

For bounding box regression, we attach two fully connected layers after res5 which has 2048 channels. The final output of our regressor has a dimension of 4 for class-agnostic manner instead of $4C$ where $C$ is the number of classes for traditional class-specific manner. It means that class-agnostic regressor is shared with all classes.

During training, a proposal and its nearest pseudo-ground-truth proposal pair $(\mathrm{p}, g)$ is converted to a regression offset $t = [t_x, t_y, t_w, t_h]$ as follows:

$$\begin{aligned} t_x &= (g_x - \mathrm{p}_x)/\mathrm{p}_w, \\ t_y &= (g_y - \mathrm{p}_y)/\mathrm{p}_h, \\ t_w &= \log(g_w/\mathrm{p}_w), \\ t_h &= \log(g_h/\mathrm{p}_h), \end{aligned} \tag{4}$$

where $g = [g_x, g_y, g_w, g_h]$ is a target pseudo-ground-truth proposal for a proposal, $\mathrm{p} = [\mathrm{p}_x, \mathrm{p}_y, \mathrm{p}_w, \mathrm{p}_h]$.

### 1.3. Instance Mask Generation (IMG) Module

We use CAM [43] for instance mask generation module. It can be substituted by other object localization algorithms based on image-level labels such as Grad-CAM [52] and Grad-CAM++ [49].

#### 1.2.1 Class Activation Map (CAM)

CAM [45] highlights areas of discriminative parts of objects over each class and is often used for the pseudo-ground-truth for weakly supervised semantic segmentation. CAM is built on a classification task leveraging Global Average Pooling (GAP) [51]. It is applied to the last convolutional layer followed by a fully connected layer and a softmax layer to predict image-level class labels. For each class c, CAM, $\mathbf{M}_c(x, y)$ is defined as follows:

$$\mathbf{M}_c(x, y) = \mathbf{w}_c^T \cdot \mathbf{F}(x, y), \tag{5}$$

where $\mathbf{F}(x, y)$ is a feature vector from the last convolutional layer with respect to spatial grid $(x, y)$, and $\mathbf{w}_c$ is a weight vector of fully connected layer.

## 2. Time Cost of Post Processing

Note that our model without post-processing has competitive results compared to existing methods, and our post-processing is finding best matching MCG proposal for each predicted mask. The computational cost for post-processing is not signficant compared to our main algorithm based on a deep neural network. Specifically, the inference through our network takes 4 seconds per image (5 multi-scales with flip) on a single TITAN Xp GPU but the post-processing takes $0 \sim 4$ seconds on a CPU.

## 3. Additional Ablation Study

### 3.1. Multiple CAMs

We present the instance segmentation performance at $\text{mAP}_{0.5}$ with respect to the number of CAMs in Table 1.

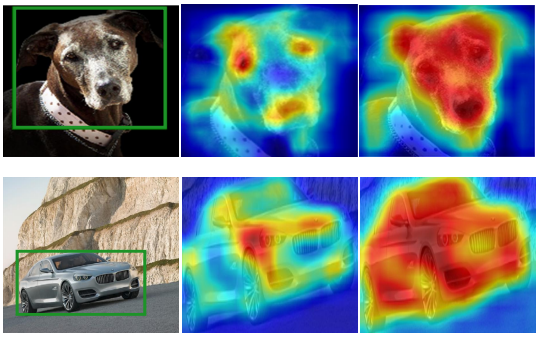

Figure 2. Comparison between the outputs of the conventional CAM network (middle) and one with feature smoothing (right) for two images.

Table 1. Accuracy of the various number of CAMs in IMG module based on ResNet50 without REG and IS modules on PASCAL VOC 2012 segmentation *val* set.

| The number of CAMs | 1 | 2 | 3 (ours) | 4 |
|---|---|---|---|---|
| $mAP_{0.5}$ | 29.7 | 30.6 | **32.8** | 31.3 |

The results come from our Detector + IMG module which does not have REG and IS modules and the postprocesing to directly show the effectiveness of multiple CAMs. The multi-scale representations are helpful to capture whole objects rather than discriminative parts only.

## 4. Additional Qualitative Results

### 4.1. Instance Segmentation

Figure 1 shows additional instance segmentation results. Images in the first two rows are success cases and those in the last row are failure cases. In the failure cases, the model is confused with dog and cat and cannot detect human hands and leg, dark sheep. differentiate adjacent three sheep, and remove false positive.

### 4.2. Feature Smoothing

To penalize CAM focusing excessively on discriminative parts on target objects, we smooth the input features to CAM networks using a non-linear activation function. As illustrated in Figure 2, the function helps produce more spatially regularized activation maps which are more appropriate to enclose entire target objects by segmentation.

### 4.3. Bounding Box Regression

We qualitatively compare our model with class-agnostic regressor and with class-specific regressor on Figure 3. Our model with class-agnostic regressor achieves better performance than with class-specific regressor. The difference between two regressors is remarkable on "cat" and "dog" classes. With class-agnostic regressor, our model detects their entire bodies while the model with class-specific counterpart still spotlights their discriminative parts, faces. Figure 4 presents the effectiveness of our class-agnostic regressor compared to our model without regressor on PASCAL VOC segmentation *val* set.

## References


[49] Aditya Chattopadhyay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-CAM++: Improved Visual Explanations for Deep Convolutional Networks. In *WACV*, 2018.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016.

[51] Min Lin, Qiang Chen, and Shuicheng Yan. Network In Network. *arXiv*, abs/1312.4400, 2013.

[52] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. In *ICCV*, 2017.

[53] Yuxin Wu and Kaiming He. Group Normalization. In *ECCV*, 2018.

Figure 3. Qualitative results regarding class-specific and class-agnostic regressors on PASCAL VOC 2012 segmentation *val* set. Red rectangle is a ground-truth, blue rectangle represents the output of our model with class-agnostic regressor and orange one is our model with class-specific regressor.

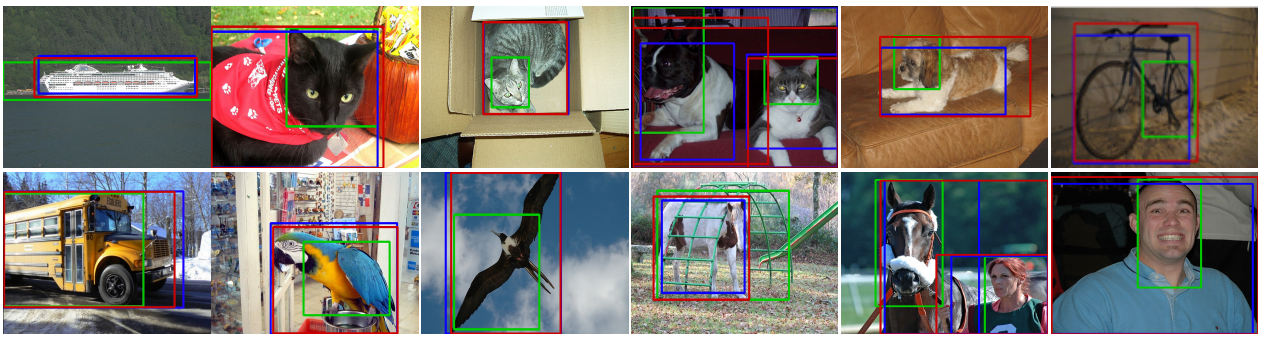

Figure 4. Qualitative results of detection on PASCAL VOC 2012 segmentation *val* set. Red rectangle indicates ground-truth, green rectangle is generated by our model without regressor, and blue one represents the output of our model with class-agnostic regressor.