Extracting Class Activation Maps from Non-Discriminative Features as well

Zhaozheng Chen
Singapore Management University
zzchen.2019@phdcs.smu.edu.sg

Qianru Sun
Singapore Management University
qianrusun@smu.edu.sg

Abstract

Extracting class activation maps (CAM) from a classification model often results in poor coverage on foreground objects, i.e., only the discriminative region (e.g., the "head" of "sheep") is recognized and the rest (e.g., the "leg" of "sheep") mistakenly as background. The crux behind is that the weight of the classifier (used to compute CAM) captures only the discriminative features of objects. We tackle this by introducing a new computation method for CAM that explicitly captures non-discriminative features as well, thereby expanding CAM to cover whole objects. Specifically, we omit the last pooling layer of the classification model, and perform clustering on all local features of an object class, where "local" means "at a spatial pixel position". We call the resultant K cluster centers local prototypes *— represent local semantics like the "head", "leg", and "body" of "sheep". Given a new image of the class, we compare its unpooled features to every prototype, derive K similarity matrices, and then aggregate them into a heatmap (i.e., our CAM). Our CAM thus captures all local features of the class without discrimination. We evaluate it in the challenging tasks of weakly-supervised semantic segmentation (WSSS), and plug it in multiple state-of-the-art WSSS methods, such as MCTformer [45] and AMN [26], by simply replacing their original CAM with ours. Our extensive experiments on standard WSSS benchmarks (PASCAL VOC and MS COCO) show the superiority of our method: consistent improvements with little computational overhead. Our code is provided at https://github.com/zhaozhengChen/LPCAM.*

1. Introduction

Extracting CAM [50] from classification models is the essential step for training semantic segmentation models when only image-level labels are available, i.e., in the WSSS tasks [8, 22, 24, 26, 46]. More specifically, the general pipeline of WSSS consists of three steps: 1) training a multi-label classification model with the image-level labels; 2) extracting CAM of each class to generate a 0-1 mask (usually called seed mask), often with a further step of refinement to generate pseudo mask [1, 20]; and 3) taking all-class pseudo masks as pseudo labels to train a semantic segmentation model in a fully-supervised fashion [5, 6, 41]. It is clear that the CAM in the first step determines the performance of the final semantic segmentation model. However, the conventional CAM and its variants often suffer from the poor coverage of foreground objects in the image, i.e., a large amount of object pixels are mistakenly recognized as background, as demonstrated in Figure 1(a) where only few pixels are activated in warm colors.

We point out that this locality is due to the fact that CAM is extracted from a discriminative model. The training of such model naturally discards the non-discriminative regions which confuse the model between similar as well as highly co-occurring object classes. This is a general problem of discriminative models, and is particularly obvious when the number of training classes is small [37, 46, 48]. To visualize the evidence, we use the classifier weights of confusing classes, e.g., "car", "train" and "person", to compute CAMs for the "bus" image, and show them in Figure 1(b). We find from (a) and (b) that the heating regions in ground truth class and confusing classes are complementary. E.g., the upper and frontal regions (on the "bus" image) respectively heated in the CAMs of "car" and "train" (confusing classes) are missing in the CAM of "bus" (ground truth), which means the classifier de-activates those regions for "bus" as it is likely to recognize them as "car" or "train".

Technically, for each class, we use two factors to compute CAM: 1) the feature map block after the last conv layer, and 2) the weight of the classifier for that class. As aforementioned, the second factor is often biased to discriminative features. Our intuition is to replace it with a non-biased one. The question becomes how to derive a *non-biased* classifier from a biased classification model, where *non-biased* means representing all local semantics of the class. We find the biased classifier is due to incomplete features, i.e., only discriminative local features fed into the classifier, and the non-discriminative ones are kicked out by the global average pooling (GAP) after the last conv layer. To let the model pay attention to non-discriminative features as well, we pro-

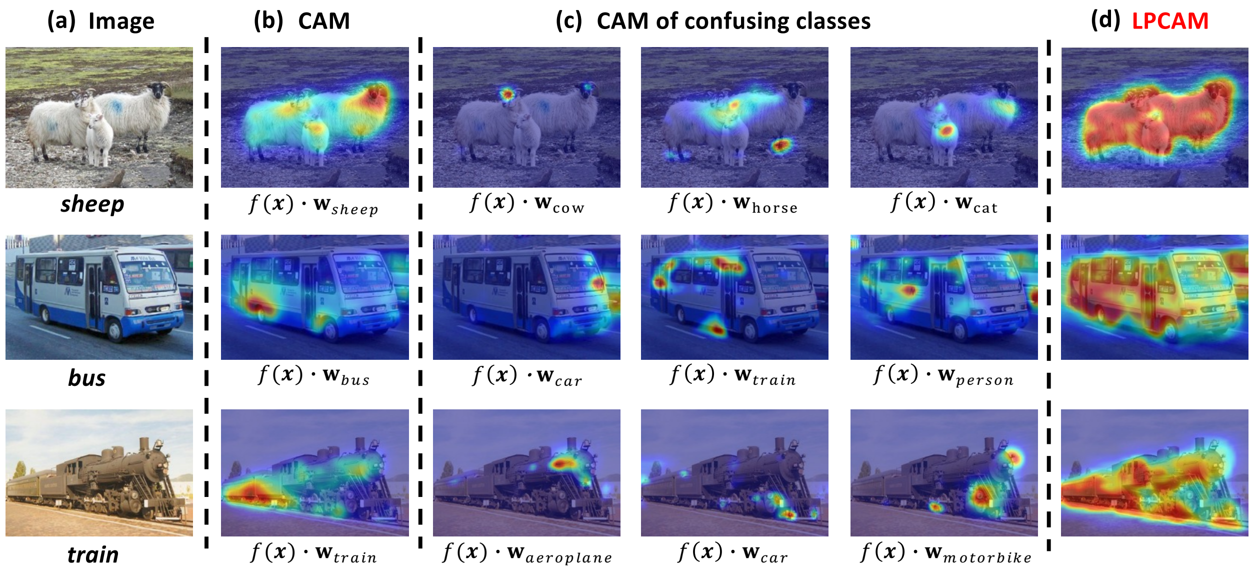

(a) Image
(b) CAM
(c) CAM of confusing classes
(d) LPCAM
sheep
$f(\boldsymbol{x}) \cdot \mathbf{w}_{sheep}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{\text{cow}}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{\text{horse}}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{\text{cat}}$
bus
$f(\boldsymbol{x}) \cdot \mathbf{w}_{bus}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{car}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{train}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{person}$
train
$f(\boldsymbol{x}) \cdot \mathbf{w}_{train}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{aeroplane}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{car}$
$f(\boldsymbol{x}) \cdot \mathbf{w}_{motorbike}$

Figure 1. The CAM of image $\boldsymbol{x}$ is computed by $f(\boldsymbol{x}) \cdot \mathbf{w}_c$, where $f(\boldsymbol{x})$ is the feature map block (before the last pooling layer of the multi-label classification model) and $\mathbf{w}_c$ denotes the classifier weights of class c. (a) Input images. (b) CAMs generated from the classifier weights of ground truth class. (c) CAMs generated from the classifier weights of confusing classes. (d) LPCAM (our method).

pose to omit GAP, derive a prototype-based classifier by clustering all local features (collected across all spatial locations on the feature map blocks of all training samples in the class) into K local prototypes each representing a local semantic of the class. In Section 3, we give a detailed justification that this prototype-based classifier is able to capture both discriminative and non-discriminative features.

Then, the question is how to use local prototypes on the feature map block (i.e., the 1st factor) to generate CAM. We propose to apply them one-by-one on the feature map block to generate K similarity maps (e.g., by using cosine distance), each aiming to capture the local regions that contain similar semantics to one of the prototypes. We highlight that this "one-by-one" is important to preserve non-discriminative regions as the normalization on each similarity map is independent. We provide a detailed justification from the perspective of normalization in Section 3.2. Then, we average across all normalized similarity maps to get a single map—we call Local Prototype CAM (LPCAM). In addition, we extend LPCAM by using the local prototypes of contexts as well. We subtract the context similarity maps (computed between the feature map block and the clustered context prototypes) from LPCAM. The idea is to remove the false positive pixels (e.g., the "rail" of "train" [25]). Therefore, our LPCAM not only captures the missing local features of the object but also mitigates the spurious features caused by confusing contexts.

LPCAM is a new operation to compute class activation maps based on clustered local prototypes. In principle, it can be taken as a generic substitute of the conventional CAM in CAM-based WSSS methods. To evaluate LPCAM on different WSSS methods (as they use different backbones, pre-training strategies or extra data), we conduct extensive experiments by plugging it in multiple methods: the popular refinement method IRN [1], the top-performing AMN [26], the saliency-map-based EDAM [39], and the transformer-arch-based MCTformer [45], on two popular benchmarks of semantic segmentation, PASCAL VOC 2012 [11] and MS COCO 2014 [30].

Our Contributions in this paper are thus two-fold. 1) A novel method LPCAM that leverages non-discriminative local features and context features (in addition to discriminative ones) to generate class activation maps with better coverage on the complete object. 2) Extensive evaluations of LPCAM by plugging it in multiple WSSS methods, on two popular WSSS benchmarks.

2. Related Works

Image classification models are optimized to capture only the discriminative local regions (features) of objects, leading to the poor coverage of its CAM on the objects. While the ideal pseudo labels (extended from CAM) to train WSSS models should cover all parts of the object regardless of dicriminativeness. To this end, many efforts have been made in the field of WSSS. Below, we introduce only the variants for seed generation and mask refinement.

Seed Generation One direction is the erasing-based methods. AE-PSL [37] is an adversarial erasing strategy running in an iterative manner. It masks out the discriminative regions in the current iteration, to explicitly force the model to discover new regions in the next iteration. ACoL [48] is an improved method. It is based on an end-to-end learning framework composed of two branches. One branch applies a feature-level masking on the other. However, these two methods have an over-erasing problem, especially for small objects. CSE [21] is a class-specific erasing method. It masks out a random object class based on CAM and then explicitly penalizes the prediction of the erased class. In this way, it gradually approaches to the boundary of the object on the image. It can not only discover more non-discriminative regions but also alleviate the over-erasing problem (of the above two methods) since it also penalizes over-erased regions. However, erasing-based methods have the low-efficiency problem as they have to feed forward an image multiple times.

Besides of erasing, there are other advanced methods recently. RIB [22] interpreted the poor coverage problem of CAM in an information bottleneck principle. It re-trains the multi-label classifier by omitting the activation function in the last layer to encourage the information transmission of information non-discriminative regions to the classification. Jiang et al. [18] empirically observed that classification models can discover more discriminative regions when taking local image patches rather than whole input image as input. They proposed a local-to-global attention transfer method contains a local network that produces local attentions with rich object details for local views as well as a global network that receives the global view as input and aims to distill the discriminative attention knowledge from the local network. Some other researchers explore to utilizing contrastive learning [10,51], graph neural network [29], and self-supervised learning [3, 36] to discover more non-discriminative regions.

Compared to aforementioned methods, our method have the following advantages: 1) it does not require additional training on the basis of CAM; and 2) it can be taken as a generic substitute of the conventional CAM and plugged into many CAM-based WSSS frameworks.

Mask Refinement One category of refinement methods [1, 2, 4, 44] 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 [16, 17, 23, 27, 39] utilize saliency maps [15, 49] to extract background cues and combine them with object cues. EPS [27] proposed a joint training strategy to combine CAM and saliency maps. EDAM [39] introduced a post-processing method to integrate the confident areas in the saliency map into CAM. Our LPCAM is orthogonal to them and can be plugged into those methods.

3. Method

In Section 3.1, we introduce the pipeline of generating LPCAM using a collection of class-wise prototypes including class prototypes and context prototypes, without any retraining on the classification model. The step-by-step illustration is shown in Figure 2, demonstrating the steps of generating local prototypes from all images of a class and using these prototypes to extract LPCAM for each single image. In Section 3.2, we justify 1) the advantages of using clustered local prototypes in LPCAM; and 2) the effectiveness of LPCAM from the perspective of map normalization.

3.1. LPCAM Pipeline

Backbone and Features. We use a standard ResNet-50 [14] as the network backbone (i.e., feature encoder) of the multi-label classification model to extract features, following related works [1, 8, 22, 24, 26, 46]. Given an input image $\boldsymbol{x}$, and its multi-hot class label $\boldsymbol{y} \in \{0,1\}^N$, we denote the output of the trained feature encoder as $f(\boldsymbol{x}) \in \mathbb{R}^{W\times H\times C}$. C denotes the number of channels, H and W are the height and width, respectively, and N is the total number of foreground classes in the dataset.

Extracting CAM. Before clustering local prototypes of class as well as context, we need the rough location information of foreground and background. We use the conventional CAM to achieve this. We extract it for each individual class n given the feature $f(\boldsymbol{x})$ and the corresponding classifier weights $\mathbf{w}_n$ in the FC layer, as follows,

$$\mathrm{CAM}_n(\boldsymbol{x}) = \frac{\mathrm{ReLU}\left(\boldsymbol{A}_n\right)}{\max\left(\mathrm{ReLU}\left(\boldsymbol{A}_n\right)\right)}, \quad \boldsymbol{A}_n = \mathbf{w}_n^\top f(\boldsymbol{x}). \tag{1}$$

Clustering. We perform clustering for every individual class. Here we discuss the details for class n. Given an image sample $\boldsymbol{x}$ of class n, we divide the feature block $f(\boldsymbol{x})$ spatially into two sets, $\mathcal{F}$ and $\mathcal{B}$, based on CAM:

$$f(\boldsymbol{x})^{i,j} \in \begin{cases} \mathcal{F}, & \text{if } \mathrm{CAM}_n^{i,j}(\boldsymbol{x}) \geq \tau \\ \mathcal{B}, & \text{otherwise} \end{cases} \tag{2}$$

where $f(\boldsymbol{x})^{i,j} \in \mathbb{R}^C$ denotes the local feature at spatial location (i, j). τ is the threshold to generate a 0-1 mask from $\mathrm{CAM}_n(\boldsymbol{x})$. $\mathcal{F}$ denotes the set of foreground local features, and $\mathcal{B}$ for the set of background (context) local features.

Similarly, we can collect $\mathcal{F}$ to contain the foreground features of all samples[1] in class n, and $\mathcal{B}$ for all background

[1]We use a random subset of samples for each class in the real implementation, to reduce the computation costs of clustering.

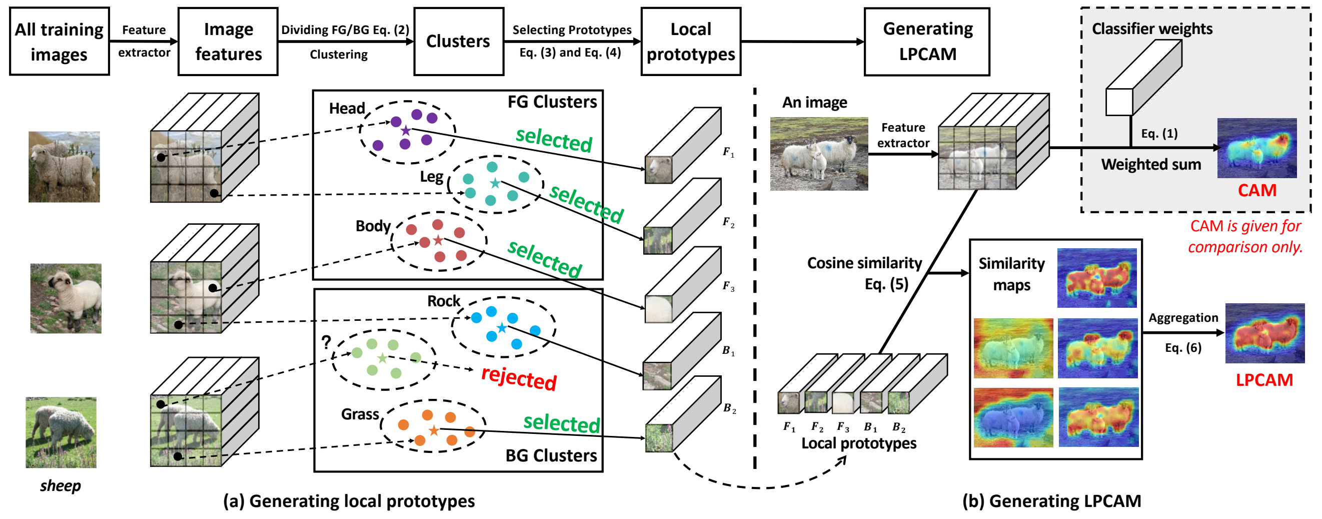


Figure 2. The pipeline for generating LPCAM. (a) Generating local prototypes from all training images of an individual class, e.g., "sheep". (b) Generating LPCAM by *sliding* both class prototypes and context prototypes over the feature block of the input image, and *aggregating* the obtained similarity maps. In the gray block, the generation process of CAM is shown for comparison.

features, where the subscript n is omitted for brevity. After that, we perform K-Means clustering, respectively, for $\mathcal{F}$ and $\mathcal{B}$, to obtain K class centers in each of them, where K is a hyperparameter. We denote the foreground cluster centers as $\mathbf{F} = \{\mathbf{F}_1, \cdots, \mathbf{F}_K\}$ and the background cluster centers as $\mathbf{B} = \{\mathbf{B}_1, \cdots, \mathbf{B}_K\}$.

Selecting Prototypes. The masks of conventional CAM are not accurate or complete, e.g., background features could be grouped into $\mathcal{F}$. To solve this issue, we need an "evaluator" to check the eligibility of cluster centers to be used as prototypes. The intuitive way is to use the classifier $\mathbf{w}_n$ as an auto "evaluator": using it to compute the prediction score of each cluster center $\mathbf{F}_i$ in $\mathbf{F}$ by:

$$\boldsymbol{z}_i = \frac{\exp(\mathbf{F}_i \cdot \mathbf{w}_n)}{\sum_j \exp(\mathbf{F}_i \cdot \mathbf{w}_j)}. \tag{3}$$

Then, we select those centers with high confidence: $\boldsymbol{z}_i > \mu_f$, where μ_f is a threshold—usually a very high value like 0.9. We denote selected ones as $\mathbf{F}' = \{\mathbf{F}'_1, \cdots, \mathbf{F}'_{K'_1}\}$. Intuitively, confident predictions indicate strong local features, i.e., prototypes, of the class.

Before using these local prototypes to generate LPCAM, we highlight that in our implementation of LPCAM, we not only *preserve* the non-discriminative features but also *suppress* strong context features (i.e., false positive), as the extraction and application of context prototypes are convenient—similar to class prototypes but in a reversed manner. We elaborate these in the following. For each $\mathbf{B}_i$ in the context cluster center set $\mathbf{B}$, we apply the same method (as for $\mathbf{F}_i$) to compute a prediction score:

$$\boldsymbol{z}_i = \frac{\exp(\mathbf{B}_i \cdot \mathbf{w}_n)}{\sum_j \exp(\mathbf{B}_i \cdot \mathbf{w}_j)}. \tag{4}$$

Intuitively, if the model is well-trained on class labels, its prediction on context features should be extremely low. Therefore, we select the centers with $\boldsymbol{z}_i < \mu_b$ (where μ_b is usually a value like 0.5), and denote them as $\mathbf{B}' = \{\mathbf{B}'_1, \cdots, \mathbf{B}'_{K'_2}\}$. It is worth noting that our method is not sensitive to the values of the hyperparameters μ_f and μ_b, given reasonable ranges, e.g., μ_f should have a large value around 0.9. We show an empirical validation for this in Section 4.

Generating LPCAM. Each of the prototypes represents a local visual pattern in the class: $\mathbf{F}'_i$ for class-related pattern (e.g., the "leg" of "sheep" class) and $\mathbf{B}'_i$ for context-related pattern (e.g., the "grassland" in "sheep" images) where the context often correlates with the class. Here we introduce how to apply these prototypes on the feature map block to generate LPCAM. LPCAM can be taken as a substitute of CAM. In Subsection 3.2, we will justify why LPCAM is superior to CAM from two perspectives: Global Average Pooling (GAP) and Normalization.

For each prototype, we slide it over all spatial positions on the feature map block, and compute its similarity to the local feature at each position. We adopt cosine similarity as we used it for K-Means. In the end, we get a cosine similarity map between prototype and feature. After computing all similarity maps (by sliding all local prototypes), we aggregate them as follows,

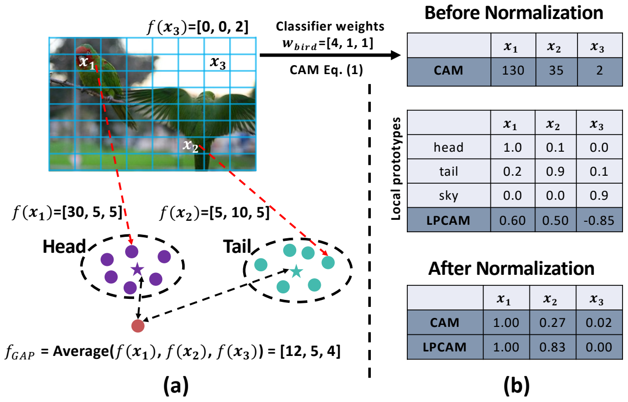
$$\begin{aligned} \boldsymbol{FG}_n &= \frac{1}{K'_1} \sum_{\mathbf{F}'_\mathbf{i} \in \mathbf{F}'} sim(f(\boldsymbol{x}), \mathbf{F}'_\mathbf{i}), \\ \boldsymbol{BG}_n &= \frac{1}{K'_2} \sum_{\mathbf{B}'_\mathbf{i} \in \mathbf{B}'} sim(f(\boldsymbol{x}), \mathbf{B}'_\mathbf{i}), \end{aligned} \tag{5}$$

where $sim()$ denotes cosine similarity. As $sim()$ value is always within the range of $[-1, 1]$, each pixel on the maps of $\boldsymbol{FG}_n$ and $\boldsymbol{BG}_n$ has a normalized value, i.e., $\boldsymbol{FG}_n$ and $\boldsymbol{BG}_n$ are normalized. Intuitively, $\boldsymbol{FG}_n$ highlights the class regions in the input image correlated to the n-th prototype, while $\boldsymbol{BG}_n$ highlights the context regions. The former needs to be preserved and the latter (e.g., pixels highly correlated to backgrounds) should to be removed. Therefore, we can formulate LPCAM as follows:

$$\begin{aligned} &\mathrm{LPCAM}_n(\boldsymbol{x}) = \frac{\mathrm{ReLU}\left(\boldsymbol{A}_n\right)}{\max\left(\mathrm{ReLU}\left(\boldsymbol{A}_n\right)\right)}, \\ &\boldsymbol{A}_n = \boldsymbol{FG}_n - \boldsymbol{BG}_n, \end{aligned} \tag{6}$$

where the first formula is the application of the maximum-value based normalization (the same as in CAM).

3.2. Justifications



	x_1	x_2	x_3
CAM	130	35	2

	x_1	x_2	x_3
head	1.0	0.1	0.0
tail	0.2	0.9	0.1
sky	0.0	0.0	0.9
LPCAM	0.60	0.50	-0.85

	x_1	x_2	x_3
CAM	1.00	0.27	0.02
LPCAM	1.00	0.83	0.00

Figure 3. Justifying the advantages of LPCAM over CAM from two perspectives: (a) clustering and (b) normalization. For simplicity, in (a), we consider only three local regions on a "bird" image: $\boldsymbol{x_1}$: "head", $\boldsymbol{x_2}$: "tail", $\boldsymbol{x_3}$: "sky". In (b), we assume two class prototypes ("head" and "tail"), and one context prototype ("sky") are selected after local feature clustering.

We justify the effectiveness of LPCAM from two perspectives: clustering and normalization. We use the "bird" example shown in Figure 3. In (a), we consider only three local regions ($\boldsymbol{x_1}$, $\boldsymbol{x_2}$ and $\boldsymbol{x_3}$), for simplicity. Their semantics are respectively: $\boldsymbol{x_1}$ as "head" (a discriminative object region), $\boldsymbol{x_2}$ as "tail" (a non-discriminative region), and $\boldsymbol{x_3}$ as "sky" (a context region). We suppose $f(\boldsymbol{x_1})$, $f(\boldsymbol{x_2})$, and $f(\boldsymbol{x_3})$ are 3-dimensional local features (2048-dimensional features in our real implementation) respectively extracted from the three regions, where the three dimensions represent the attributes of "head", "tail" and "sky", respectively. The discriminativeness is reflected as follows. First, $f(\boldsymbol{x_1})$ extracted on the head region $\boldsymbol{x_1}$ has a significantly higher value of the first dimension than $f(\boldsymbol{x_2})$ and $f(\boldsymbol{x_3})$. Second, $f(\boldsymbol{x_2})$ has the highest value on the second dimension but this value is lower than the first dimension of $f(\boldsymbol{x_1})$, because "tail" ($\boldsymbol{x_2}$) is less discriminative than "head" ($\boldsymbol{x_1}$) for recognizing "bird". In (b), we assume three local class prototypes ("head", "tail", and "sky") are selected.

Clustering. As shown in Figure 3(a), **in LPCAM**, $\boldsymbol{x_1}$ and $\boldsymbol{x_2}$ go to different clusters. This is determined by their dominant feature dimensions, i.e., the first dimension in $f(\boldsymbol{x_1})$ and the second dimension in $f(\boldsymbol{x_2})$. Given all samples of "bird", their features clustered into the "head" cluster all have high values in the first dimension, and features in the "tail" have high values in the second dimension. The centers of these clusters are taken as local prototypes and equally used for generating LPCAM. Sliding each of prototypes over the feature map block (of an input image) can highlight the corresponding local region. The intuition is each prototype works like a spatial-wise filter that amplifies similar regions and suppresses dissimilar regions.

However, **in CAM**, the heatmap computation uses the classifier weights biased on discriminative dimensions[2]. It is because the classifier is learned from the global average pooling features, e.g., $f_{GAP} = \frac{1}{3}(f(\boldsymbol{x_1}) + f(\boldsymbol{x_2}) + f(\boldsymbol{x_3})) = [12, 5, 4]$ biased to the "head" dimension. As a result, only discriminative regions (like "head" for the class of "bird") are highlighted on the heatmap of CAM.

Normalization. We justify the effectiveness of **LPCAM** by presenting the normalization details in Figure 3(b). We denote the two class prototypes ("head" and "tail") as $\mathbf{F'_1}, \mathbf{F'_2}$ and the context prototype ("sky") as $\mathbf{B'_1}$. Based on Eq. 5 and Eq. 6, we have $\boldsymbol{A}(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{2} sim(f(\boldsymbol{x}), \mathbf{F'_i}) - sim(f(\boldsymbol{x}), \mathbf{B'_1})$, where $\boldsymbol{x}$ denotes any local region. For simplicity, we use the first term for explanation: the prototype "head" $\mathbf{F'_1}$ has the highest similarity (1.0) to region $\boldsymbol{x_1}$ and the prototype "tail" $\mathbf{F'_2}$ has the highest similarity (0.9) to $\boldsymbol{x_2}$. 0.9 and 1.0 are very close. After the final maximum-value based normalization (as in the first formula of Eq. 6), they become 1.00 and 0.83, i.e., only a small gap between discriminative ($\boldsymbol{x_1}$) and non-discriminative ($\boldsymbol{x_2}$) regions. However, **CAM** (Eq. 1) uses $\mathbf{w}^\top f(\boldsymbol{x})$, resulting a much higher activation value of $\boldsymbol{x_1}$ than $\boldsymbol{x_2}$, as $\mathbf{w}$ is obviously biased to "head", i.e., 130 vs. 35 ("tail"). After the final maximum-value based normalization, there is no change to this bias: 1.00 and 0.27—a large gap. In other words, the non-discriminative feature is closer to background $\boldsymbol{x_3}$, making the boundary between foreground and background blurry, and hard to find a threshold to separate them.

One may argue that "separate $\boldsymbol{x_2}$ and $\boldsymbol{x_3}$" can be achieved in either CAM or LPCAM if the threshold is carefully selected in each method. However, it is not realistic to do such "careful selection" for every input image. The general way in WSSS is to use a common threshold for all images. Our LPCAM makes it easier to find such a threshold, since its heatmap has a much clearer boundary between foreground and background. We conduct a threshold sensitivity analysis in experiments to validate this.

[2]We empirically validate this in the supplementary materials.

4. Experiments

4.1. Datasets and Implementation Details

Datasets are the commonly used WSSS datasets: PASCAL VOC 2012 [11] and MS COCO 2014 [30]. PASCAL VOC 2012 contains 20 foreground object categories and 1 background category with $1,464$ `train` images, $1,449$ `val` images, and $1,456$ `test` images. Following related works [1, 8, 10, 24, 26, 51], we use the enlarged training set with $10,582$ training images provided by SBD [13]. MS COCO 2014 dataset consists of 80 foreground categories and 1 background category, with $82,783$ and $40,504$ images in `train` and `val` sets, respectively.

Evaluation Metrics. To evaluate the quality of seed mask and pseudo mask, we first generate them for every image in the `train` set and then use the ground truth masks to compute mIoU. For semantic segmentation, we train the segmentation model, use it to predict masks for the images in `val` and `test` sets, and compute mIoU based on ground truth masks.

Implementation Details. We follow [1, 8, 24, 26, 46] to use ResNet-50 [14] pre-trained on ImageNet [9] as the backbone of multi-label classification model. For fair comparison with related works, we also follow [10, 36, 51] to use WideResNet-38 [40] as backbone and follow EDAM [39] to use saliency maps [31] to refine CAM.

Extra Hyperparameters. For K-Means clustering, we set K as 12 and 20 for VOC and MS COCO, respectively. The threshold τ in Eq. 2 is set to 0.1 for VOC and 0.25 for MS COCO. For the selection of prototypes, μ_f is set to 0.9 on both datasets, and μ_b is 0.9 and 0.5 on VOC and MS COCO, respectively. We conduct the sensitivity analysis on these four hyperparameters to show that LPCAM is not sensitive to any of them.

Common Hyperparameter (in CAM methods). The hard threshold used to generate 0-1 seed mask is 0.3 for LPCAM on both datasets. Please note that we follow previous works [8, 10, 22, 24, 26, 46] to select this threshold by using the ground truth masks in the training set as "validation".

Time Costs. In K-Means clustering, we use all `train` set on VOC and sample 100 images per class on MS COCO (to control time costs). If taking the time cost of training a multi-label classification model as unit 1, our extra time cost (for clustering) is about 0.9 and 1.1 on VOC and MS COCO, respectively.

For Semantic Segmentation. When using DeepLabV2 [5] for semantic segmentation, we follow the common settings [1, 8, 10, 22, 24, 26] as follows. The backbone of DeepLabV2 model is ResNet-101 [14] and is pre-trained on ImageNet [9]. We crop each training image to the size of 321×321 and use horizontal flipping and random crop for data augmentation. We train the model for $20k$ and $100k$ iterations on VOC and MS COCO, respectively, with the respective batch size of 5 and 10. The weight decay is set to 5e-4 on both datasets and the learning rate is 2.5e-4 and 2e-4 on VOC and MS COCO, respectively.

When using UperNet, we follow ReCAM [8]. We resize the input images to $2,048 \times 512$ with a ratio range from 0.5 to 2.0, and then crop them to 512×512 randomly. Data augmentation includes horizontal flipping and color jitter. We train the models for $40k$ and $80k$ iterations on VOC and MS COCO datasets, respectively, with a batch size of 16. We deploy 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 polynomial decay schedule.

4.2. Results and Analyses

Ablation Study. We conduct an ablation study on the VOC dataset to evaluate the two terms of LPCAM in Eq. 6: foreground term $\boldsymbol{FG}_n$ and background term $\boldsymbol{BG}_n$ that accord to class and context prototypes, respectively. In Table 1, we show the mIoU results (of seed masks), false positive (FP), false negative (FN), precision, and recall. We can see that our methods of using class prototypes (LPCAM-F and LPCAM) greatly improve the recalls—11.4% and 12.0% higher than CAM, reducing the rates of FN a lot. This validates the ability of our methods to capture non-discriminative regions of the image. We also notice that LPCAM-F increases the rate of FP over CAM. The reason is that confusing context features (e.g., "railroad" for "train") may be wrongly taken as class features. Fortunately, when we explicitly resolve this issue by applying the negative context term $-\boldsymbol{BG}_n$ in LPCAM, this rate can be reduced (by 3.3% for VOC), and the overall performance (mIoU) can be improved (by 2.8% for VOC). We are thus confident to take LPCAM as a generic substitute of CAM in WSSS methods (see empirical validations below).

	FP	FN	mIoU	Prec.	Recall
`CAM`	26.5	26.2	48.8	65.0	65.2
`LPCAM-F`	33.1+6.6	16.2-10.0	52.1+3.3	61.3-3.7	76.6+11.4
`LPCAM`	29.8+3.3	16.7-9.5	54.9+6.1	64.9-0.1	77.2+12.0

Table 1. An ablation study on VOC dataset. "-F" denotes only the "Foreground" term $\boldsymbol{FG}_n$ is used in Eq. 6. *Please refer to the supplementary materials for the results on MS COCO.*

Generality of LPCAM. We validate the generality of LPCAM based on multiple WSSS methods, the popular IRN [1], the top-performing AMN [26], the saliency-map-based EDAM [39], and the transformer-arch-based MCTformer [45]), by simply replacing CAM with LPCAM. Table 2 and Table 3 show the consistent superiority of LPCAM. For example, on the first row of Table 2 (plugging LPCAM in IRN), LPCAM outperforms CAM by 6.1% on seed masks and 4.7% on pseudo masks. These margins are almost maintained when using pseudo masks to train se-

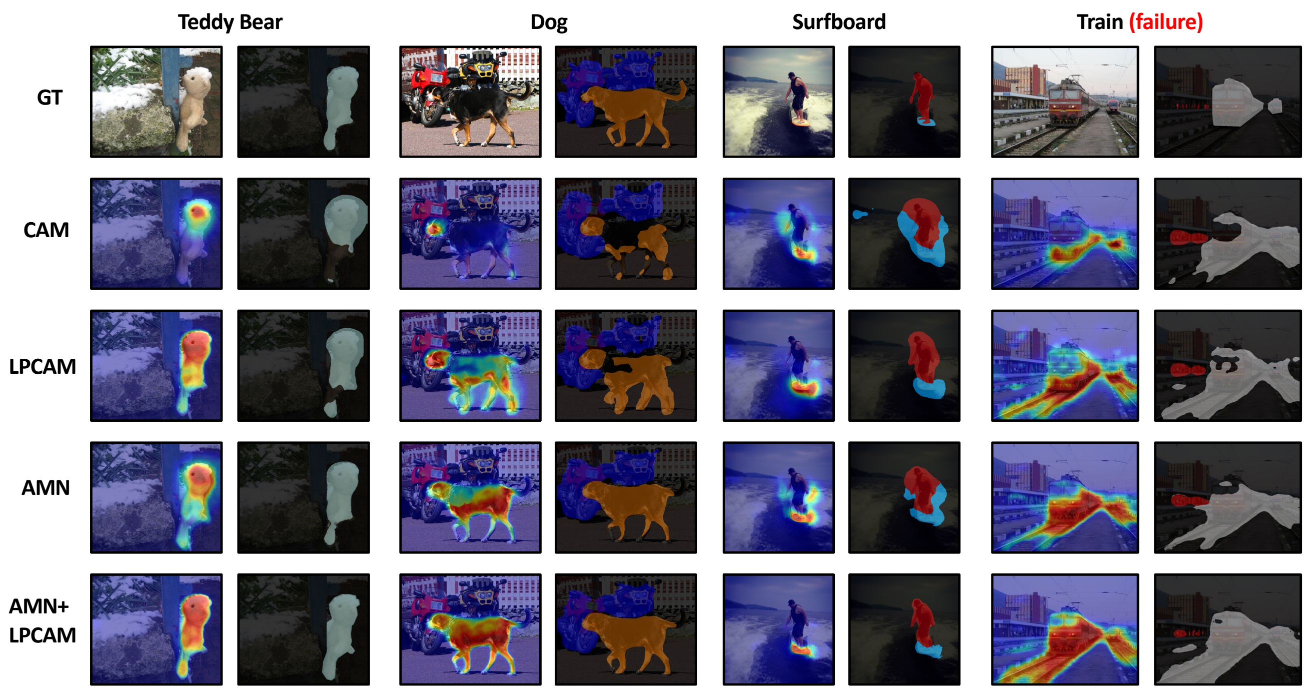


Figure 4. Qualitative results on MS COCO. In each example pair, the left is heatmap and the right is seed mask. *Please refer to the supplementary materials for the qualitative results on VOC.*

	Methods	Seed Mask		Pseudo Mask	
		CAM	LPCAM	CAM	LPCAM
VOC	IRN [1]	48.8	54.9+6.1	66.5	71.2+4.7
	EDAM [39]	52.8	54.9+2.1	68.1	69.6+1.5
	MCTformer [45]	61.7	63.5+1.8	69.1	70.8+1.7
	AMN [26]	62.1	65.3+3.2	72.2	72.7+0.5
COCO	IRN [1]	33.1	35.8+2.7	42.5	46.8+4.3
	AMN [26]	40.3	42.5+2.2	46.7	47.7+1.0

Table 2. Taking LPCAM as a substitute of CAM in state-of-the-art WSSS methods. Except MCTformer [45] using DeiT-S [35], other methods all use ResNet-50 as feature extractor.

mantic segmentation models in Table 3. The improvements on the large-scale dataset MS COCO are also obvious and consistent, e.g., 2.7% and 2.2% for generating seed masks in IRN and AMN, respectively.

Sensitivity Analysis for Hyperparameters. In Figure 5, we show the quality (mIoU) of generated seed masks when plugging LPCAM in AMN on VOC dataset. We perform hyperparameter sensitivity analyses by changing the values of (a) the threshold τ for dividing foreground and background local features, (b) the threshold μ_f for selecting class prototypes and the threshold μ_b for selecting context prototypes, (c) the number of clusters K in K-Means, and (d) the threshold used to generate 0-1 seed mask (a common hyperparameter in all CAM-based methods). Figure 5(a)

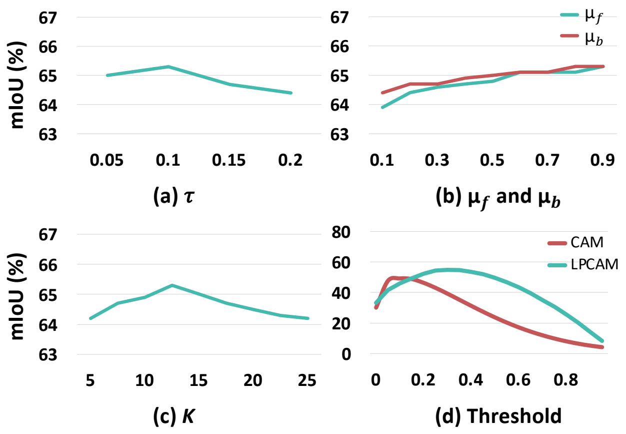


Figure 5. Sensitivity analysis on VOC, in terms of (a) τ for dividing foreground and background local features, (b) μ_f for selecting class prototypes and μ_b for selecting context prototypes, (c) the number of clusters K in k-Means, and (d) the threshold used to generate 0-1 seed masks from heatmaps (a common hyperparameter in all CAM-based methods). *Please refer to the supplementary materials for the results on MS COCO.*

shows that the optimal value of τ is 0.1. Adding a small change does not make any significant effect on the results, e.g., the drop is less than 1% if increasing τ to 0.2. Figure 5(b) shows that the optimal values of μ_f and μ_b are both 0.9. The gentle curves show that LPCAM is little sensitive to μ_f and μ_b. This is because classification models (trained in the first step of WSSS) often produce overcon-

Methods	VOC								MS COCO			
	DeepLabV2				UperNet-Swin				DeepLabV2		UperNet-Swin	
	`CAM`		`LPCAM`		`CAM`		`LPCAM`		`CAM`	`LPCAM`	`CAM`	`LPCAM`
	`val`	`test`	`val`	`test`	`val`	`test`	`val`	`test`	`val`	`val`	`val`	`val`
IRN [1]	63.5	64.8	$68.6_{+5.1}$	$68.7_{+3.9}$	65.9	67.7	$71.1_{+5.2}$	$71.8_{+4.1}$	42.0	$44.5_{+2.5}$	44.0	$47.0_{+3.0}$
AMN [26]	69.5	69.6	$70.1_{+0.6}$	$70.4_{+0.8}$	71.7	71.8	$73.1_{+1.4}$	$73.4_{+1.6}$	44.7	$45.5_{+0.8}$	47.1	$48.3_{+1.2}$
EDAM [39]	70.9*	70.6*	$71.8^{*}_{+0.9}$	$72.1^{*}_{+1.5}$	71.2	71.0	$72.7_{+1.6}$	$72.5_{+1.5}$	40.6	$42.1_{+1.5}$	41.7	$43.0_{+1.3}$
MCTformer [45]	$71.9^{\dagger}$	$71.6^{\dagger}$	$72.6^{\dagger}_{+0.7}$	$72.4^{\dagger}_{+0.8}$	70.6	70.3	$72.0_{+1.4}$	$72.5_{+2.2}$	-	-	-	-

Table 3. The mIoU results (%) of WSSS using different segmentation models on VOC and MS COCO. Seed masks are generated by either CAM or LPCAM, and mask refinement methods are row titles. * denotes the segmentation model (ResNet-101 based DeepLabV2) is pre-trained on MS COCO. † denotes the segmentation model (WideResNet-38 based DeepLabV2) is pre-trained on VOC.

	Methods	Sal.	VOC		MS COCO
			`val`	`test`	`val`
ResNet-50	IRN [1] CVPR'19		63.5	64.8	42.0
	LayerCAM [19] TIP'21		63.0	64.5	-
	AdvCAM [24] CVPR'21		68.1	68.0	44.2
	RIB [22] NeurIPS'21		68.3	68.6	44.2
	ReCAM [8] CVPR'22		68.5	68.4	42.9
	IRN+`LPCAM`		68.6	68.7	44.5
	SIPE [7] CVPR'22		68.8	69.7	40.6
	OOD [25]+Adv CVPR'22		69.8	69.9	-
	AMN [26] CVPR'22		69.5	69.6	44.7
	AMN+`LPCAM`		70.1	70.4	45.5
	ESOL [28] NeurIPS'22		69.9*	69.3*	42.6
	CLIMS [42] CVPR'22		70.4*	70.0*	-
	EDAM [39] CVPR'21	✓	70.9*	71.8*	-
	EDAM+`LPCAM`	✓	71.8*	72.1*	42.1
WideResNet-38	Spatial-BCE [38] ECCV'22		70.0	71.3	35.2
	BDM [43] ACMMM'22	✓	71.0	71.0	36.7
	RCA [51]+OOA CVPR'22	✓	71.1	71.6	35.7
	RCA [51]+EPS CVPR'22	✓	72.2	72.8	36.8
	HGNN [47] ACMMM'22	✓	70.5*	71.0*	34.5
	EPS [27] CVPR'21	✓	70.9*	70.8*	-
	RPIM [34] ACMMM'22	✓	71.4*	71.4*	-
	L2G [18] CVPR'22	✓	72.1*	71.7*	44.2
DeiT-S	MCTformer [45] CVPR'22		$71.9^{\dagger}$	$71.6^{\dagger}$	42.0
	MCTformer+`LPCAM`		$72.6^{\dagger}$	$72.4^{\dagger}$	42.8

Table 4. The mIoU results (%) based on DeepLabV2 on VOC and MS COCO. The side column shows three backbones of multi-label classification model. "Sal." denotes using saliency maps. * denotes the segmentation model is pre-trained on MS COCO. † denotes the segmentation model is pre-trained on VOC.

fident (sharp) predictions [12], i.e., output probabilities are often close to 0 or 1. It is easy to set thresholds (μ_f and μ_b) on such sharp values. In Figure 5(c), the best mIoU of seed mask is 65.3% when K=12, and it drops by only 0.7 percentage points when K goes up to 20. In Figure 5(d), LPCAM shows much gentler slopes than CAM around their respective optimal points, indicating its lower sensitivity to the changes of this threshold.

Qualitative Results. Figure 4 shows qualitative examples where LPCAM leverages both discriminative and non-discriminative local features to generate heatmaps and 0-1 masks. In the leftmost two examples, CAM focuses on only discriminative features, e.g., the "head" regions of "teddy bear" and "dog", while our LPCAM has better coverage on the non-discriminative feature, e.g., the "leg", "body" and "tail" regions. In the "surfboard" example, the context prototype term $-\boldsymbol{BG}_n$ in Eq. 6 helps to remove the context "waves". In the rightmost example, we show a failure case: LPCAM succeeds in capturing more object parts of "train" but unnecessarily covers more on the context "railroad". We think the reason is the strong co-occurrence of "train" and "railroad" in the images of "train".

Comparing to Related Works. We compare LPCAM with state-of-the-art methods in WSSS. As shown in Table 4, on the common setting (ResNet-50 based classification model, and ResNet-101 based DeepLabV2 segmentation model pre-trained on ImageNet), our AMN+LPCAM achieves the state-of-the-art results on VOC (70.1% mIoU on `val` and 70.4% on `test`). On the more challenging MS COCO dataset, our AMN+LPCAM (ResNet-50 as backbone) outperforms the state-of-the-art result AMN and all related works based on WRN-38.

5. Conclusions

We pointed out that the crux behind the poor coverage of the conventional CAM is that the global classifier captures only the discriminative features of objects. We proposed a novel method dubbed LPCAM to leverage both discriminative and non-discriminative local prototypes to generate activation maps with complete coverage on objects. Our extensive experiments and studies on two popular WSSS benchmarks showed the superiority of LPCAM over CAM.

Acknowledgments The author gratefully acknowledges the support of the Lee Kong Chian (LKC) Fellowship fund awarded by Singapore Management University, and the A*STAR under its AME YIRG Grant (Project No.A20E6c0101).

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, 6, 7, 8

[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

[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

[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, 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

[7] Qi Chen, Lingxiao Yang, Jian-Huang Lai, and Xiaohua Xie. Self-supervised image-specific prototype exploration for weakly supervised semantic segmentation. In *CVPR*, pages 4288–4298, 2022. 8

[8] Zhaozheng Chen, Tan Wang, Xiongwei Wu, Xian-Sheng Hua, Hanwang Zhang, and Qianru Sun. Class re-activation maps for weakly-supervised semantic segmentation. In *CVPR*, 2022. 1, 3, 6, 8

[9] 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

[10] Ye Du, Zehua Fu, Qingjie Liu, and Yunhong Wang. Weakly supervised semantic segmentation by pixel-to-prototype contrast. In *CVPR*, pages 4320–4329, 2022. 3, 6

[11] 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, 6

[12] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, pages 1321–1330, 2017. 8

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

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 3, 6

[15] 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

[16] 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

[17] 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

[18] Peng-Tao Jiang, Yuqi Yang, Qibin Hou, and Yunchao Wei. L2g: A simple local-to-global knowledge transfer framework for weakly supervised semantic segmentation. In *CVPR*, 2022. 3, 8

[19] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. *TIP*, 30:5875–5888, 2021. 8

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

[21] 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

[22] Jungbeom Lee, Jooyoung Choi, Jisoo Mok, and Sungroh Yoon. Reducing information bottleneck for weakly supervised semantic segmentation. In *NeurIPS*, pages 27408–27421, 2021. 1, 3, 6, 8

[23] 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

[24] 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, 3, 6, 8

[25] Jungbeom Lee, Seong Joon Oh, Sangdoo Yun, Junsuk Choe, Eunji Kim, and Sungroh Yoon. Weakly supervised semantic segmentation using out-of-distribution data. In *CVPR*, 2022. 2, 8

[26] Minhyun Lee, Dongseob Kim, and Hyunjung Shim. Threshold matters in wsss: Manipulating the activation for the robust and accurate segmentation model against thresholds. In *CVPR*, 2022. 1, 2, 3, 6, 7, 8

[27] 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

[28] Jinlong Li, Zequn Jie, Xu Wang, Xiaolin Wei, and Lin Ma. Expansion and shrinkage of localization for weakly-supervised semantic segmentation. In *NeurIPS*, 2022. 8

[29] 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. 3

[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. 2, 6

[31] Jiang-Jiang Liu, Qibin Hou, Ming-Ming Cheng, Jiashi Feng, and Jianmin Jiang. A simple pooling-based design for real-time salient object detection. In *CVPR*, pages 3917–3926, 2019. 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] Chen Qian and Hui Zhang. Region-based pixels integration mechanism for weakly supervised semantic segmentation. In *ACMMM*, pages 6165–6173, 2022. 8

[35] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357, 2021. 7

[36] 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, 6

[37] 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. 1, 2

[38] Tong Wu, Guangyu Gao, Junshi Huang, Xiaolin Wei, Xiaoming Wei, and Chi Harold Liu. Adaptive spatial-bce loss for weakly supervised semantic segmentation. In *ECCV*, pages 199–216, 2022. 8

[39] 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, 6, 7, 8

[40] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel. Wider or deeper: Revisiting the resnet model for visual recognition. *Pattern Recognition*, 90:119–133, 2019. 6

[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

[42] Jinheng Xie, Xianxu Hou, Kai Ye, and Linlin Shen. Cross language image matching for weakly supervised semantic segmentation. In *CVPR*, pages 4483–4492, 2022. 8

[43] Jianjun Xu, Hongtao Xie, Hai Xu, Yuxin Wang, Sun-ao Liu, and Yongdong Zhang. Boat in the sky: Background decoupling and object-aware pooling for weakly supervised semantic segmentation. In *ACMMM*, pages 5783–5792, 2022. 8

[44] 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. 3

[45] Lian Xu, Wanli Ouyang, Mohammed Bennamoun, Farid Boussaid, and Dan Xu. Multi-class token transformer for weakly supervised semantic segmentation. In *CVPR*, pages 4310–4319, 2022. 1, 2, 6, 7, 8

[46] Dong Zhang, Hanwang Zhang, Jinhui Tang, Xiansheng Hua, and Qianru Sun. Causal intervention for weakly-supervised semantic segmentation. In *NeurIPS*, pages 655–666, 2020. 1, 3, 6

[47] Meijie Zhang, Jianwu Li, and Tianfei Zhou. Multi-granular semantic mining for weakly supervised semantic segmentation. In *ACMMM*, pages 6019–6028, 2022. 8

[48] 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. 1, 2

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

[50] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, pages 2921–2929, 2016. 1

[51] Tianfei Zhou, Meijie Zhang, Fang Zhao, and Jianwu Li. Regional semantic contrast and aggregation for weakly supervised semantic segmentation. In *CVPR*, pages 4299–4309, 2022. 3, 6, 8