Class Re-Activation Maps for Weakly-Supervised Semantic Segmentation

Zhaozheng Chen[1]   Tan Wang[2]   Xiongwei Wu[1]
Xian-Sheng Hua[3]   Hanwang Zhang[2]   Qianru Sun[1]

[1]Singapore Management University   [2]Nanyang Technological University   [3]Damo Academy, Alibaba Group

zzchen.2019@phdcs.smu.edu.sg   {tan317, hanwangzhang}@ntu.edu.sg
xiansheng.hxs@alibaba-inc.com   {xwwu,qianrusun}@smu.edu.sg

Abstract

*Extracting class activation maps (CAM) is arguably the most standard step of generating pseudo masks for weakly-supervised semantic segmentation (WSSS). Yet, we find that the crux of the unsatisfactory pseudo masks is the binary cross-entropy loss (BCE) widely used in CAM. Specifically, due to the sum-over-class pooling nature of BCE, each pixel in CAM may be responsive to multiple classes co-occurring in the same receptive field. As a result, given a class, its hot CAM pixels may wrongly invade the area belonging to other classes, or the non-hot ones may be actually a part of the class. To this end, we introduce an embarrassingly simple yet surprisingly effective method: Reactivating the converged CAM with BCE by using softmax cross-entropy loss (SCE), dubbed **ReCAM**. Given an image, we use CAM to extract the feature pixels of each single class, and use them with the class label to learn another fully-connected layer (after the backbone) with SCE. Once converged, we extract ReCAM in the same way as in CAM. Thanks to the contrastive nature of SCE, the pixel response is disentangled into different classes and hence less mask ambiguity is expected. The evaluation on both PASCAL VOC and MS COCO shows that ReCAM not only generates high-quality masks, but also supports plug-and-play in any CAM variant with little overhead. Our code is public at https://github.com/zhaozhengChen/ReCAM.*

1. Introduction

Weakly-supervised semantic segmentation (WSSS) aims to lower the high cost in annotating "strong" pixel-level masks by using "weak" labels instead, such as scribbles [29, 36], bounding boxes [7, 35], and image-level class labels [1, 19, 27, 28, 42, 46]. The last one is the most economic yet challenging budget and thus is our focus in this paper. A common pipeline has three steps: 1) training a
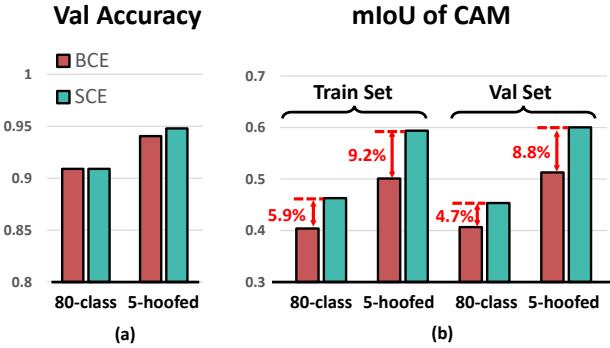

Figure 1. We train two models respectively using binary cross entropy (BCE) and softmax cross entropy (SCE) losses. Our `train` and `val` sets contain only single-label images of MS COCO [30]. "80-class" model uses the complete label set. "5-hoofed" model is trained on only the samples of 5 hoofed animals each causing false positive flaws to another, e.g., between `cow` and `horse`.

multi-label classification model with the image-level class labels; 2) extracting the class activation map (CAM) [51] of each class to generate a 0-1 mask, with potential refinement such as erosion and expansion [1, 23]; and 3) taking all-class masks as pseudo labels to learn the segmentation model in a standard fully-supervised fashion [5, 6, 41]. There are different factors affecting the performance of the final segmentation model, but the classification model in the first step is definitely the root. We often observe two common flaws. In the CAM of an object class A, there are 1) false positive pixels that are activated for class A but have the actual label of class B, where B is usually a confusing class to A rather than `background`—a special class in semantic segmentation; and 2) false negative pixels that belong to class A but are wrongly labeled as `background`.

Findings. We point out that these flaws are particularly obvious when the model is trained with the binary cross-entropy (BCE) loss with sigmoid activation function. Specifically, the sigmoid function is $\frac{e^x}{e^x+1}$ where x denotes the prediction logit of any individual class. The output is

fed into the BCE function to compute a loss. This loss represents the penalty strength for misclassification corresponding to x. The BCE loss is thus not class mutually exclusive—the misclassification of one class does not penalize the activation on others. This is indispensable for training multi-label classifiers. However, when extracting CAM via these classifiers, we see the drawbacks: non-exclusive activation across different classes (resulting in false positive pixels in CAM); and the activation on total classes is limited (resulting in false negative pixels) since partial activation is shared.

Motivation. We conduct a few toy experiments to empirically show the poor quality of CAM when using BCE. We pick single-label training images in MS COCO 2014 [30] (about 20% in the `train` set) to train 5-class and 80-class classifiers, respectively, where for 5-class, we pick 5 hoofed animal classes (e.g., `horse` and `cow`) that suffer from the confused activation. We train every model using two losses, respectively: BCE loss and softmax cross-entropy (SCE) loss—the most common one for classification. We use the single-label images in `val` set to evaluate models' classification performance, as shown in Figure 1 (a), and use the single-label images in both `train` and `val` sets to inspect models' ability of activating correct regions on the objects—the quality of CAM, as compared in Figure 1 (b).

Intrigued, 1) for 80-class models, BCE and SCE yield equal-quality classifiers but clearly different CAMs, and 2) the CAMs of SCE models are of higher mIoU, and this superiority is almost maintained for validation images. A small yet key observation is that for 5 hoofed animal classes, BCE shows weaker to classify them. We point out this is because the sigmoid activation function of BCE does not enforce class-exclusive learning, confusing the model between similar classes. However, SCE is different. Its softmax activation function $\frac{e^x}{e^x + \Sigma_y e^y}$, where y denotes the prediction of any negative class, explicitly enforces class-exclusion by using exponential terms in the denominator. SCE encourages to improve the logit of ground truth and penalizes others simultaneously. This makes two effects on CAM: 1) reducing false positive pixels which confuse the model among different classes; and 2) encouraging the model to explore class-specific features that reduce false negative pixels. We show the empirical evidence in Figure 1 (b) where the mIoU improvements by SCE over BCE are especially significant for 5-hoofed. **Please note** that the functions of BCE and SCE are different. To give more concrete comparison between them, we elaborate the comparison between their produced gradients in Section 4.2, theoretically and empirically.

Our Solution. Our intuition is to use SCE loss function to train a model for CAM. However, directly replacing BCE with SCE does not make sense for multi-label classification tasks where the probabilities of different classes are not in-dependent [34, 47]. Instead, we use SCE as an additional loss to *Re*activate the model and generate *Re*CAM. Specifically, when the model converges with BCE, for every individual class labeled in the image, we extract the CAM in the format of normalized soft mask, i.e., without hard thresholding [40, 51]. We apply all masks on the feature (i.e., the feature map block output by the backbone), respectively, each "highlighting" the feature pixels contributing to the classification of a specific class. In this way, we branch the multi-label feature to a set of single-label features. We can thus use these features (and labels) to train a multi-class classifier with SCE, e.g., by plugging another fully-connected layer after the backbone. The SCE loss penalizes any misclassification caused by either poor features or poor masks. Then, backpropagating its gradients improves both. Once converged, we extract ReCAM in the same way of CAM.

Empirical Evaluations. To evaluate the ReCAM, we conduct extensive WSSS experiments on two popular benchmarks of semantic segmentation, PASCAL VOC 2012 [9] and MS COCO 2014 [30]. A standard pipeline of WSSS is to use CAM [51] as seeds and then deploy refinement methods such as AdvCAM [23] or IRN [1] to expand the seeds to pseudo masks—the labels used to train the segmentation model. We design the following comparisons to show the generality and superiority of ReCAM. 1) *ReCAM as seeds, too*. We extract ReCAM and use refinement methods afterwards, showing that the superiority over CAM is maintained after strong refinement steps. 2) *ReCAM as another refinement method*. We compare ReCAM with existing refinement methods, regarding the quality of generated masks as well as the computational overhead added to baseline CAM [51]. In the stage of learning semantic segmentation models, we use the ResNet-based DeepLabV2 [5], DeepLabV3+ [6] and the transformer-based UperNet [41].

Our Contributions in this paper are thus two-fold. 1) A simple yet effective method ReCAM for generating pseudo masks for WSSS. 2) Extensive evaluations of ReCAM on two popular WSSS benchmarks, with or without incorporating advanced refinement methods [1, 23].

2. Related Works

The training of multi-label classification and semantic segmentation models are almost uniform in the works of WSSS. Below, we introduce only the variants for seed generation and mask refinement.

Seed Generation. Vanilla CAM [51] first scales the feature maps (e.g., output by the last residual block) by using the FC weights learned for each individual class. Then, it produces the seed masks by channel-wise averaging, spatial-wise normalization, and hard thresholding (see Section 3). Based on this CAM, there are improved methods. GAIN [25] applies the CAM on original images to

generate masked images, and minimizes the model prediction scores on them, forcing the model to capture the features in other regions (outside the current CAM) in the new training. A similar idea was used in erasing-based methods [14, 20, 39, 49]. The difference is erasing methods directly perturbed the regions (inside the CAM) and fed the perturbed images into the model to generate the next-round CAM that is expected to capture new regions. Score-CAM [37] is a different CAM method. It replaces the FC weights used in vanilla CAM with a new set of scores predicted from the images masked by channel-wise (not class-specific) activation maps. EDAM [40] is a recent work of using CAM-based perturbation to optimize an additional classifier.One may argue that our ReCAM is similar to EDAM. We highlight two differences. 1) EDAM uses an extra layer to produce class-specific soft masks, while our soft masks are simply from the byproducts of CAM without needing any parameters. 2) EDAM still uses BCE loss for training with perturbed input, while we inspect the limitations of BCE and propose a different training method by leveraging SCE (see Section 4.2).

Mask Generation. Seed masks generated by CAM or its variants can go through a refinement step. One category of refinement methods [1, 2, 4, 42] propagate the object regions in the seed to semantically similar pixels in the neighborhood. It is achieved by the random walk [33] on a transition matrix where each element is an affinity score. The related methods have different designs of this matrix. PSA [2] is an AffinityNet to predict semantic affinities between adjacent pixels. IRN [1] is an inter-pixel relation network to estimate class boundary maps based on which it computes affinities. Another method is BES [4] that learns to predict boundary maps by using CAM as pseudo ground truth. All these methods introduced additional network modules to vanilla CAM. Another category of refinement methods [15,17,22,26,44,48] utilize saliency maps [13,50]. EPS [24] proposed a joint training strategy to combine CAM and saliency maps. EDAM [40] introduced a post-processing method to integrate the confident areas in the saliency map into CAM. In experiments, we plug ReCAM in them to evaluate its performance with additional saliency data. A more recent category of methods leverage iterative post-processing to refine CAM. OOA [16] ensembles the CAM generated in multiple training iterations. CONTA [45] iterated through the whole process of WSSS including a sequence of model training and inference. AdvCAM [23] used the gradients with respect to the input image to perturb the image, and iteratively find newly activated pixels. Overall, these refinement methods are based on the seed generated by CAM [51]. Our ReCAM is a method of leveraging SCE to reactivate more pixels in CAM, and is thus convenient to incorporate it. We conduct extensive plug-and-play experiments in Section 5.

Other ideas of improving CAM include ICD [10] that learned intra-class boundaries on feature manifolds, SC-CAM [3] that learned fine-grained classification models (with pseudo fine-grained labels); and SEAM [38] that enforced the consistency of CAM extracted from different transformations of the image. A recent work RIB [21] did a careful analysis based on the theory of information bottleneck, and proposed to retrain the multi-label classification model without the last activation function. Our ReCAM does not remove any activation function but adds a softmax activation based loss (SCE), as shown in Figure 3. Another difference is in the inference stage. RIB needs 10 iterations of feeding forward and backward for each test image, but ReCAM feeds forward the image only once. For example, on PASCAL VOC 2012 [9] dataset, RIB costs 8 hrs in its inference (its training cost is the same as vanilla CAM), while our total cost over vanilla CAM is only 0.6 hrs.

3. Preliminaries

CAM. The first step of CAM [51] is to train a multi-label classification model with global average pooling (GAP) followed by a prediction layer (e.g., the FC layer of a ResNet [12]). The prediction loss on each training example is computed by BCE function in the following formula:

$$\mathcal{L}_{bce} = -\frac{1}{K}\sum_{k=1}^{K} y\,[k]\log\sigma\,(z[k]) + (1-y[k])\log\left[1-\sigma\,(z[k])\right],\tag{1}$$

where $z[k]$ denotes the prediction logit of the k-th class, $\sigma(\cdot)$ is the sigmoid function, and K is the total number of foreground object classes (in the dataset). $y[k] \in \{0,1\}$ is the image-level label for the k-th class, where 1 denotes the class is present in the image and 0 otherwise.

Once the model converges, we feed the image $\boldsymbol{x}$ into it to extract the CAM of class k appearing in $\boldsymbol{x}$:

$$\text{CAM}_k(\boldsymbol{x}) = \frac{\text{ReLU}\,(\boldsymbol{A}_k)}{\max\,(\text{ReLU}\,(\boldsymbol{A}_k))},\ \boldsymbol{A}_k = \mathbf{w}_k^\top f(\boldsymbol{x}),\quad(2)$$

where $\mathbf{w}_k$ denotes the classification weights (e.g., the FC layer of a ResNet) corresponding to the k-th class, and $f(\boldsymbol{x})$ represents the feature maps of $\boldsymbol{x}$ before the GAP.

Please note that for simplicity, we assume the classification head of the model is always a single FC layer, and use $\mathbf{w}$ to denote its weights in the following.

Pseudo Masks. There are a few options to generate pseudo masks from CAM: 1) thresholding CAM to be 0-1 masks; 2) refining CAM with IRN [1]—a widely used refinement method; 3) iteratively refining CAM through the classification model, e.g., using AdvCAM [23]; and 4) cascading options 3 and 2. In Figure 2, we illustrate these options with our ReCAM plugged in. We elaborate these in Section 4.1.

Semantic Segmentation. This is the last step of WSSS. We use the pseudo masks to train the semantic segmentation
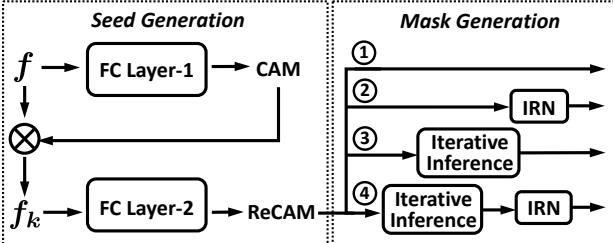
Figure 2. The pipeline of using ReCAM to generate pseudo masks for WSSS. There are two steps, seed generation and mask generation, and our ReCAM is taken as a module plugged in seed generation step. The mask generation has a few options: 1) take the ReCAM directly as pseudo mask; 2) refine the ReCAM with the most common refinement method IRN [1]; 3) iteratively infer better masks via the model of ReCAM; and 4) cascade options 3 and 2. The details of learning ReCAM model are shown in Figure 3. Table 2 shows the overall comparison results for these options.

model in a fully-supervised way. The objective function is as follows:

$$\mathcal{L}_{ss} = -\frac{1}{HW} \sum_{i=1}^{H} \sum_{j=1}^{W} \frac{1}{K+1} \sum_{k=1}^{K+1} y_{i,j}[k] \log \frac{\exp(z_{i,j}[k])}{\sum_k \exp(z_{i,j}[k])}, \tag{3}$$

where $y_{i,j}$ and $z_{i,j}$ denote the label and the prediction logit at pixel (i, j), respectively. $y_{i,j}[k]$ and $z_{i,j}[k]$ denote the k-th element of $y_{i,j}$ and $z_{i,j}$, respectively. H and W are the height and width of the image. K is the total number of classes. $K+1$ means including the `background` class.

For implementation, we deploy DeepLab variants [5, 6] with ResNet-101 [12], following related works [1, 21, 23, 45]. In addition, we employ a recent model UperNet [41] with a stronger backbone—Swin Transformer[1] [31].

4. Class Re-Activation Maps (ReCAM)

In Section 4.1, we elaborate our method of reactivating the classification model and extracting ReCAM from it. Note that we also use "ReCAM" to name our method. In Section 4.2, we justify the advantages of class-exclusive learning in ReCAM, by comparing the gradients of SCE with BCE theoretically and empirically.

4.1. ReCAM Pipeline

Backbone and Multi-Label Features. We use a standard ResNet-50 [12] as our backbone (i.e., feature encoder) to extract features, following related works [1, 21, 23, 45].

Given an input image x and its multi-hot class label $y \in \{0, 1\}^K$, we denote the output of feature encoder as $f(x) \in \mathbb{R}^{W \times H \times C}$. C denotes the number of channels, H and W denote the height and width, respectively. K is the total

---

[1]Please note that we did not implement our method on transformer-based classification models and will take this as the future work.
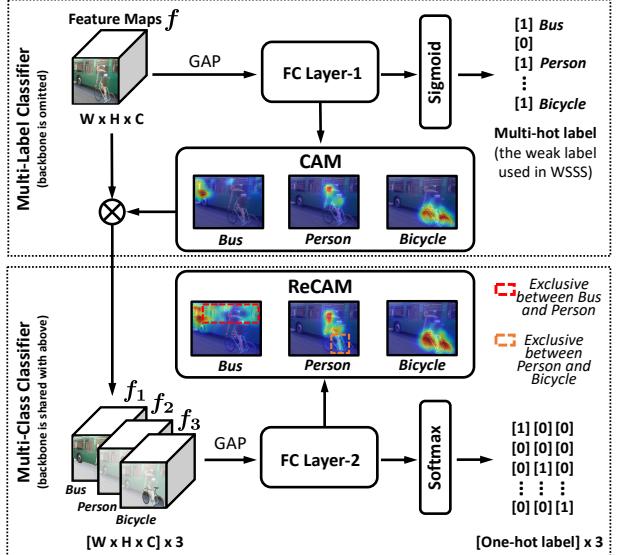

Figure 3. The training framework of ReCAM. In the upper block, it is the conventional training of multi-label classifiers using BCE. The feature extraction via backbone is omitted for conciseness. We extract the CAM for each class and then apply it (as a normalized soft mask) on the feature maps f to obtain the class-specific feature f_k. In the lower block, we use f_k and its single label to learn multi-class classifiers with SCE loss. The gradients of this loss are backpropagated through the whole network including backbone.

number of foreground classes in the dataset. Please note that in Figure 3, 1) the feature extraction process is omitted for conciseness; and 2) the feature $f(x)$ is written as f in the upper block and usually represents multiple objects.

FC Layer-1 with BCE Loss. In the conventional model of CAM, the feature $f(x)$ first goes through a GAP layer and the result is fed into a FC layer to make prediction [51]. Hence, the prediction logits can be denoted as

$$z = \mathrm{FC}_1(\mathrm{GAP}(f(x))). \tag{4}$$

Then, z and the image-level labels y are used to compute a BCE loss. An element-wise formula is given in Eq. (1).

Extracting CAM. We extract the CAM for each individual class k given the feature $f(x)$ and the corresponding weights $\mathbf{w}_k$ of the FC layer, as formulated in Eq. (2). For brevity, we denote the $\mathrm{CAM}_k(x)$ as $M_k \in \mathbb{R}^{W \times H}$.

Single-Label Feature. As shown in Figure 3, we use M_k as a soft mask to apply on $f(x)$ to extract the class-specific feature $f_k(x)$. We compute the element-wise multiplication between M_k and each channel of $f(x)$ as follows,

$$f_k^c(x) = M_k \otimes f^c(x), \tag{5}$$

where $f^c(x)$ and $f_k^c(x)$ indicate the single channel before and after the multiplication (by using M_k), c ranges from 1 to C and C is number of feature maps (i.e., channels). The feature map block $f_k(x)$ (each contains C channels) corresponds to the examples f_1, f_2, f_3 in Figure 3.

FC Layer-2 with SCE Loss. Each $f_k(\boldsymbol{x})$ has a single object label (i.e., a one-hot label where the k-th position is 1). Then, we feed it to FC Layer-2 (see Figure 3) to learn multi-class classifier, so we have new prediction logits for $\boldsymbol{x}$ as:

$$\boldsymbol{z}'_k = \text{FC}_2(\text{GAP}(f_k(\boldsymbol{x}))), \qquad (6)$$

where FC_2 has the same architecture as FC_1.

By this way, we succeed to convert the BCE-based model on multi-label images to the SCE-based model on single-label features. The SCE loss is formulated as:

$$\mathcal{L}_{sce} = -\frac{1}{\sum_{i=1}^{K} \boldsymbol{y}[i]} \sum_{k=1}^{K} \boldsymbol{y}[k] \log \frac{\exp(\boldsymbol{z}'_k[k])}{\sum_j \exp(\boldsymbol{z}'_k[j])}, \qquad (7)$$

where $\boldsymbol{y}[k]$ and $\boldsymbol{z}'_k[k]$ denotes the k-th elements of $\boldsymbol{y}$ and $\boldsymbol{z}'_k$, respectively. We use the gradients of $\mathcal{L}_{sce}$ to update the model including the backbone.

Therefore, our overall objective function for reactivating the BCE model is as follows:

$$\mathcal{L}_{ReCAM} = \mathcal{L}_{bce} + \lambda \mathcal{L}_{sce}, \qquad (8)$$

where λ is to balance between BCE and SCE. Please note the re-optimization of FC1 with $\mathcal{L}_{bce}$ is also included because we need to use FC1 to produce updated soft masks M_k during the learning.

Extracting ReCAM. After the reactivation, we feed the image $\boldsymbol{x}$ into it to extract its ReCAM of each class k as follows,

$$\text{ReCAM}_k(\boldsymbol{x}) = \frac{\text{ReLU}(\boldsymbol{A}_k)}{\max(\text{ReLU}(\boldsymbol{A}_k))}, \boldsymbol{A}_k = \mathbf{w}''^{\top}_k f(\boldsymbol{x}), \qquad (9)$$

where $\mathbf{w}''_k$ denotes the classification weights corresponding to the k-th class. As we have two FC layers, our implementation takes $\mathbf{w}''$ optional as: 1) $\mathbf{w}$, 2) $\mathbf{w}'$, 3)$\mathbf{w} \oplus \mathbf{w}'$, or 4)$\mathbf{w} \otimes \mathbf{w}'$, where $\oplus$ and $\otimes$ are element-wise addition and multiplication, respectively. We show the performances of these options in Section 5.2.

Refining ReCAM (Optional). As introduced in Section 3, there are a few options to refine ReCAM: **1) AdvCAM** [23] iteratively refines ReCAM by perturbing images $\boldsymbol{x}$ through adversarial climbing:

$$
\begin{aligned}
\boldsymbol{x}^t &= \boldsymbol{x}^{t-1} + \xi \nabla_{\boldsymbol{x}^{t-1}} \mathcal{L}_{adv}, \\
\mathcal{L}_{adv} &= \boldsymbol{y}^{t-1}[k] - \sum_{j \in K \backslash k} \boldsymbol{y}^{t-1}[j] \\
&\quad - \mu \left\| \mathcal{M} \otimes \left| \text{ReCAM}_k\left(\boldsymbol{x}^{t-1}\right) - \text{ReCAM}_k\left(\boldsymbol{x}^0\right)\right| \right\|_1,
\end{aligned}
\qquad (10)
$$

where $t \in [1, T]$ is the adversarial step index, $\boldsymbol{x}^t$ is the manipulated image at the t-th step. k and j are the positive and negative classes, respectively. ξ and μ are hyper-parameters (same as in [23]). $\mathcal{M} = \mathbb{1}\left(\text{ReCAM}_k\left(\boldsymbol{x}^{t-1}\right) > 0.5\right)$ is a restricting mask of ReCAM for regularization. The final

refined activation map $M'_k = \frac{\sum_{t=0}^{T} \text{ReCAM}_k\left(\boldsymbol{x}^t\right)}{\max \sum_{t=0}^{T} \text{ReCAM}_k\left(\boldsymbol{x}^t\right)}$, note that here we follow AdvCAM [23] to use ReCAM without max normalization in Eq. (9). **2) IRN** [1] takes ReCAM as the input and trains an inter-pixel relation network (IRNet) to estimate the class boundary maps $\mathcal{B}$. Here, we omit the training details of IRNet for brevity. Then, it applies a random walk to refine ReCAM with $\mathcal{B}$ and the transition probability matrix $\mathbf{T}$:

$$\text{vec}\left(\boldsymbol{M}'_k\right) = \mathbf{T}^t \cdot \text{vec}\left(\text{ReCAM}_k(\boldsymbol{x}) \otimes (1 - \mathcal{B})\right), \quad (11)$$

where t denotes the number of iterations and $\text{vec}(\cdot)$ represents vectorization. Finally, we use $\{M'_k\}$ as pixel-level labels of the image, where k denotes every positive class in the image, to train semantic segmentation models.

4.2. Justification: BCE *vs* CE

In this section, we justify the advantages of introducing SCE loss in ReCAM. We compare the effects of SCE and BCE on optimizing the classification model, theoretically and empirically.

For any input image, let $\boldsymbol{z}$ denote the prediction logits and $\boldsymbol{y}$ as the one-hot label. Based on the derivation chain rule, the gradients of BCE and SCE[2] losses on logits can be derived as:

$$
\begin{aligned}
\nabla_{\boldsymbol{z}} \mathcal{L}_{bce} &= \frac{\sigma_{sig}(\boldsymbol{z}) - \boldsymbol{y}}{K}, \\
\nabla_{\boldsymbol{z}} \mathcal{L}_{sce} &= \sigma_{sof}(\boldsymbol{z}) - \boldsymbol{y},
\end{aligned}
\qquad (12)
$$

where σ_{sig} and σ_{sof} represent sigmoid and softmax functions, respectively.

Theoretically. For the ease of analysis, we consider the binary-class ($K = 2$) situation with the positive class p and negative class q. Eq. (12) can be further derived as:

$$
\begin{array}{ll}
① \ \nabla_{z_p} \mathcal{L}_{bce} = \dfrac{-1}{2 + 2e^{z_p}} & ② \ \nabla_{z_q} \mathcal{L}_{bce} = \dfrac{1}{2 + 2e^{-z_q}} \\[2mm]
③ \ \nabla_{z_p} \mathcal{L}_{sce} = \dfrac{-1}{1 + e^{z_p - z_q}} & ④ \ \nabla_{z_q} \mathcal{L}_{sce} = \dfrac{1}{1 + e^{z_p - z_q}}
\end{array}
\qquad (13)
$$

Then, we consider different situations of z_p and z_q to compare the *magnitude* of gradient terms for both positive class p (① and ③) and negative class q (② and ④). a) $z_p \ll z_q$: the negative class logit is much larger than that of positive class. This case is quite rare and most are due to the false labeling. In this case, $\|①\|$ and $\|②\|$ are less than 0.5, but $\|③\|$ and $\|④\|$ approach 1—SCE converges faster. b) $z_p \gg z_q$. This appears when model is converging. All the four gradient terms are close to 0, which cannot tell any difference.

Next, we consider the last and most confusing case: c) $z_p \approx z_q$. We split it into two subcases: c1) both z_p and z_q are large, e.g., around 10 (as we observed in the MS COCO "5 hoofed" experiments). We can find that the magnitude of

---

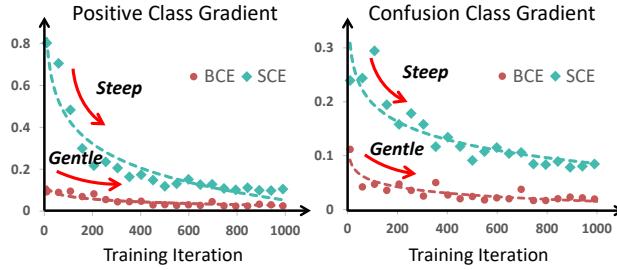[2] The SCE loss here is the vanilla SCE rather than Eq. (7).

Figure 4. The gradients with respect to the logits of the target class (i.e., the only positive class p) and the confusing class (i.e., the negative class q with the highest logit value). Both BCE and SCE models are trained with the 5 hoofed animal classes in MS COCO `train` set. These gradients are calculated on the `val` set.

the SCE loss gradients (i.e., $\|③\|$ and $\|④\|$) both approach 0.5, while $\|①\| \approx 0$ and $\|②\| \approx 0.5$. c2) z_p and z_q are small, e.g., around -10. $\|③\|$ and $\|④\|$ keep the same (as 0.5), yet $\|①\| \approx 0.5$ and $\|②\| \approx 0$. We can find that, in both confusing cases, SCE loss yields gradients to encourage the prediction of positive class as well as to penalize the prediction of negative class. The reason is the exponential terms in the denominator of the softmax function explicitly involve both classes. Based on this, SCE guarantees a class-exclusion learning—simultaneously improve the positive and suppress the negative, when confronting confusion. By contrast, in BCE each case focuses on either positive or negative class. It does not guarantee no reduction on the positive when penalizing the negative, or no promotion on the negative when encouraging the positive, leading to inefficient learning especially for confusing classes.

Empirically. One may argue that larger magnitude of gradients might not directly lead to stronger optimization, because common optimizers (e.g., Adam [18]) use adaptive learning rates. To justify the effectiveness of SCE in practice, we monitor the gradients when running real models. In specific, we review the toy experiments of "5 hoofed animals" where the models are trained with the Adam optimizer. We compute the gradients of both BCE and SCE losses (produced via two independent models) with respect to each prediction logit. As shown in Figure 4, we show the gradients with respect to the logits of the target class (i.e., the only positive class p) and the confusing class (i.e., the negative class q with the highest logit value), respectively. We can see that the gradients of SCE loss change more rapidly for both positive and negative classes, indicating that its model learns more actively and efficiently.

5. Experiments

5.1. Datasets and Settings

Datasets include the commonly used PASCAL VOC 2012 [9] and MS COCO 2014 [30]. VOC contains 20 foreground object classes and 1 `background` class. It has

$1,464$, $1,449$, and $1,456$ samples in `train`, `val`, and `test` sets, respectively. Following related works [1,23,45], we used the enlarged training set with $10,582$ training images provided by Hariharen et al. [11]. MS COCO contains 80 object classes and 1 `background` class. It has $80k$ and $40k$ samples, respectively, in its *train* and *val* sets. On both datasets, we used their image-level labels only during the training—the most challenging setting in WSSS.

Evaluation Metrics. We have mainly two evaluation steps. *Mask Generation.* We generate pseudo masks for the images in the `train` set and use their corresponding ground truth masks to compute the mIoU. *Semantic Segmentation.* We train the segmentation model, use it to predict masks for the images in `val` or `test` sets, and compute the mIoU based on their ground truth masks. We also provide the results of F1 and pixel accuracy in the supplementary.

Network Architectures. For mask generation, we follow [1, 23, 45] to use ResNet-50 as backbone and its produced feature map size is $32 \times 32 \times 2048$. For semantic segmentation, we employ ResNet-101 (following [1, 23, 45]) and Swin Transformer [31] (the first time in WSSS). Both are pre-trained on ImageNet [8]. We incorporated ResNet-101 into DeepLabV2 [5] and DeepLabV3+ [6], where the results for the latter is in the supplementary due to space limits. We incorporated Swin into UperNet [41].

Implementation Details. For mask generation, we train FC Layer-1 with the same setting as in [1]. We train FC layer-2 by: setting λ as 1 and 0.1 on VOC and MS COCO, respectively; running 4 epochs with the initial learning rate $5e^{-4}$ and polynomial learning rate decay on both datasets. We follow IRN [1] to apply the same data augmentation and weight decay strategies. All hyper-parameters in Eq. (10) and Eq. (11) follow original AdvCAM [23] and IRN [1] paper. For the DeepLabV2 in the step of semantic segmentation, we use the same training settings as in [1, 21, 23]. Please refer to the details in the supplementary. For the UperNet, the input image was first resized uniformly as $2,048 \times 512$ with a ratio range from 0.5 to 2.0, and then cropped to be 512×512 randomly before fed into the model. Data augmentation included horizontal flipping and color jitter. We trained the models for $40k$ and $80k$ iterations on VOC and MS COCO datasets, respectively, with a common batch size of 16. We deployed AdamW [32] solver with an initial learning rate $6e^{-5}$ and weight decay as 0.01. The learning rate is decayed by a power of 1.0 according to the polynomial decay schedule.

5.2. Results and Analyses

SCE on FC Layer-1 (FC1) or Layer-2 (FC2). One may argue that SCE is not necessary to be applied on an additional classifier FC2. We conduct the experiments of using SCE on FC1 (i.e., *w/o* FC2) and show the results in upper block of Table 1. "$\mathcal{L}_{bce}$ only" is the baseline of using

False Negative   **False Positive**   **Failure Cases**

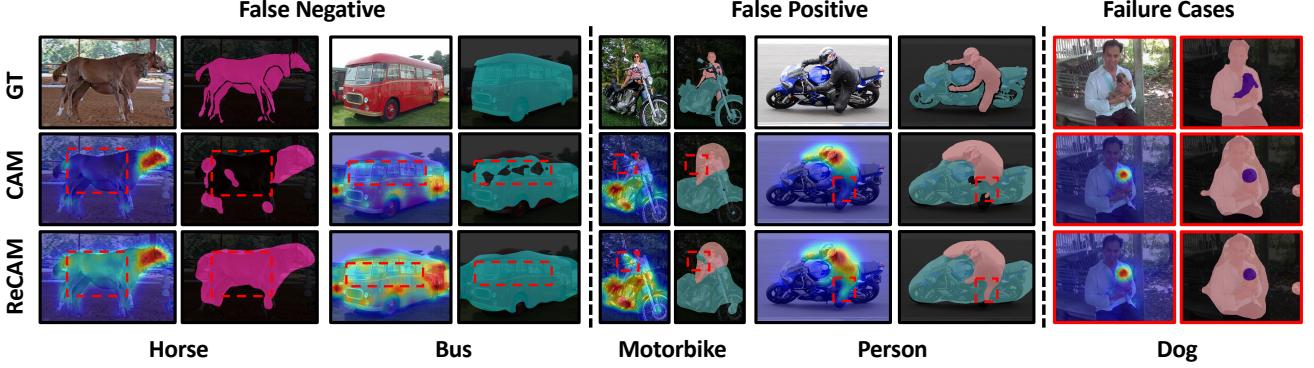Horse   Bus   Motorbike   Person   Dog

Figure 5. Visualization of 0-1 masks generated by using CAM and ReCAM on the VOC dataset (before training segmentation models). Left two blocks (each with four columns) present the two flaws introduced in Section 1: false negative pixels and false positive pixels, respectively. Red dashed boxes highlight the regions improved by ReCAM. The last block shows an example of failure case.

	Methods	VOC	MS COCO
w/o FC2	$\mathcal{L}_{bce}$ only	48.8	33.1
	$\mathcal{L}_{sce}$ only	44.6	27.9
	$\mathcal{L}_{sce}$ for single only	49.4	33.4
w/ FC2	$\mathbf{w}$ (FC1 weights)	52.1	34.6 (rp.)
	$\mathbf{w}'$ (FC2 weights)	54.1	33.2
	$\mathbf{w} \oplus \mathbf{w}'$	52.7	33.7
	$\mathbf{w} \otimes \mathbf{w}'$	54.8 (rp.)	34.0

Table 1. The upper block shows the mIoU results (%) of training a conventional multi-label classification model with different loss functions: BCE, SCE and their mixture (SCE for single-label images and BCE for multi-label images). The lower block shows the results of extracting ReCAM using different weights: the weights of FC Layer-1 or FC Layer-2 or their mixture variants (element-wise addition or multiplication). "rp." denotes the options we used to report the final results (including the mIoUs of mask refinement and semantic segmentation). **Please note** the results of using other options (e.g., $\mathbf{w}'$ used for VOC) are in the supplementary.

only BCE loss for FC1. "$\mathcal{L}_{sce}$ only" is to use SCE only for FC1, with modifying the original multi-hot labels to be normalized (summed up to 1). For example, $[1, 1, 0, 1, 0]$ is modified as $[1/3, 1/3, 0, 1/3, 0]$. "$\mathcal{L}_{sce}$ for single only" is to apply BCE for learning multi-label images but SCE for single-label images (i.e., the subset of training images containing one object class). It shows that "$\mathcal{L}_{sce}$ only" performs the worst. This is because SCE does not make sense for multi-label classification tasks where the probabilities of different classes are not independent [47]. "$\mathcal{L}_{sce}$ for single only" combines two losses to handle different images, which increases the complexity of the method. Moreover, it does not gain much, especially for MS COCO dataset where there are a smaller number of single-label images and is a more general segmentation scenario in practice.

Using the Weights of FC1 and FC2 in Eq.(9). As we have two FC layers, our implementation of $\mathbf{w}''$ has a few op-

	Methods	CAM		ReCAM (ours)	
		mIoU (%)	Time (ut)	mIoU (%)	Time (ut)
VOC	ResNet-50 [51]	48.8	1.0	54.8	1.9
	IRN [1]	66.3	8.2	<u>70.9</u>	9.1
	AdvCAM [23]	55.6	316.3	56.6	317.2
	AdvCAM + IRN	69.9	323.3	70.5	324.2
MS COCO	ResNet-50 [51]	33.1*	1.0	34.6	2.1
	IRN [1]	42.4*	8.5	44.1	9.6
	AdvCAM [23]	35.8*	302.5	37.8	303.8
	AdvCAM + IRN	45.6*	311.0	<u>46.3</u>	312.2

Table 2. Comparing ReCAM with baselines in terms of pseudo mask mIoU (%) and consumption time on VOC and MS COCO dataset. "Time" means the total computing time <u>from</u> training the model (with an ImageNet pre-trained backbone) <u>to</u> generating 0-1 masks of all training images. The unit time (ut) is 0.7 hours on VOC [9] and 5.4 hours for MS COCO [30]. * denotes results are from our re-implementation (no MS COCO results in original papers). Underline highlights our best results.

tions: 1) $\mathbf{w}$, 2) $\mathbf{w}'$, 3) $\mathbf{w} \oplus \mathbf{w}'$, or 4) $\mathbf{w} \otimes \mathbf{w}'$, where $\oplus$ and $\otimes$ are element-wise addition and multiplication, respectively. We show the results in the lower block of Table 1. We can see that all options get better results than the baseline (i.e., "$\mathcal{L}_{bce}$ only" without FC2). ReCAM with $\mathbf{w} \otimes \mathbf{w}'$ achieves best performance on VOC. The reason is that the element-wise multiplication strengthens the representative feature maps and suppresses the confusing ones. Intriguingly, on MS COCO, ReCAM with $\mathbf{w}$ achieves a better performance than $\mathbf{w} \otimes \mathbf{w}'$. This is perhaps because the feature $f_k(\boldsymbol{x})$ input to FC2 is poor in this difficult dataset and FC2 is not well-trained[3]. Based on these results, we use $\mathbf{w} \otimes \mathbf{w}'$ for all experiments on VOC, and $\mathbf{w}$ for MS COCO.

It is worth highlighting that the effectiveness of ReCAM

---

[3]The limitation of ReCAM is its FC2 may overfit to the noisy features extracted by the poor backbone. We hope to tackle this in the future by leveraging strong pre-training methods that can upgrade the backbone.

Methods	VOC								MS COCO			
	DeepLabV2				UperNet-Swin				DeepLabV2		UperNet-Swin	
	CAM		ReCAM		CAM		ReCAM		CAM	ReCAM	CAM	ReCAM
	val	test	val	test	val	test	val	test	val	val	val	val
ResNet-50 [51]	54.3	55.0	$59.0_{+4.7}$	$58.7_{+3.7}$	48.5	49.6	$54.6_{+6.1}$	$55.3_{+5.7}$	35.7	$36.5_{+0.8}$	35.9	$36.8_{+0.9}$
IRN [1]	63.5	64.8	$\underline{68.7}_{+5.2}$	$\underline{68.5}_{+3.7}$	65.9	67.7	$\underline{70.9}_{+5.0}$	$71.5_{+3.8}$	42.0	$42.9_{+0.9}$	44.0	$46.0_{+2.0}$
AdvCAM [23]	58.3	57.9	$59.1_{+0.8}$	$59.0_{+1.1}$	55.8	56.2	$57.3_{+1.5}$	$57.4_{+1.2}$	37.0	$39.4_{+2.4}$	37.8	$39.6_{+1.8}$
AdvCAM + IRN	68.1	68.0	$68.4_{+0.3}$	$68.2_{+0.2}$	70.2	70.4	$70.4_{+0.2}$	$\underline{71.7}_{+1.3}$	44.2	$\underline{45.0}_{+0.8}$	46.8	$\underline{47.9}_{+1.1}$

Table 3. The mIoU results (%) of WSSS using different segmentation models on two benchmarks. Seed masks are generated by either CAM or ReCAM, and mask refinement methods are row titles. We provide the results of DeepLabV3+ in the supplementary materials.
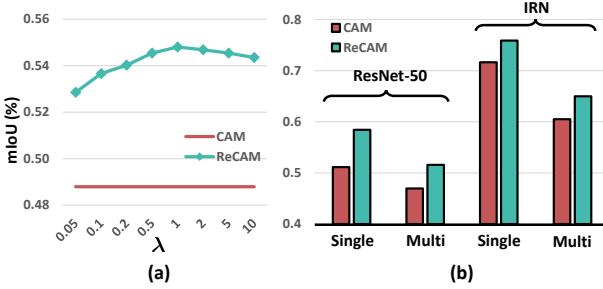

Figure 6. (a) The sensitiveness of ReCAM to the value of λ in Eq. (8), on VOC. (b) Decomposing the mIoU results on the first two lines of Table 2 into the respective results of single-label images ("Single") and multi-label ("Multi") images.

w/o saliency	val	test	w/ saliency	val	test
IRN [1]	63.5	64.8	DSRG [15]	61.4	63.2
OOA [16]	63.9	65.6	OOA* [16]	65.2	66.4
ICD [10]	64.1	64.3	SGAN [43]	66.2	66.9
SEAM [38]	64.5	65.7	ICD [10]	67.8	68.0
SC-CAM [3]	66.1	65.9	NSROM [44]	68.3	68.5
BES [4]	65.7	66.6	EDAM* [40]	70.9	70.6
CONTA [45]	65.3	66.1	EPS* [24]	70.9	70.8
AdvCAM [23]	68.1	68.0	*plugin results:*		
RIB [21]	68.3	$\underline{68.6}$	ReCAM-E*	71.6	71.4
ReCAM	$\underline{68.5}$	68.4	ReCAM-M*	$\underline{71.8}$	$\underline{72.2}$

Table 4. The mIoU results (%) using DeepLabV2 on VOC, with or without saliency detection models. On the left, the methods are with IRN (by default) if they reported such combo in their papers. On the right, we plug ReCAM respectively into EPS* (-E*) and EDAM* (-M*), or equivalently, adding their saliency encoding modules respectively into our framework, where * denotes DeepLabV2 is pre-trained on MS COCO.

is validated on both datasets, if comparing any row in the second block with the first row in the Table 1—any option of using ReCAM yields better masks than the baseline.

Effects of Different λ Values. λ in Eq. (8) controls the balancing between BCE and SCE. We study the pseudo mask quality (mIoU) of ReCAM by traversing the value of λ on VOC, as shown in Figure 6 (a). We can observe that the optimal value of λ is 1, but the difference is not significant when using other values, i.e., ReCAM is not sensitive to λ. Please kindly refer to supplementary materials for more sensitivity analysis, e.g., on learning rates.

Generality of ReCAM. *We take ReCAM as seed*, and evaluate its generality by: 1) comparing it to the vanilla CAM—the most commonly used seed generation method; and 2) applying different refinement methods after it. From the results in Table 2 and 3, we can find that ReCAM shows consistent advantages over CAM on both VOC and MS COCO. Specifically on the first row of Table 2, ReCAM itself outperforms CAM by 6% on VOC. This margin is almost maintained when using ReCAM as pseudo masks to learn semantic segmentation models, as shown in the first row of Table 3. It is worth mentioning that the margin is larger on the stronger segmentation model UperNet-Swin, e.g., 6.1% compared to the 4.7% using DeepLabV2, on VOC val.

For refining ReCAM, we have two observations: 1) the computational cost increases significantly (Table 2), e.g.,

about 4.5 times caused by IRN and 160 times by AdvCAM (over the vanilla ReCAM on ResNet-50); and 2) the best performance of WSSS is achieved always with the help of IRN, as shown in the underlined numbers in Table 3.

Figure 6 (b) shows that ReCAM generates better masks for single-label as well as multi-label images[4]. The improvements of ReCAM are maintained when adding IRN. Figure 5 shows 4 examples where ReCAM mitigates the two flaws we mentioned in Section 1: false negative pixels and false positive pixels. The rightmost block in Figure 5 shows a failure case: both CAM and ReCAM fail to capture the object parts with the occlusion or similar color to the surrounding, e.g., between "dog" and "human hands".

Superiority of ReCAM. *We may also take ReCAM as a refinement method*, and compare it with related methods such as IRN and AdvCAM. In Table 2, compared to AdvCAM (55.6%), ReCAM achieves a comparable result of 54.8% on VOC, yet it is more efficient—$160\times$ faster than AdvCAM (1.9 ut *v.s.* 316.3 ut). By cascading IRN in addition, ReCAM surpasses AdvCAM by 1% (70.9% *v.s.* 69.9%), and

---

[4]Single-label images have the main issues of false negative pixels and multi-label images have more false positive pixels due to the co-occurring classes. We provide detailed statistics in the supplementary materials.

ReCAM is more efficient (only `8.2 ut`). Besides, we can see from Table 4 that ReCAM supports plug-and-play in different CAM variants including saliency-based methods.

6. Conclusions

We started from the two common flaws of the conventional CAM. We pointed out the crux is the widely used BCE loss and demonstrated the superiority of SCE loss theoretically and empirically. We proposed a simple yet effective method named ReCAM by plugging SCE into the BCE-based model to reactivate the model. We showed its generality and superiority via extensive experiments and various case studies on two popular WSSS benchmarks.

Acknowledgments

References

[1] Jiwoon Ahn, Sunghyun Cho, and Suha Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. In *CVPR*, pages 2209–2218, 2019. 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14

[2] Jiwoon Ahn and Suha Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. In *CVPR*, pages 4981–4990, 2018. 3

[3] Yu-Ting Chang, Qiaosong Wang, Wei-Chih Hung, Robinson Piramuthu, Yi-Hsuan Tsai, and Ming-Hsuan Yang. Weakly-supervised semantic segmentation via sub-category exploration. In *CVPR*, pages 8991–9000, 2020. 3, 8

[4] Liyi Chen, Weiwei Wu, Chenchen Fu, Xiao Han, and Yuntao Zhang. Weakly supervised semantic segmentation with boundary exploration. In *ECCV*, pages 347–362, 2020. 3, 8

[5] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 40(4):834–848, 2017. 1, 2, 4, 6

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *ECCV*, pages 801–818, 2018. 1, 2, 4, 6

[7] Jifeng Dai, Kaiming He, and Jian Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, pages 1635–1643, 2015. 1

[8] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, pages 248–255, 2009. 6

[9] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 88(2):303–338, 2010. 2, 3, 6, 7

[10] Junsong Fan, Zhaoxiang Zhang, Chunfeng Song, and Tieniu Tan. Learning integral objects with intra-class discriminator for weakly-supervised semantic segmentation. In *CVPR*, pages 4283–4292, 2020. 3, 8

[11] Bharath Hariharan, Pablo Arbeláez, Lubomir Bourdev, Subhransu Maji, and Jitendra Malik. Semantic contours from inverse detectors. In *ICCV*, pages 991–998. IEEE, 2011. 6

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 4, 11

[13] Qibin Hou, Ming-Ming Cheng, Xiaowei Hu, Ali Borji, Zhuowen Tu, and Philip HS Torr. Deeply supervised salient object detection with short connections. In *CVPR*, pages 3203–3212, 2017. 3

[14] Qibin Hou, Peng-Tao Jiang, Yunchao Wei, and Ming-Ming Cheng. Self-erasing network for integral object attention. In *NeurIPS*, pages 549–559, 2018. 3

[15] Zilong Huang, Xinggang Wang, Jiasi Wang, Wenyu Liu, and Jingdong Wang. Weakly-supervised semantic segmentation network with deep seeded region growing. In *CVPR*, pages 7014–7023, 2018. 3, 8

[16] Peng-Tao Jiang, Qibin Hou, Yang Cao, Ming-Ming Cheng, Yunchao Wei, and Hong-Kai Xiong. Integral object mining via online attention accumulation. In *ICCV*, pages 2070–2079, 2019. 3, 8

[17] Beomyoung Kim, Sangeun Han, and Junmo Kim. Discriminative region suppression for weakly-supervised semantic segmentation. In *AAAI*, volume 35, pages 1754–1761, 2021. 3

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[19] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *ECCV*, pages 695–711, 2016. 1

[20] Hyeokjun Kweon, Sung-Hoon Yoon, Hyeonseong Kim, Daehee Park, and Kuk-Jin Yoon. Unlocking the potential of ordinary classifier: Class-specific adversarial erasing framework for weakly supervised semantic segmentation. In *ICCV*, pages 6994–7003, 2021. 3

[21] Jungbeom Lee, Jooyoung Choi, Jisoo Mok, and Sungroh Yoon. Reducing information bottleneck for weakly supervised semantic segmentation. In *NeurIPS*, 2021. 3, 4, 6, 8, 13

[22] Jungbeom Lee, Eunji Kim, Sungmin Lee, Jangho Lee, and Sungroh Yoon. Ficklenet: Weakly and semi-supervised semantic image segmentation using stochastic inference. In *CVPR*, pages 5267–5276, 2019. 3

[23] Jungbeom Lee, Eunji Kim, and Sungroh Yoon. Anti-adversarially manipulated attributions for weakly and semi-supervised semantic segmentation. In *CVPR*, pages 4071–4080, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 11, 13

[24] Seungho Lee, Minhyun Lee, Jongwuk Lee, and Hyunjung Shim. Railroad is not a train: Saliency as pseudo-pixel supervision for weakly supervised semantic segmentation. In *CVPR*, pages 5495–5505, 2021. 3, 8

[25] Kunpeng Li, Ziyan Wu, Kuan-Chuan Peng, Jan Ernst, and Yun Fu. Tell me where to look: Guided attention inference network. In *CVPR*, pages 9215–9223, 2018. 2

[26] Kunpeng Li, Yulun Zhang, Kai Li, Yuanyuan Li, and Yun Fu. Attention bridging network for knowledge transfer. In *ICCV*, pages 5198–5207, 2019. 3

[27] Xueyi Li, Tianfei Zhou, Jianwu Li, Yi Zhou, and Zhaoxiang Zhang. Group-wise semantic mining for weakly supervised semantic segmentation. In *AAAI*, volume 35, pages 1984–1992, 2021. 1

[28] Yi Li, Zhanghui Kuang, Liyang Liu, Yimin Chen, and Wayne Zhang. Pseudo-mask matters in weakly-supervised semantic segmentation. In *ICCV*, pages 6964–6973, 2021. 1

[29] Di Lin, Jifeng Dai, Jiaya Jia, Kaiming He, and Jian Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, pages 3159–3167, 2016. 1

[30] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pages 740–755, 2014. 1, 2, 6, 7

[31] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 4, 6

[32] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 6

[33] László Lovász. Random walks on graphs. *Combinatorics, Paul erdos is eighty*, 2(1-46):4, 1993. 3

[34] Aditya K Menon, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Multilabel reductions: what is my loss optimising? In *NeurIPS*, pages 10600–10611, 2019. 2

[35] Chunfeng Song, Yan Huang, Wanli Ouyang, and Liang Wang. Box-driven class-wise region masking and filling rate guided loss for weakly supervised semantic segmentation. In *CVPR*, pages 3136–3145, 2019. 1

[36] Paul Vernaza and Manmohan Chandraker. Learning random-walk label propagation for weakly-supervised semantic segmentation. In *CVPR*, pages 7158–7166, 2017. 1

[37] Haofan Wang, Zifan Wang, Mengnan Du, Fan Yang, Zijian Zhang, Sirui Ding, Piotr Mardziel, and Xia Hu. Score-cam: Score-weighted visual explanations for convolutional neural networks. In *CVPRW*, pages 24–25, 2020. 3

[38] Yude Wang, Jie Zhang, Meina Kan, Shiguang Shan, and Xilin Chen. Self-supervised equivariant attention mechanism for weakly supervised semantic segmentation. In *CVPR*, pages 12275–12284, 2020. 3, 8

[39] Yunchao Wei, Jiashi Feng, Xiaodan Liang, Ming-Ming Cheng, Yao Zhao, and Shuicheng Yan. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. In *CVPR*, pages 1568–1576, 2017. 3

[40] Tong Wu, Junshi Huang, Guangyu Gao, Xiaoming Wei, Xiaolin Wei, Xuan Luo, and Chi Harold Liu. Embedded discriminative attention mechanism for weakly supervised semantic segmentation. In *CVPR*, pages 16765–16774, 2021. 2, 3, 8

[41] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, pages 418–434, 2018. 1, 2, 4, 6

[42] Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, Ferdous Sohel, and Dan Xu. Leveraging auxiliary tasks with affinity learning for weakly supervised semantic segmentation. In *ICCV*, pages 6984–6993, 2021. 1, 3

[43] Qi Yao and Xiaojin Gong. Saliency guided self-attention network for weakly and semi-supervised semantic segmentation. *IEEE Access*, 8:14413–14423, 2020. 8

[44] Yazhou Yao, Tao Chen, Guo-Sen Xie, Chuanyi Zhang, Fumin Shen, Qi Wu, Zhenmin Tang, and Jian Zhang. Non-salient region object mining for weakly supervised semantic segmentation. In *CVPR*, pages 2623–2632, 2021. 3, 8

[45] Dong Zhang, Hanwang Zhang, Jinhui Tang, Xiansheng Hua, and Qianru Sun. Causal intervention for weakly-supervised semantic segmentation. In *NeurIPS*, pages 655–666, 2020. 3, 4, 6, 8

[46] Fei Zhang, Chaochen Gu, Chenyue Zhang, and Yuchao Dai. Complementary patch for weakly supervised semantic segmentation. In *ICCV*, pages 7242–7251, 2021. 1

[47] Min-Ling Zhang and Zhi-Hua Zhou. A review on multi-label learning algorithms. *TKDE*, 26(8):1819–1837, 2013. 2, 7

[48] Tianyi Zhang, Guosheng Lin, Weide Liu, Jianfei Cai, and Alex Kot. Splitting vs. merging: Mining object regions with discrepancy and intersection loss for weakly supervised semantic segmentation. In *ECCV*, pages 663–679. Springer, 2020. 3

[49] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In *CVPR*, pages 1325–1334, 2018. 3

[50] Ting Zhao and Xiangqian Wu. Pyramid feature attention network for saliency detection. In *CVPR*, pages 3085–3094, 2019. 3

[51] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016. 1, 2, 3, 4, 7, 8, 11

Supplementary materials

We present more details about the toy experiments in Section A, the quantitative results in Section B supplementing for Table 3 (main paper), more quantitative results in Section C related to Table 1 (main paper), the statistics of false positive and false negative pixels in Section D, the pseudo mask quality (mIoU) of ReCAM by traversing the value of λ on MS COCO in Section E, the sensitivity analysis to learning rates in Section F, the detailed derivation of SCE and BCE in Section G, the algorithm of ReCAM in Section H, more training details in Section I, and more qualitative results in Section J supplementing for Figure 5 (main paper).

A. More Details about Toy Experiments

The 5 hoofed-animal classes include `horse`, `sheep`, `cow`, `elephant`, and `bear` on the MS COCO. We chose the images that contain only one of these classes and ignored other MS COCO classes co-occurring in the image (e.g., the image containing a `person` and a `horse` will be selected but it is labeled with a one-hot label `horse`). There are 6,340 and 3,001 such images in MS COCO `train` and `val` set, respectively. Then, we trained the 5-class classification models on the 6,430 `train` images using BCE or SCE losses, respectively, and evaluated the models on the 3,001 `val` images (please note we also showed the class activation results (mIoU) for training images in the main paper).

B. More WSSS Results (DeepLabV3+)

Table S1 presents the mIoU (%) results of WSSS when exploiting DeepLabV3+ models. It is to supplement for Table 3 in the main paper.

Methods	VOC		MS COCO	
	CAM	ReCAM	CAM	ReCAM
ResNet-50 [51]	53.5	57.8	36.2	37.0
IRN [1]	64.2	69.3	43.1	44.3
AdvCAM [23]	57.3	58.3	37.7	40.2
AdvCAM + IRN	68.5	68.6	44.4	45.8

Table S1. The semantic segmentation results (mIoU%) of using DeepLabV3+, on VOC and MS COCO. Seed masks are generated by either CAM or ReCAM, and then fed into mask refinement algorithms (listed as row titles). The refined masks are used as pseudo labels to train the semantic segmentation model. Finally, the model is evaluated on the `val` set.

	Weights	VOC	MS COCO
ResNet-50 [12]	$\mathbf{w}$ (FC1 weights)	56.5	<u>36.5</u>
	$\mathbf{w}'$ (FC2 weights)	58.2	35.6
	$\mathbf{w} \otimes \mathbf{w}'$	<u>59.0</u>	36.1
IRN [1]	$\mathbf{w}$ (FC1 weights)	64.8	<u>42.9</u>
	$\mathbf{w}'$ (FC2 weights)	67.5	42.1
	$\mathbf{w} \otimes \mathbf{w}'$	<u>68.7</u>	42.6

Table S2. The semantic segmentation results (mIoU%) using DeepLabV2 on VOC and MS COCO. It is to show the difference of using different FC weights for computing ReCAM. The pseudo masks (used to train the models) are either ReCAM in the first block or refined masks of ReCAM with IRN in the second block.

C. Different Weights for ReCAM

Table S2 shows the mIoU results (%) of WSSS (using DeepLabV2) when applying different FC weights to extract ReCAM. We show two blocks of WSSS results: one for using ReCAM to generate seeds (and directly using seed masks to train WSSS models), and the other one with a further step of refining the seed masks using IRN (and then using refined masks to train WSSS models).

D. Statistics of Two Flaws

	VOC			MS COCO	
	CAM	ReCAM (0.21)	ReCAM (0.26)	CAM	ReCAM
TP	2.05e8 (82.5)	2.09e8 (84.3)	2.1e8 (84.9)	1.69e10 (74.1)	1.72e10 (75.3)
FP (obj)	1.86e6 (0.8)	2.08e6 (0.8)	1.80e6 (0.7)	7.94e8 (3.5)	7.45e8 (3.3)
FN	2.33e7 (9.4)	1.65e7 (6.7)	2.09e7 (8.4)	2.76e9 (12.1)	2.60e9 (11.4)
FP (bg)	2.02e7 (8.1)	2.25e7 (9.1)	1.66e7 (6.7)	3.16e9 (13.9)	3.03e9 (13.3)

Table S3. The number of pixels and the percentage (%) for different pixels in the seed masks. "TP" indicates the true positive. "FP (obj)" indicates the false positive whose actual label is another object class. "FP (bg)" is false positive whose actual label is `background`. "FN" is false negative which is misclassified as `background`. The summation of the percentages of "TP", "FN" and "FP (bg)" is 100% in each column.

In Table S3, we show the detailed statistics of two flaws analyzed in the main paper. We also provide the numbers of TP and FP (bg) as additional reference. It is worth mentioning that when thresholding CAM to be 0-1 mask, there is a trade-off between FP and FN—a higher threshold value results in less FP and more FN. *Please note that we follow AdvCAM [23] to do the fine-grained grid search for the value of the threshold.*

Table S3 presents the results of using two different thresholds for VOC (threshold=0.21 was used in the main paper). We can see that when comparing ReCAM (0.21)

with CAM, there is a significant decrease in FN pixels, but a slight increase in FP pixels (including both obj and bg). When increasing the threshold to 0.26, both FN and FP pixels are reduced. However, the overall improvements over CAM drops a bit (threshold=0.21 results in 54.8% mIoU, however, threshold=0.26 results in 53.8% mIoU). We are happy to see that ReCAM reduces both FP and FN clearly when it achieves the best performance on the more challenging dataset—MS COCO.

E. λ on MS COCO

The hyperparameter λ balances the effects of BCE and SCE terms in the loss function (see Eq. (8) in the main paper). We show the effects on the results when traversing the value of λ on MS COCO in Figure S1. This is to supplement for Figure 6 (a) of the main paper. The optimal value of λ on MS COCO is 0.1, and the model performance drops a lot when using large λ values (e.g. 2). We have explained the reason in the paragraph of Section 5.2 entitled as "Using the Weights of FC1 and FC2 in Eq.(9)".
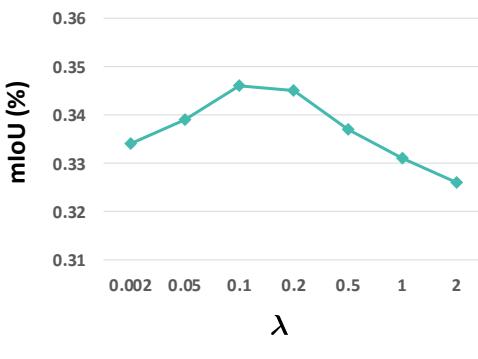
Figure S1. The sensitiveness of ReCAM to the value of λ on MS COCO.

F. Sensitivity to learning rate

We run experiments using different learning rates (LR). We realize this by applying different scalars (denoted as learning rate ratios in Figure S2) on the default LR values used in the paper. Please note that our setting of LR for baseline CAM exactly follows IRN [1]. We can see in Figure S2 that large LR values make the training of CAM unstable and end with a NaN loss. In contrast, our ReCAM is not sensitive. We think this is because the two FC layers in ReCAM are trained not from scratch but from the weights of a pre-trained baseline BCE model (that is why we call it "Reactivating CAM"). We have highlighted this reason in the paragraph of Section 5.2 entitled as "Using the Weights of FC1 and FC2 in Eq.(9)".
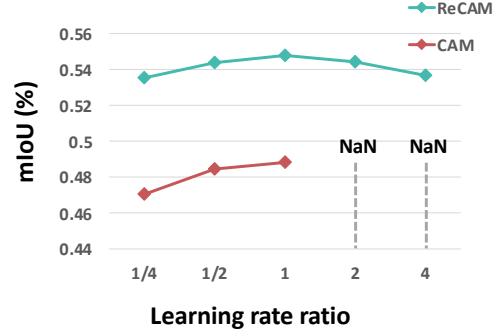
Figure S2. The mIoU results of CAM and ReCAM trained with different learning rates on VOC. Learning rate ratio denotes the scalar applied on the default learning rates used in the paper.

G. Gradients of BCE and CE

The gradients of BCE and SCE losses on logits can be derived as:

$$\nabla_{\boldsymbol{z}}\mathcal{L}_{bce} = \frac{\sigma_{sig}(\boldsymbol{z}) - \boldsymbol{y}}{K},$$
$$\nabla_{\boldsymbol{z}}\mathcal{L}_{sce} = \sigma_{sof}(\boldsymbol{z}) - \boldsymbol{y}, \tag{14}$$

here we will show how to compute them.

G.1. BCE

BCE Loss in multi-label classification:

$$\mathcal{L}_{bce} = -\frac{1}{K}\sum_{i} \boldsymbol{y}_i \log \sigma_{sig}(\boldsymbol{z}_i) + (1-\boldsymbol{y}_i)\log(1-\sigma_{sig}(\boldsymbol{z}_i)). \tag{15}$$

The derivative of sigmoid function is

$$\begin{aligned}\nabla_i \sigma_{sig}(i) &= \nabla_i \frac{1}{1+e^{-i}} \\ &= \frac{e^{-i}}{(1+e^{-i})^2} \\ &= \sigma_{sig}(i)(1-\sigma_{sig}(i)).\end{aligned} \tag{16}$$

For positive class p, $\boldsymbol{y}_p$=1, the gradient of $\mathcal{L}_{bce}$ with respect to class p is

$$\begin{aligned}\nabla_{\boldsymbol{z}_p}\mathcal{L}_{bce} &= -\frac{1}{K}\nabla_{\boldsymbol{z}_p}(\boldsymbol{y}_p \log \sigma_{sig}(\boldsymbol{z}_p)) \\ &= -\frac{1}{K}\frac{\sigma_{sig}(\boldsymbol{z}_p)(1-\sigma_{sig}(\boldsymbol{z}_p))}{\sigma_{sig}(\boldsymbol{z}_p)} \\ &= \frac{\sigma_{sig}(\boldsymbol{z}_p) - 1}{K} \\ &= \frac{\sigma_{sig}(\boldsymbol{z}_p) - \boldsymbol{y}_p}{K}.\end{aligned} \tag{17}$$

For negative class q, $\boldsymbol{y}_q=0$, the gradient of $\mathcal{L}_{bce}$ with respect to class q is

$$
\begin{aligned}
\nabla_{\boldsymbol{z}_q}\mathcal{L}_{bce} &= -\frac{1}{K}\nabla_{\boldsymbol{z}_q}((1-\boldsymbol{y}_p)\log(1-\sigma_{sig}(\boldsymbol{z}_q))) \\
&= \frac{1}{K}\frac{(1-\sigma_{sig}(\boldsymbol{z}_q))\sigma_{sig}(\boldsymbol{z}_q)}{1-\sigma_{sig}(\boldsymbol{z}_q)} \\
&= \frac{\sigma_{sig}(\boldsymbol{z}_q)}{K} \\
&= \frac{\sigma_{sig}(\boldsymbol{z}_q)-\boldsymbol{y}_q}{K}.
\end{aligned}
\tag{18}
$$

Then, the gradients of $\mathcal{L}_{bce}$ on logits can be derived as

$$
\nabla_{\boldsymbol{z}}\mathcal{L}_{bce} = \frac{\sigma_{sig}(\boldsymbol{z})-\boldsymbol{y}}{K}.
\tag{19}
$$

G.2. CE

SCE Loss in multi-label classification:

$$
\mathcal{L}_{sce} = -\sum_i \boldsymbol{y}_i \log \sigma_{sof}(\boldsymbol{z}_i).
\tag{20}
$$

For positive class p, $\boldsymbol{y}_p=1$, the gradient of $\mathcal{L}_{sce}$ with respect to class p is

$$
\begin{aligned}
\nabla_{\boldsymbol{z}_p}\mathcal{L}_{sce} &= \nabla_{\boldsymbol{z}_p}(-\log e^{\boldsymbol{z}_p} + \log \sum_i e^{\boldsymbol{z}_i}) \\
&= -1 + \frac{e^{\boldsymbol{z}_p}}{\sum_i e^{\boldsymbol{z}_i}} \\
&= \sigma_{sof}(\boldsymbol{z}_p)-\boldsymbol{y_p}.
\end{aligned}
\tag{21}
$$

For negative class q, $\boldsymbol{y}_q=0$, the gradient of $\mathcal{L}_{sce}$ with respect to class q is

$$
\begin{aligned}
\nabla_{\boldsymbol{z}_q}\mathcal{L}_{sce} &= \nabla_{\boldsymbol{z}_q}(-\log e^{\boldsymbol{z}_p} + \log \sum_i e^{\boldsymbol{z}_i}) \\
&= \frac{e^{\boldsymbol{z}_q}}{\sum_i e^{\boldsymbol{z}_i}} \\
&= \sigma_{sof}(\boldsymbol{z}_q)-\boldsymbol{y}_q.
\end{aligned}
\tag{22}
$$

Then, the gradients of $\mathcal{L}_{sce}$ on logits can be derived as

$$
\nabla_{\boldsymbol{z}}\mathcal{L}_{sce} = \sigma_{sof}(\boldsymbol{z})-\boldsymbol{y},
\tag{23}
$$

H. Algorithm

The training pipeline of ReCAM is in Algorithm 1.

I. Training Details of DeepLabV2

We supplement the details in the training process of DeepLabV2. Following [21, 23], we cropped each training image to the size of 321×321. We trained the model for 20k and 100k iterations on VOC and MS COCO datasets, respectively, with the respective batch size of 5 and 10. The learning rate was set as 2.5e-4 and weight decay as 5e-4. Horizontal flipping and random crop were used for data augmentation.

Algorithm 1 ReCAM

Input: Training images $\mathcal{X}$ and image-level labels $\mathcal{Y}$
Output: ReCAM (soft masks before thresholding)
1: Load ImageNet pre-trained feature extractor Θ
2: Randomly initialize weights $\mathbf{w}$ of FC Layer-1
3: **for** Image batches in $\mathcal{X}$ **do**
4:     Optimize Θ and $\mathbf{w}$ using Eq. (1)
5: **end for**
6: Randomly initialize weights $\mathbf{w}$' of FC Layer-2
7: **for** Image batches in $\mathcal{X}$ **do**
8:     Optimize Θ, $\mathbf{w}$, and $\mathbf{w}'$ using Eq. (8)
9: **end for**
10: Extract ReCAM using Eq. (9)

J. More Qualitative Results

Figure S3 shows more qualitative results of heatmaps and 0-1 masks generated by CAM and ReCAM, on VOC `train` set. This is to supplement for Figure 5 in the main paper. Figure S4 shows the refined masks using IRN [1] on VOC and MS COCO `train` set. Figure S5 shows the resulted masks of semantic segmentation (using DeepLabV2) on VOC and MS COCO `val` set.
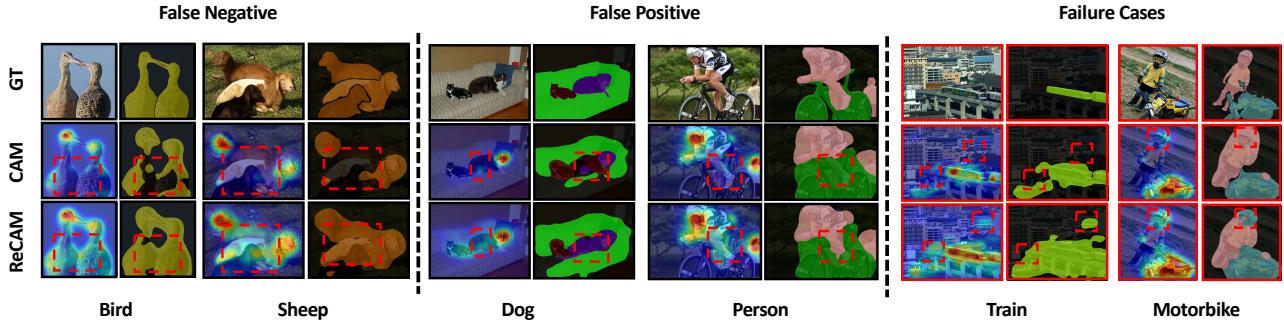
Figure S3. Visualization of the soft masks and 0-1 masks, generated by CAM and ReCAM on the VOC dataset.
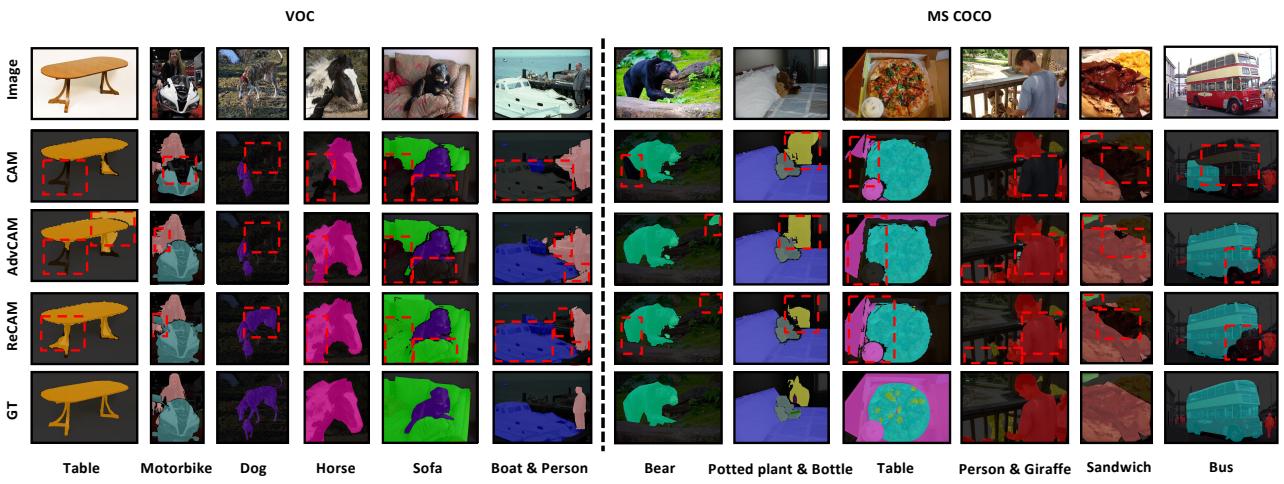

Figure S4. Visualization of the masks (from different CAM variants) refined by IRN [1] on the VOC and MS COCO datasets.
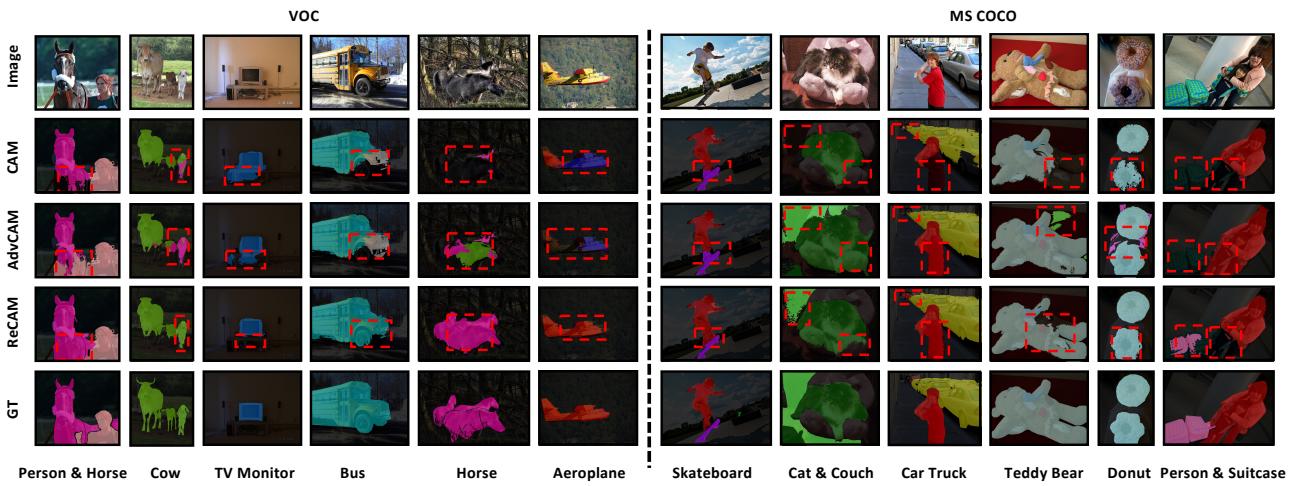

Figure S5. Visualization of the semantic segmentation results (i.e., predicted masks) using DeepLabV2 on the VOC and MS COCO datasets. The seeds (of pseudo masks) are generated by different methods first and then are all refined by IRN.