Appendix for:
gScoreCAM: What objects is CLIP looking at?

A1 Derive bounding box from heatmap

A1.1 Get bounding box with Otsu's method

Fig. S1 shows the intermediate results during the procedure (prompt: dog). As described in Algorithm 1, we obtain the bounding box using the following procedure:

1. From an input image, we used a CAM method to get a heatmap.
2. Binarize the heatmap with Otsu's method (using OpenCV function $cv.threshold$ with option $cv.THRESH\_OTSU$).
3. Find the contours of the binary map (using the OpenCV function $cv2.findContours$).
4. For each contour, determine a minimal bounding box (using the OpenCV function $cv2.boundingRect$).
5. Then choose the largest bounding box as result.

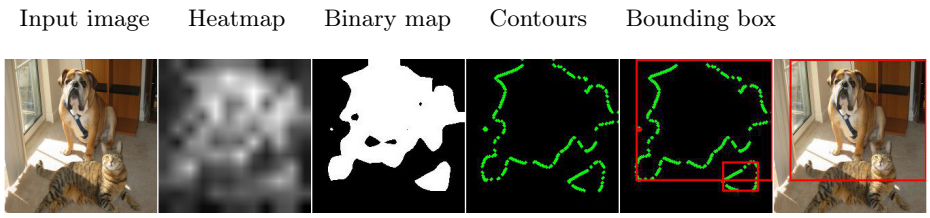


Fig. S1: From left to right is the procedure we derive bounding box from heatmap. We first get the heatmap from input image. Then use Otsu's method to find the binary map. We find sets of bounding boxes from the contours and then choose the largest one as our final result.

Algorithm 1 Derive Bounding box from Heatmap

Input: (i) A heatmap $M \in \mathbb{R}^{u \times v}$ from any CAM methods. (ii) Input image size (w, h).
Output: A bounding box for target class or prompt.

$M \leftarrow Upsample(M, size=(w,\ h), method="bilinear")$
$M \leftarrow Otsu(M)$
$contours \leftarrow findContours(M)$
$boxes = [[0, 0, 0,]...[0, 0, 0, 0]]$
for i *in* $len(contours)$:
    $boxes[i] \leftarrow boundingRect(contours[i])$
$u \leftarrow \arg\max Area(boxes)$
$output \leftarrow boxes[u]$

A1.2 Effects on using Otsu's method

In this section, we study how Otsu's method differs from the commonly used single threshold by grid search. We performe the experiment on the COCO validation set. We keep the other procedure the same as in Algorithm 1, except that we replace Otsu's binarization with using a single threshold. We perform a grid search for the optimal threshold on the subset of images in the training set (100 images per class) for each method with $gridsize = 0.05$. The search is based on the mean IoU over the search samples. We find that the difference between using Otsu's method and using optimal threshold by grid search is trivial.

Table S1: Difference between Otsu binarzation with optimal thesholding. The difference between Otsu's method and using optimal threshold value is very small.

	Single value	Otsu	Difference
GradCAM	11.95%	11.56%	-0.39%
GradCAM++	9.03%	9.68%	0.65%
xGradCAM	6.85%	5.60%	-1.25%
GroupCAM	5.58%	4.52%	-1.06%
LayerCAM	9.99%	9.19%	-0.80%
ScoreCAM	20.74%	20.43%	-0.31%
gScoreCAM	20.27%	20.83%	0.56%
Hila' method(ViT-B/32)	13.33%	12.82%	-0.51%
ScoreCAM(ViT-B/32)	9.60%	10.21%	0.61%
gScoreCAM(ViT-B/32)	9.40%	10.10%	0.70%
Mean	11.67%	11.49%	-0.18%

A2 Hyperpapamters of CLIP RN50, RN50x16 and ViT-B/32

We list some key hyperparameters of the three CLIP models (RN50, RN50x16 and ViT-B/32) we use in our experiments. A complete list of hyperparameters can be found in Table 19 of Radford et al. [**?**].

Table S2: Some key hyperparameters of the CLIP models [**?**] used in our experiments.

	Embedding dimension	Input resolution	ResNet-50 blocks	ResNet-50 width	Vision Transformer (ViT) layers	Vision Transformer (ViT) width	Vision Transformer (ViT) heads
RN50x4	640	288	(4,6,10,6)	2560	N/A		
RN50x16	768	384	(6,8,18,8)	3072			
ViT-B/32	512	224	N/A		12	768	12

A3 Localization on scenes (ADE20K)

In order to study the performance in the scenario in which the target is relatively large instead of a small object, we evaluate different CAM based localization methods on ten different scenes of ADE20K. We choose the scene of *field*, *hill*, *river*, *grass*, *sky*, *sand*, *sea*, *snow*, *water*, and *road*. These scenes, on average, cover 17.5% of the image, where objects on COCO cover 4.9% of the image. Among all the methods tested, gScoreCAM provides the highest average IoU.

Table S3: We measure the average IoU on 10 different scenes; the result shows that gScoreCAM still is the best method even on large scenes.

	GradCAM	GradCAM++	xGradCAM	GroupCAM	LayerCAM	gScoreCAM	ScoreCAM
RN50x4	0.073	0.105	0.112	0.161	0.068	**0.211**	0.198
RN50x16	0.141	0.102	0.078	0.168	0.061	**0.216**	0.190

A4 Visualizations of different methods

In this section, we show a series of comparisons between different methods. We find that for these hard tasks (Fig. S4), gScoreCAM outperforms other methods in COCO and PartImageNet.

A4.1 Advantages of gScoreCAM

We show a few sample visualizations of gScoreCAM, GradCAM, ScoreCAM and HilaCAM in Figs. S3 and S4.

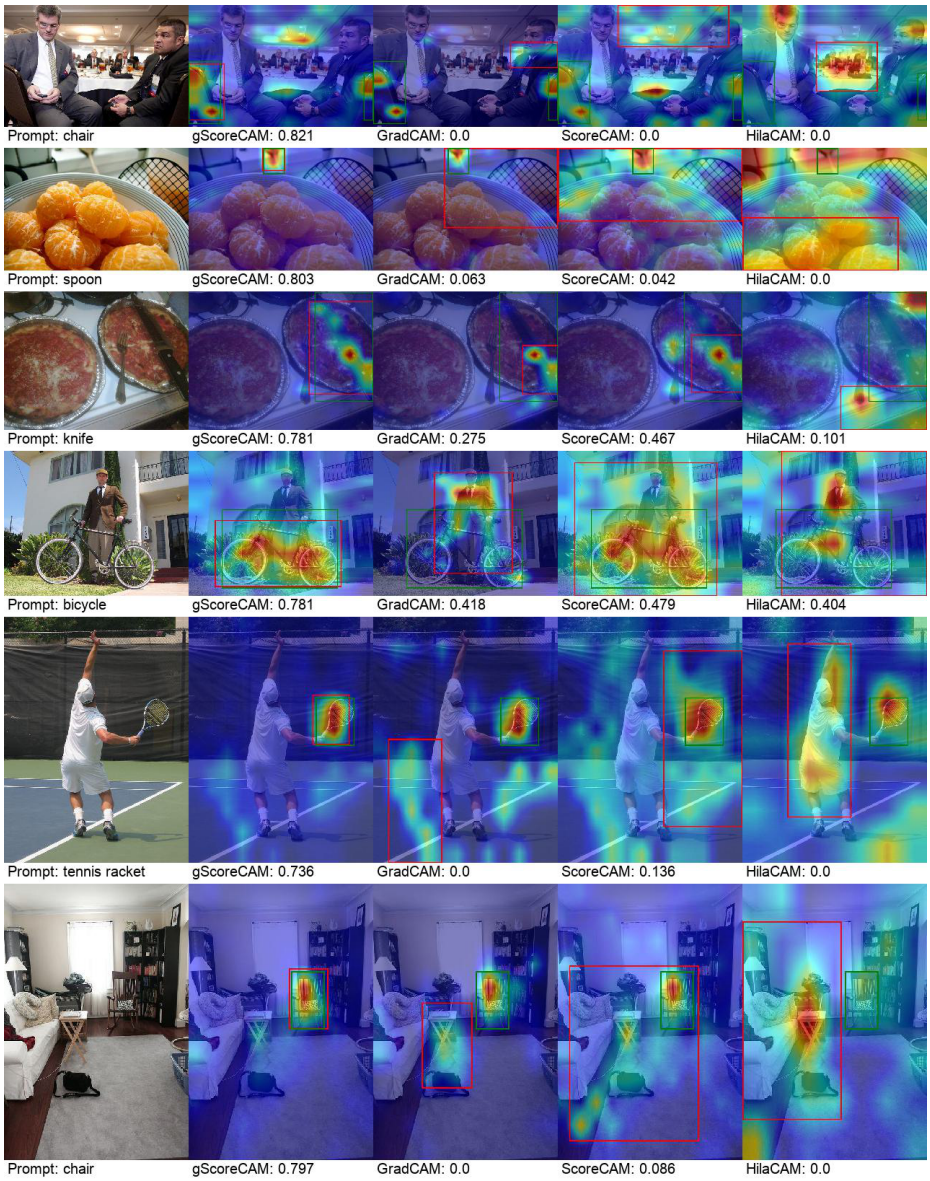


Fig. S2: ZSD results COCO dataset.

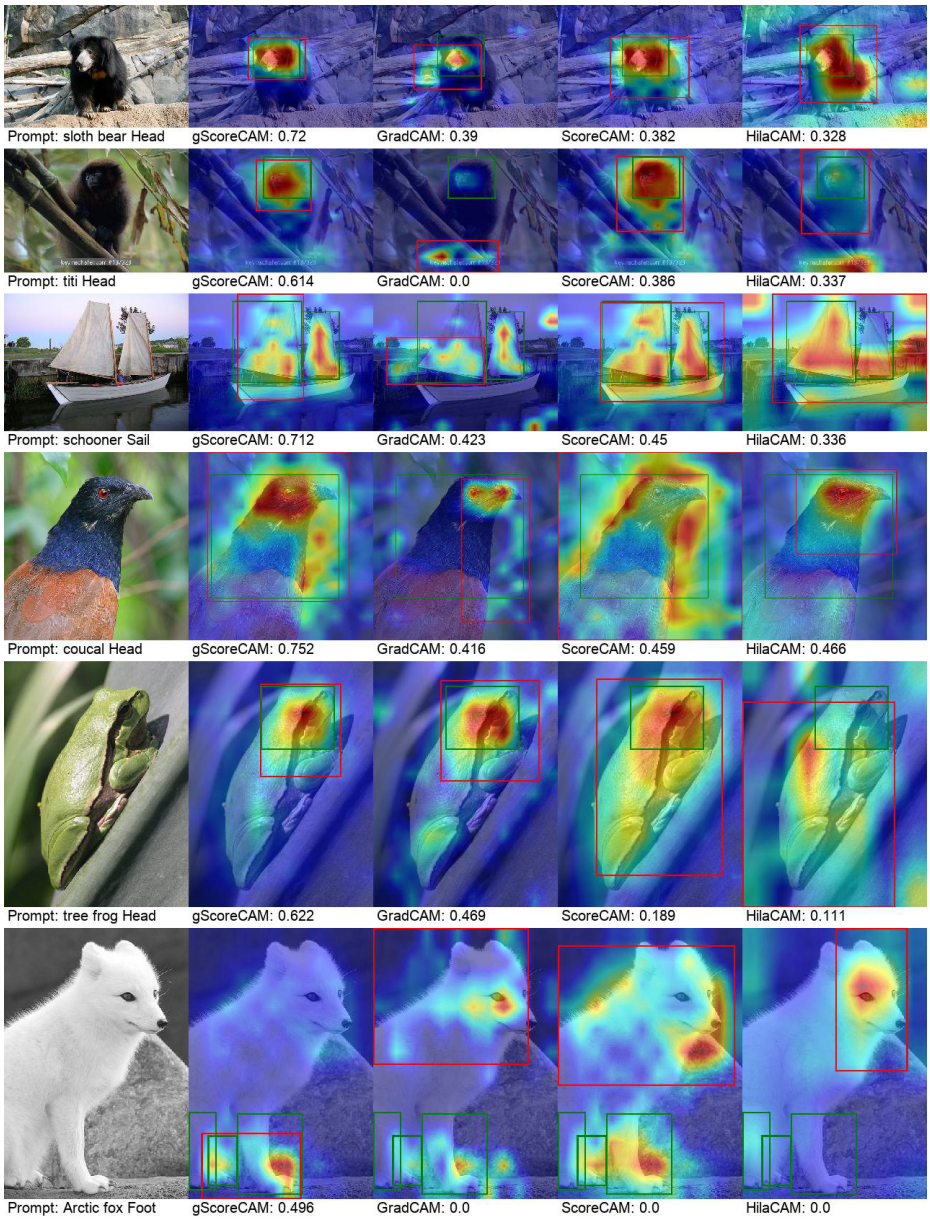
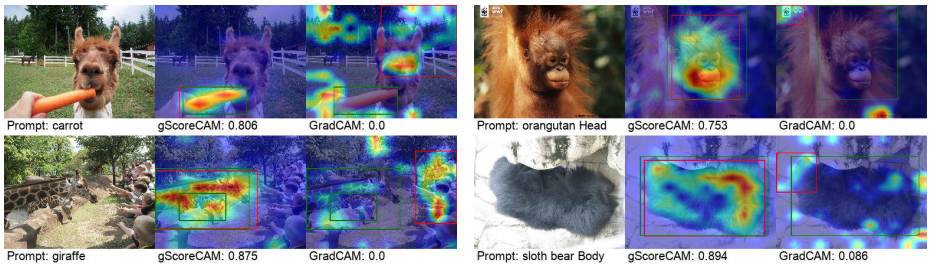
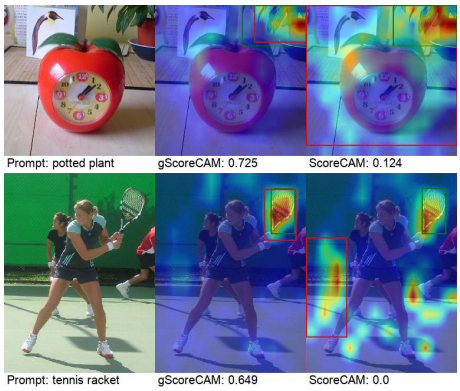


Fig. S3: ZSD results on PartImageNet dataset.

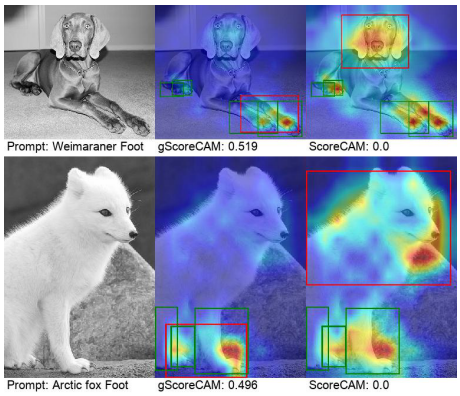


(a) COCO

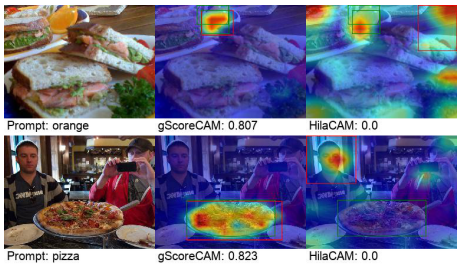
(b) Part ImageNet



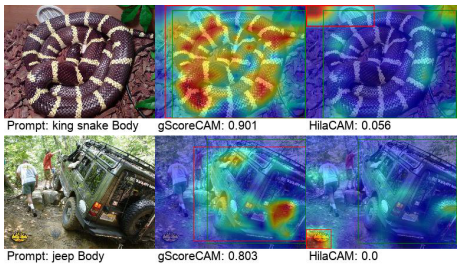
(c) COCO



(d) Part ImageNet



(e) COCO



(f) Part ImageNet

Fig. S4: Sample visualization comparison between gScoreCAM to GradCAM, ScoreCAM, and HilaCAM. We find that GradCAM always has low coverage which ScoreCAM tends to have large coverage. HilaCAM is something in the middle but have some "corner" issues. gScoreCAM can capture the target object most of the time although the resulting bounding box (red) may not be very accurate.

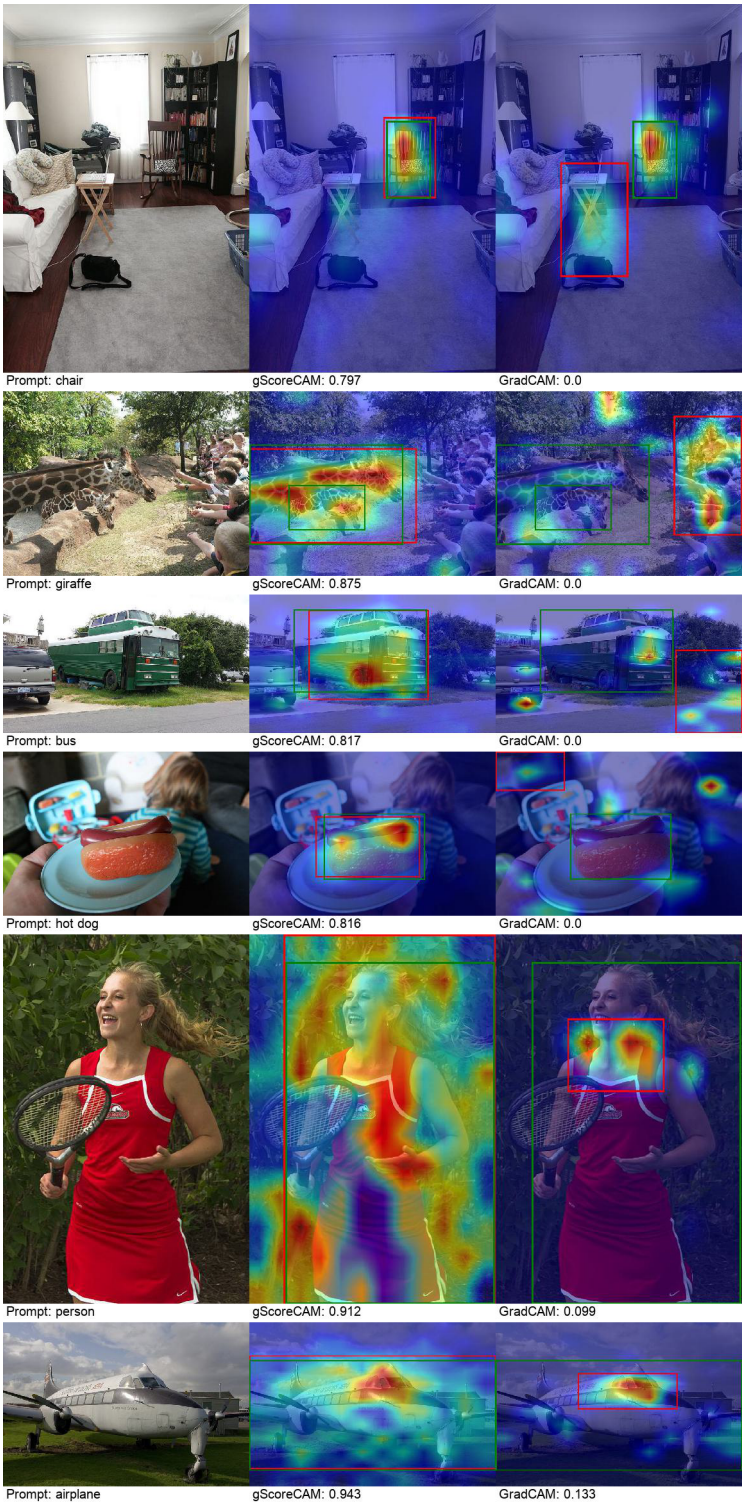


Fig. S5: Some samples that gScoreCAM performs better than GradCAM on COCO dataset.

Fig. S6: Some samples that gScoreCAM performs better than ScoreCAM on COCO dataset.

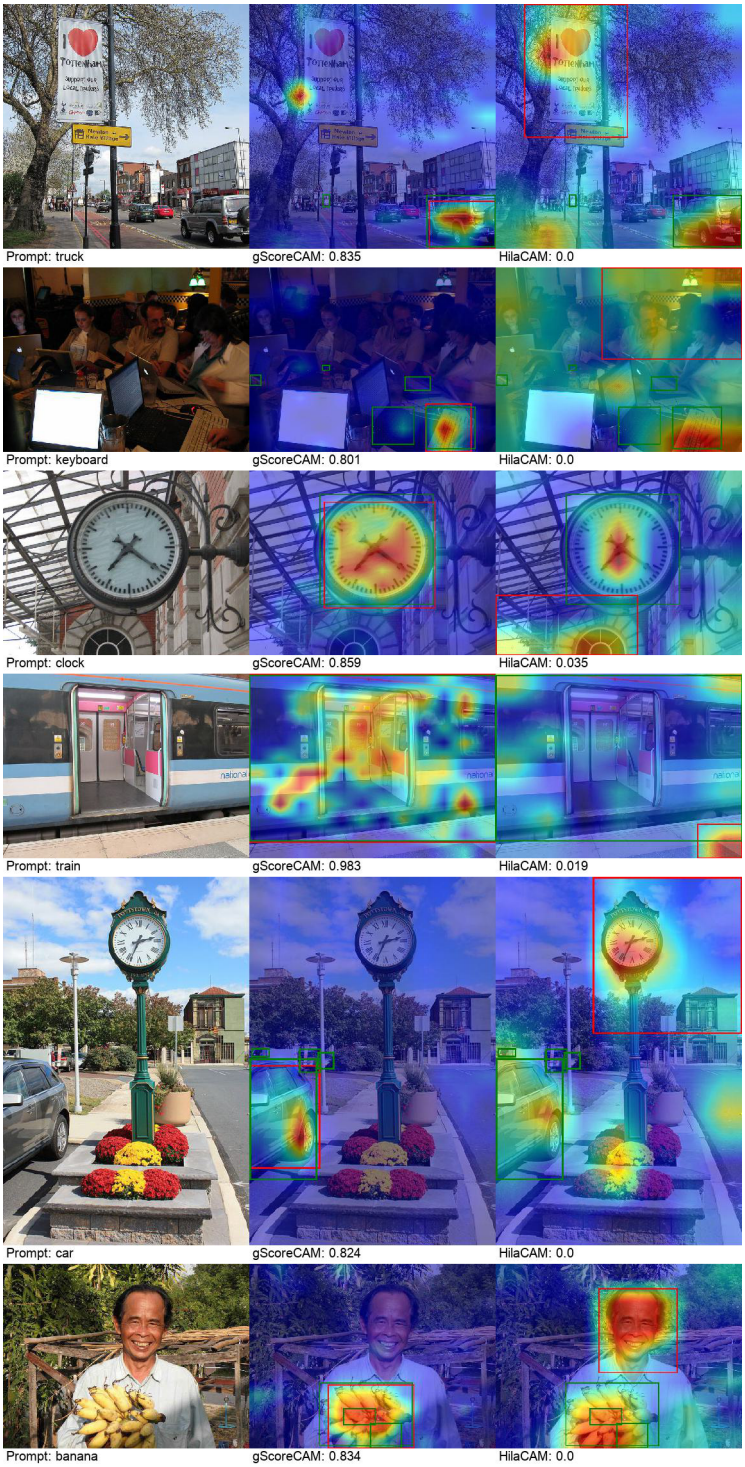


Fig. S7: Some samples that gScoreCAM performs better than HilaCAM on COCO dataset.

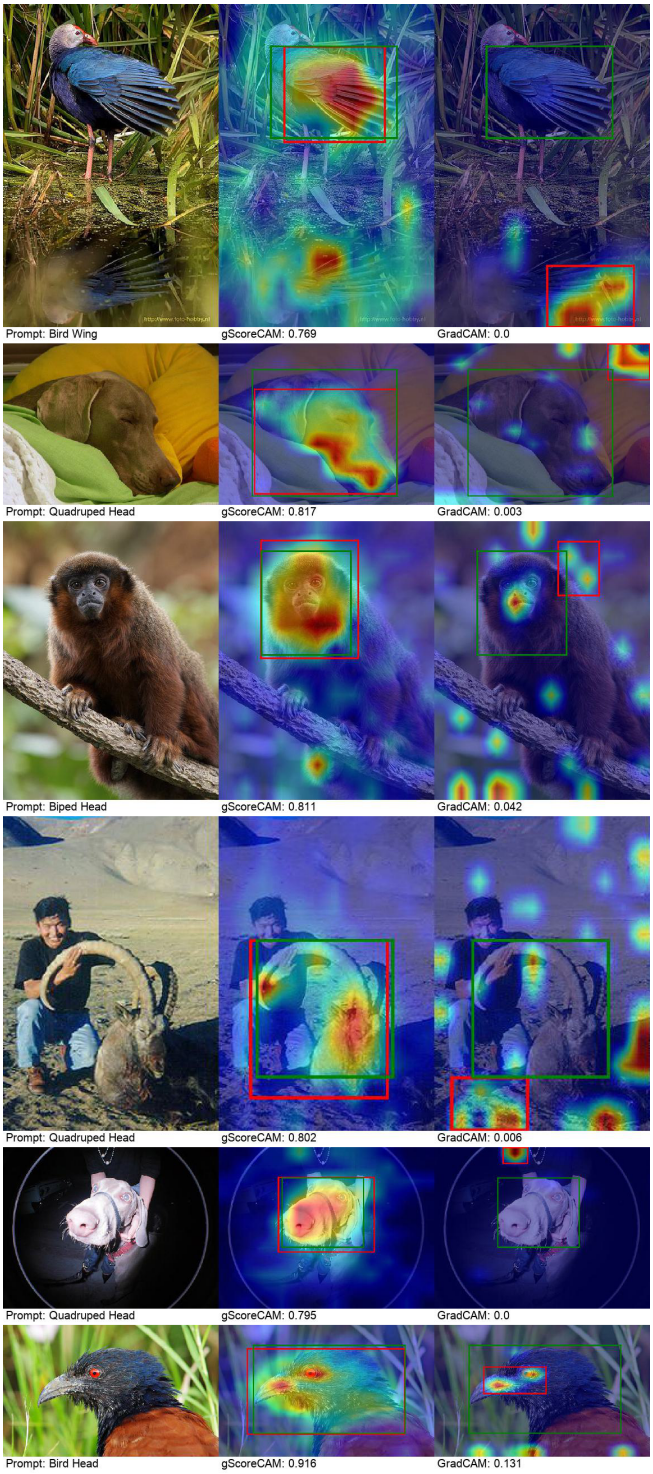


Fig. S8: Some samples that gScoreCAM performs better than GradCAM on Part ImageNet dataset.

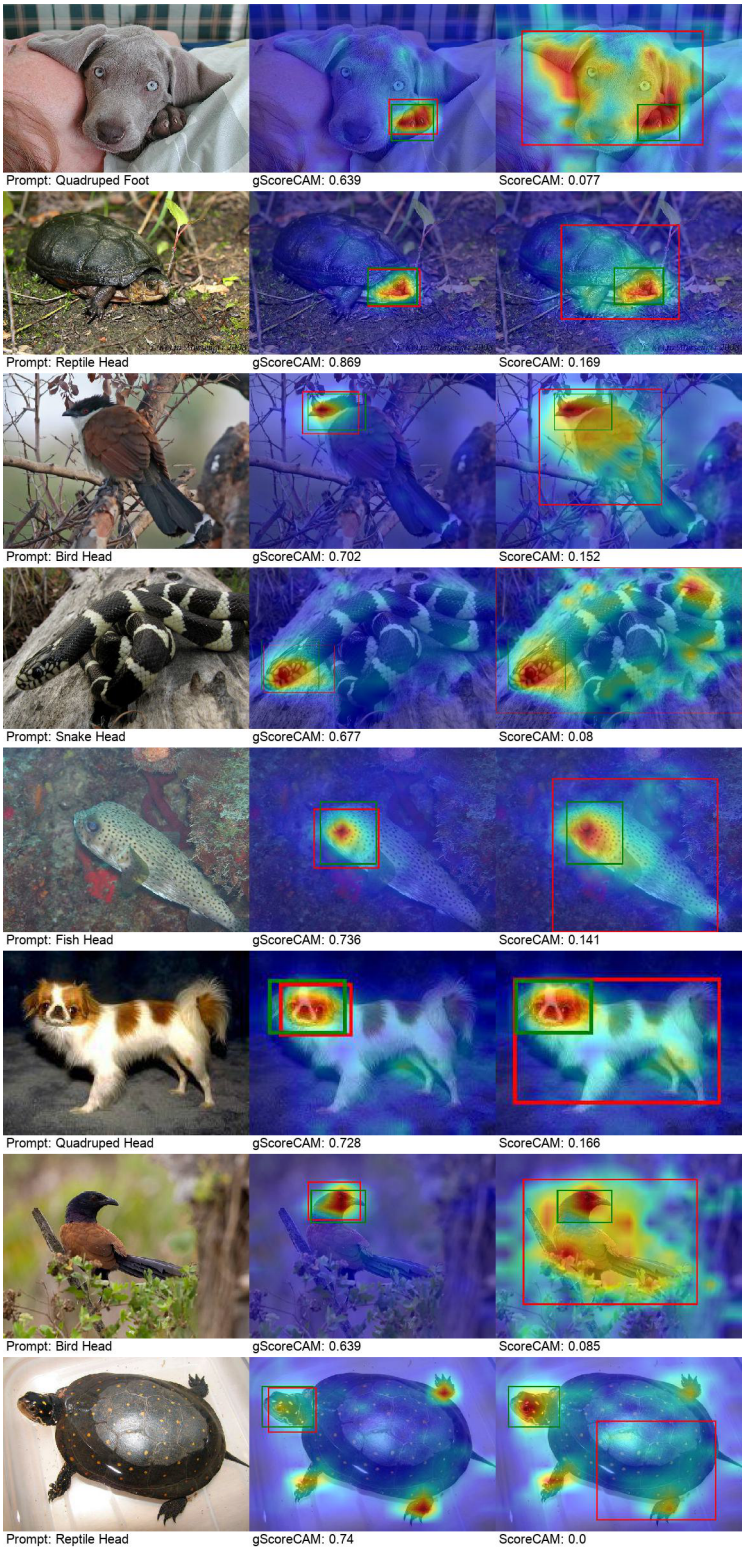


Fig. S9: Some samples that gScoreCAM performs better than ScoreCAM on Part ImageNet dataset.

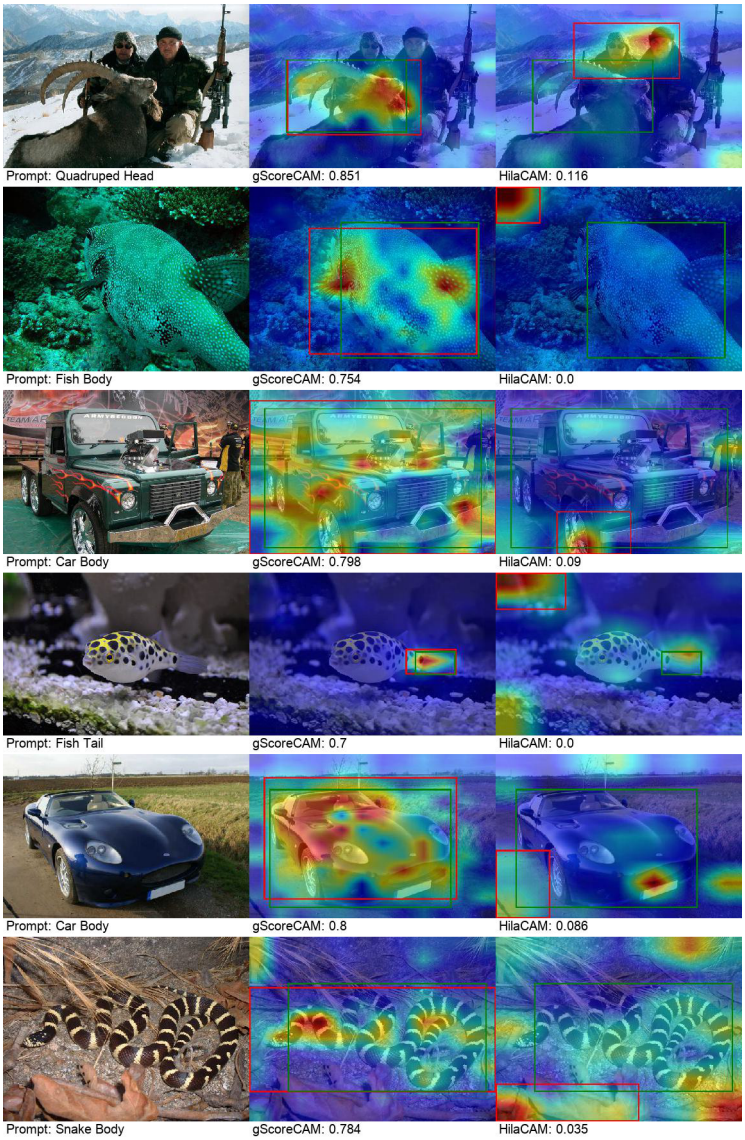


Fig. S10: Some samples that gScoreCAM performs better than HilaCAM on Part ImageNet dataset.