gScoreCAM: What objects is CLIP looking at?

Peijie Chen[1], Qi Li[1], Saad Biaz[1], Trung Bui[2], and Anh Nguyen[1]

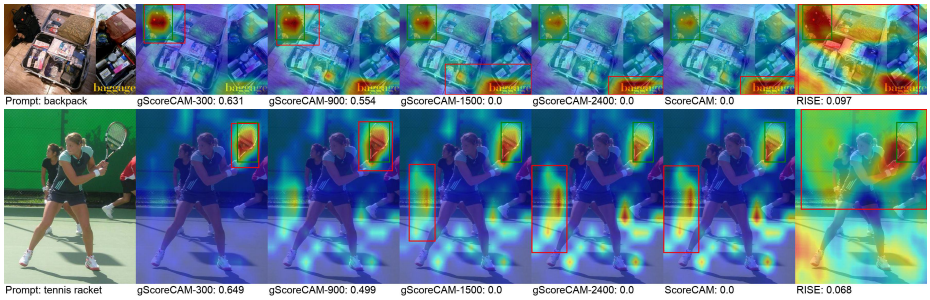
[1] Auburn University
{peijiechen, anh.ng8}@gmail.com, {qzl0019, biazsaa}@auburn.edu
[2] Adobe Research
bui@adobe.com

Abstract. Large-scale, multimodal models trained on web data such as OpenAI's CLIP are becoming the foundation of many applications. Yet, they are also more complex to understand, test, and therefore align with human values. In this paper, we propose gScoreCAM—a state-of-the-art method for visualizing the main objects that CLIP is looking at in an image. On zero-shot object detection, gScoreCAM performs similarly to ScoreCAM, the best prior art on CLIP, yet 8 to 10 times faster. Our method outperforms other existing, well-known methods (HilaCAM, RISE, and the entire CAM family) by a large margin, especially in multi-object scenes. gScoreCAM sub-samples $k = 300$ channels (from 3,072 channels—i.e. reducing complexity by almost 10 times) of the highest gradients and linearly combines them into a final "attention" visualization. We demonstrate the utility and superiority of our method on three datasets: ImageNet, COCO, and PartImageNet. Our work opens up interesting future directions in understanding and de-biasing CLIP.

1 Introduction

Large-scale, multimodal neural networks trained on web data are becoming important "foundation models" [1] for academic research, industry applications, and the wider society. Within only one year since release, OpenAI's CLIP [2], which learns to match captions and images, has powered a multitude of applications [3], including text-to-image synthesis [4,5,6], video retrieval [7,8], visual question answering [9], and image editing [10,11]. As foundation models are scaled up larger and becoming more ubiquitous, it is imperative to understand how they work internally to ensure safe deployment, avoid unexpected harms due to biases [12,13,14], and also further improve the models' functionality [15]. However, the inner-workings of foundation models (here, CLIP) are still largely unknown. For example, it is intriguing why the highly-knowledgeable CLIP model is fooled by a simple piece of paper with text [16] (see Fig. 4). In a complex, many-object scene (Fig. 1), which objects are *important* to CLIP?

Existing ViT-based interpretability methods [17,18,19] visualize the similarities between image and text tokens of vision-language models. Yet, they only perform well on single-object images [17,18,20] or require cross-attention [21], which does not exist in CLIP ViTs [2]. On the other hand, applying well-known



(a) Input (b) Ours (c) ScoreCAM (d) RISE

Fig. 1: In a complex COCO scene, SotA feature importance methods often produce noisy heatmaps for CLIP RN50x16, questioning what objects are the most important to CLIP. Here, RISE [24] heatmap covers both suitcases and the "baggage" text (top row) while ScoreCAM [27] highlights both the racket and the tennis court (bottom row), yielding a poor 0.0 IoU between the highlighted region □ and the ground truth box □. By using only the top-300 channels of the highest gradients, we (1) localize *the most important* objects in a complex scene (e.g., here, racket at IoU of 0.649); and (2) produce a SotA zero-shot, open-vocabulary, object localization method for COCO and PartImageNet.

saliency methods [22,23,24] for convolutional networks (CNNs) to CLIP ResNets often yields noisy heatmaps (see Figs. 1 & 3) perhaps because CLIP neurons are highly multifaceted [16,25] and responsive to a wide variety of information in an image, including text [16,26]. We find ScoreCAM [27] to be the highest-accuracy prior art in CLIP-based, zero-shot object localization (Table 4) but also prohibitively slow. For an input image, ScoreCAM requires $\geq$ 3,072 forward passes through CLIP RN50x16 to compute the CLIP scores, i.e., the coefficients for linearly combining 3,072 layer4 channels into a heatmap. This overhead is even higher for CLIP RN50x64, which has 4,096 layer4 channels.

To overcome this problem, we propose gScoreCAM, a simple-yet-effective technique for reducing ScoreCAM's theoretical time complexity by 10$\times$ by using only the top $k = 300$ most important (i.e., highest-gradient) channels instead of all 3,072 layer4 channels of CLIP RN50x16. By using the 10% most important channels, gScoreCAM serves as (1) an interpretability technique for localizing the *most important* objects in a complex scene (Fig. 1; backpack); (2) a state-of-the-art (SotA) *zero*-shot, open-vocabulary, object localization method that does not rely on any CLIP finetuning or object detectors. We find that:[3]

1. Compared to the CAM-based family, which proposes alternative sets of coefficients for linearly combining all layer4 channels into a single heatmap, gScoreCAM is the most accurate, zero-shot object localization method for CLIP on 2017 COCO (Sec. 4).

[3] Code and an interactive demo are at https://github.com/anguyen8/gScoreCAM.

2. gScoreCAM performs comparably to ScoreCAM, the best prior art on zero-shot object detection using CLIP, but is ∼8× faster in practice (Sec. 5.1).
3. gScoreCAM is around 2 to 4× more accurate than the other methods (excluding ScoreCAM) on three different datasets: ImageNet [28], COCO [29], and PartImageNet [30] (Sec. 5.1). In particular, our method performs better in the harder cases, i.e., localizing an object in multi-object scenes or localizing a part of an object (Sec. 5.2).
4. For both RN50x4 and RN50x16 versions of CLIP, gScoreCAM is a SotA localization method that does not require finetuning CLIP or any specific training to object localization (Sec. 5.3).

2 Related Work

Visualizing multimodal networks As vision-language Transformers become increasingly more popular, many recent methods [17,18,19] were proposed to visualize the attention of these models or the similarity between image and text tokens [21]. Some methods [21] require cross-attention, which does not exist in both CLIP ViTs and ResNets because cross-modal attention does not allow caching of image or text embeddings and therefore admits a much slower retrieval speed in practice. Other interpretability methods that do not require cross-attention only perform well on single-object image crops [17,31,18,20]. That is, applying the SotA ViT feature importance method (i.e., HilaCAM [17]) on CLIP ViT-B/32 yields a localization accuracy 1.5× *worse* than when applying gScoreCAM on CLIP RN50x16 (12.82 vs. 20.83; Table 4).

Feature importance methods for CNNs As ViT visualization methods are either not applicable to or performing poorly on CLIP ViTs, an alternative technique for interpreting CLIP is applying feature importance methods for CNNs [32,22,23] to CLIP ResNet-50 models. These methods can be grouped into two categories: **(1) white-box** i.e., using gradients or activation maps (a.k.a. channels) or both to derive an attribution map e.g. [32,22,23,27]; and **(2) black-box**, i.e. relying on perturbation-based analysis to compute the attribution of each input feature [24,33,34]. In the white-box group, visualizing the image gradients alone often yields noisy heatmaps [35,36]. CAM-based methods linearly combine the channels at the last convolutional layer in CNNs into a single "attention" heatmap. Because CLIP ResNets do not have a global average pooling (GAP) layer followed by the last classification layer, the original CAM [32] is not applicable to CLIP ResNets. Instead, one needs to resort to other CAM-based methods that compute the channel coefficients differently, e.g. using gradients [22,23] or confidence scores [27]. Compared to the CAM-based family, our method is the SotA in CLIP-based zero-shot, object localization and uses both gradients and scores for linearly combining the channels. Like ScoreCAM [27], our gScoreCAM also uses CLIP scores generated for masked images and a prompt to compute a channel's importance weight; however, we apply ScoreCAM on only the top-10% channels, discarding the rest. RISE [24] is a black-box version of ScoreCAM where it generates a random mask instead of directly leveraging an activation

map as a mask. Our gScoreCAM outperforms RISE in both efficiency and localization accuracy.

3 Methods

We first describe the original Class Activation Map (CAM) method [32] and then ScoreCAM [27] before introducing our gScoreCAM, which extends ScoreCAM.

3.1 Revisiting CAM and ScoreCAM

CAM [32] is applied to all N channels $\{A_i\}^N$ at the last convolutional layer (e.g. layer4 at ResNet-50 [37]) that is followed by a GAP layer and then a linear 1000-output classification layer (whose weight matrix $\in \mathbb{R}^{N\times 1000}$). That is, for each ImageNet class $\boldsymbol{c}$, CAM uses the N corresponding weights $\{w_i^{\boldsymbol{c}}\}^N$ to linearly combine N channels to create a saliency map $M_{\text{CAM}}^{\boldsymbol{c}}$:

$$M_{\text{CAM}}^{\boldsymbol{c}} = \sum_i^N A_i \times softmax(w_i^{\boldsymbol{c}}) \tag{1}$$

ScoreCAM [27] is the same as CAM but uses the confidence scores of the CNN in place of the weights $w_i^{\boldsymbol{c}}$ in Eq. 1. Specifically, to explain why an input image $\boldsymbol{x}$ belongs to a target class $\boldsymbol{c}$ w.r.t. a CNN $f_{\boldsymbol{c}}(.)$, the ScoreCAM algorithm is:

1. Upsample all N channels at the last convolutional layer to the input-image size using bilinear interpolation, yielding a set of upsampled channels $\{A_i^{\text{up}}\}^N$, which serve as masks in the next step.
2. Element-wise multiply each mask A_i^{up} with all color channels of the input image $\boldsymbol{x}$ and feed the resultant (masked) images to the CNN $f_{\boldsymbol{c}}(.)$ to obtain an output confidence score corresponding to the target class $\boldsymbol{c}$.
3. Use the N confidence scores obtained in place of the $w_i^{\boldsymbol{c}}$ to compute $M_{\text{CAM}}^{\boldsymbol{c}}$ following Eq. 1.

In sum, the ScoreCAM saliency map is computed by:

$$M_{\text{ScoreCAM}}^{\boldsymbol{c}} = \sum_i^N A_i \times softmax(f_{\boldsymbol{c}}(\boldsymbol{x} \odot A_i^{\text{up}})) \tag{2}$$

where $\odot$ is the Hadamard product.

3.2 Proposed method: Gradient-guided ScoreCAM (gScoreCAM)

While performing fairly accurately with CLIP CNNs (Table 4), a major drawback of ScoreCAM is its prohibitively slow runtime as it requires many (e.g., N = 3,072) forward passes to generate a single saliency map when CLIP RN50x16 has N = 3,072 layer4 channels. A second problem is that since the convolutional channels inside CLIP CNNs tend to react to a diverse variety of details (including

both graphical and textual ones) present in the input image [16], using *all* the channels (as in CAM and ScoreCAM) often yields noisy heatmaps.

To address these two problems, we propose to apply ScoreCAM but only to the top-k (e.g., $k = 300$ i.e. only $\sim$10%) of the channels of the CLIP's image encoder. This effectively reduces the runtime by almost 10$\times$ and yields more object-focused heatmaps. We choose the top-k channels by ranking them using their mean gradients over three color channels, which we empirically find to be the best proxy for *importance* among common channel-ranking criteria (Sec. 4.1).

Our full algorithm for running gScoreCAM on CLIP is:

Algorithm 1 Explaining CLIP (for both ResNet- & ViT-based CLIP encoders)

Input: Input image $\boldsymbol{x} \in \mathbb{R}^{W \times H \times C}$, a text prompt P, and a target layer L (in CNNs, L is the last convolutional layer) whose activation maps are $\in \mathbb{R}^{W' \times H'}$.
Output: A heatmap $M \in \mathbb{R}^{W' \times H'}$.

1: Run 1 forward pass through CLIP $(\boldsymbol{x}, P)$ to get N channels $\{A_i\}^N$ at layer L
2: Run 1 backward pass to get the N channel gradients at layer L
3: Take the top-k largest-gradient channels and use them as masks (as in ScoreCAM) to generate k masked images, i.e. $\{\boldsymbol{x}_i^*\}_k$ where $k \ll N$ and $\boldsymbol{x}_i^* = \boldsymbol{x} \odot A_i^{\text{up}}$.
4: Run k forward passes through CLIP $(\boldsymbol{x}_i^*, P)$ to obtain k CLIP scores.
5: Use the k CLIP scores as the coefficients in place of w_i^c in Eq. 1 to linearly combine the k channels into a single heatmap (i.e., discarding all N-k other channels).

For example, in CLIP RN50x16, L is the last convolutional layer whose the output volume is $12 \times 12 \times 3,072$ (i.e. $W' = H' = 12$ and $N = 3,072$). For ViT-B/32, L is the penultimate layer of size $7 \times 7 \times 768$ (more details in Sec. 3.5).

3.3 Evaluation Datasets

To thoroughly understand our method's localization capability, we test it on three localization benchmarks of increasing granularity: (1) localizing the main object in a single-object image (ImageNet-v2 [38]); (2) localizing one object in multi-object scenes (2017 MS COCO [29]); and (3) localizing an object part in a single-object image (PartImageNet [30]).

ImageNet-v2 is a re-make version of the original 1000-class ImageNet [28] attempting to understand how well existing algorithms overfit to the common ImageNet and generalize to a new reproduction of it. We use all 10,000 images in **train-fullsup** set (a.k.a **val2** in the data preparation script) annotated by Choe et al. [39].

COCO is an 80-class dataset designed for object detection and segmentation. COCO images often include multiple objects of different sizes and locations, making object detection more complex than ImageNet-v2. Therefore, we expect a significant contrast between different methods in the object detection of COCO. We use all 50,000 images in **val2017** set for our evaluation.

PartImageNet is a variance of ImageNet, where the selected images are further labeled at the part level. That is, it has a hierarchy label. e.g., in the bird class, the bird will be further labeled to part level: head, body, wing, foot, and tail. It consists of 24,000 images with 11 super-class. The object detection task in PartImageNet is challenging; it requires the attribution method to capture precisely the model's attention for each part of the object. We use all 4,598 images in **test** set for evaluation.

3.4 Evaluation Metrics

To assess how well a saliency method localizes an object or an object part in an image, we perform **Z**ero-**S**hot object **D**etection (ZSD) on the three datasets described in Sec. 3.3. Specifically, we first binarize the saliency map, derive a predicted bounding box, and compare it with the ground truth bounding box under two common metrics: BoxAcc [40] and MaxBoxAccV2 [39].

Inferring bounding box from a heatmap[4] A common method for deriving the bounding box from a heatmap is to maximize task performance by grid searching for binarization threshold. An alternative approach, Otsu's method, is a non-task-specific binarization method, which tries to maximize foreground and background contrast after binarization. Following Chefer et al. [17], we use Otsu's method to binarize the heatmap in COCO and ParImageNet evaluation and use the grid search approach in ImageNet-v2, which is recommended by Choe et al. [39]. Note that the choice of binarization method is trivial to the ZSD results (see Sec. A2 for more details).

BoxAcc metric For COCO and PartImageNet, we use BoxAcc [40], which measures the following accuracy:

$$BoxAcc(\tau, \delta) := \frac{1}{M} \sum_{m} 1_{IoU(box(h,\tau)_m, B_m) \geq \delta} \tag{3}$$

Where $\tau \in [0, 1]$ is the binarization threshold, h is the saliency generated by the model, $box(h, \tau)$ is the tightest box around the largest-area connected component of the binarized saliency with threshold τ, B is the ground truth box, $m \in M$ is the box index, and δ is the IoU threshold, we use $\delta = 0.5$ in our experiments. Note that the binarization threshold τ will be determined by Otsu's method in our evaluation.

MaxBoxAccV2 metric MaxBoxAccV2 [39] is defined as:

$$MaxBoxAccV2(\delta) := \frac{1}{3} \sum_{\delta} \max_{\tau} \big(BoxAcc(\tau, \delta)\big) \tag{4}$$

where $\delta \in \{0.3, 0.5, 0.7\}$, $\tau \in [0 : 0.05 : 0.95]$ is the binarization threshold to binarize saliency maps for evaluation against the binary ground truth masks. We use the default hyperparameters from the authors.

[4] We provide a detailed description for Otsu-based bounding box inferencing in Sec. A1

Intersection over Union (IoU) and Area Under the Curve (AUC) We measure the IoU at threshold $\delta = 0.5$ (Sec. 5.2) and the AUC of the IoU over different binarization thresholds (Sec. 4.2) to better understand the differences between multiple visualization methods.
Computation resources We conduct our experiments on 8 Nvidia RTX 1080Ti and 3 Nvidia Tesla T4 GPUs.

3.5 CLIP Networks

Model & Methods We conduct all our experiments based on the CLIP [2] model. We use the pre-trained model **RN50x16** for all CNN-based methods (except for the experiment in Sec. 5.3, which also includes **RN50x4**) as it provides the best performance among the convolutional-based variances available. For comparison, we also apply the CNN-based saliency method to CLIP **ViT-B/32** by reshaping the embedding of the target layer.
Target layers for visualization Similar to the idea in the CAM-based family, for interpretability, in both ResNet and ViT versions of the image encoder, we choose the valid channel closest to the CLIP prediction layer:

- RN50x16 and RN50x4: We use relu3 of the last BottleNeck in layer4, which is the last layer of the image encoder in CLIP. RN50x16 has 3072 channels with spatial dimension 12×12. RN50x4 has 2560 channels with spatial dimension 9×9.
- ViT-B/32: We use the second-last ResidualAttentionBlock in VisionTransformer. The output dimension is: $50 \times 1 \times 768$, we exclude the [CLS] vector then reshape into 7×7×768 for CAM-based visualizations.

We choose the second-to-last layer in ViT-B/32 because the gradients in the last layer are zero except for the [CLS] vector, and, only the [CLS] in the last layer embedding is used for the final prediction.

The implementation of our attribution methods for CNNs is based on PyTorchCAM [41]. The CLIP models we used are from OpenAI [42]. Our HilaCAM implementation is from the code released by [17].
Model hyperparameters We list some key hyperparameters of the models in Table S2. More hyperparameters can be found in Table 19 of Radford et al. [2].
Prompts of CLIP For ImageNet-v2 and COCO, we directly use "`{class name}`" (without quotation marks) as the prompt. For PartImageNet, we use "`{class name} {part name}`" as the prompt.

4 Design of gScoreCAM

In this section, we first introduce two ablation studies (Sec. 4.1) to explain the choices of hyperparameter k and the dimension reduction technique of the gradients. Secondly, we study the weight quality of CAM-based methods by measuring the level of overlap between the target and the activation maps (Sec. 4.2). We

find that gScoreCAM has the best weight quality among the methods that directly weigh the activation maps (Table 3). Lastly, we measure the noise level of the weighted activation map with Total Variation [43] (Sec. 4.3). Note that the noise level can not directly reflect the performance of the object localization but instead indicates the *confidence* of the visualizing method.

4.1 Ablation Study of gScoreCAM

We first introduce why we set $k = 300$ in our proposed method by studying how the hyperparameter k affects Zero-Shot object Detection (ZSD) performance. We then compare the performance of ZSD using different pooling methods to rank the channels.

Effects on the number of channels From Table 1, we find that gScoreCAM reaches its peak performance as the number of channels increases to 500. We choose $k = 300$ to conduct most of our experiments since it performs similarly to $k = 500$ but only needs 60% of its run-time.

Table 1: $k = 300$ is the smallest number of channels that yield a high localization accuracy on ImageNet-v2 and COCO.

Number of channels	ImageNet-v2 [38] (MaxBoxAccV2)	COCO [29] (BoxAcc)
300 (random)	55.12	18.55
20	49.83	15.72
100	54.97	19.25
200	53.75	20.50
300	**56.61**	**20.83**
400	56.60	20.89
500	**57.38**	**20.89**
600	56.55	20.89

Table 2: Taking the average of the channel-wise gradients yields the highest localization accuracy when $k = 300$.

Pooling Method	ImageNet-v2 [38] (MaxBoxAccV2)	COCO [29] (BoxAcc)
Average	**56.61**	**20.83**
Max-abs	56.46	20.62
Average-abs	54.57	20.35

Effects on the choice of gradient dimension reduction To use the gradients to guide us in choosing important channels, we first reduce the dimensions of the gradients from $\mathbb{R}^{c\times w\times h}$ to $\mathbb{R}^c$. Here, we study some of the most common methods, Average-Pooling, Average-Pooling over the **absolute** of the gradients, and Max-Pooling over the **absolute** of the gradients[5]. As shown in Table 2, the best method is simple Average-Pooling. This result coincides interestingly with the choice of GradCAM [22].

4.2 gScoreCAM Is the Best Weighting System among the Candidates

Experiment We design an experiment to measure the quality of the weighting systems by measuring the level of overlap of the weighted activation maps and

[5] Method with **-abs** means operate over absolute value of the gradients.

the ground truth. Precisely, we first measure the AUC of the IoU over different binary thresholds (e.g., $\tau \in [0.0 : 0.05 : 0.95]$) for each activation map that CAM-based methods use. We then compute the weighted sum of the corresponding AUC with the weights given by the method. Finally, we average the weighted AUC of all testing samples. Note that in some methods (GradCAM, xGradCAM, GradCAM++), in which the weights are not summed to one, we first discard the negative values and divide the remaining by its sum. This measurement will provide us with the upper bound of the mean weighted AUC. We set a baseline by assuming that all activation maps have equal weights.

Results We find that the gScoreCAM weighting is 2× better than the upper bound of GradCAM and slightly better than ScoreCAM (Table 3). The upper bound of the mean weighted AUC of GradCAM (in ImageNet-v2) is similar to the baseline, which explains why the ZSD performance of GradCAM is worse than the Gaussian noise baseline in [39].

Interestingly, we find a large gap between the IoU score of a random channel and the IoU score of the best channel (0.145 vs. 0.76). That is, a random channel may often not capture the content of the target class. It indicates that choosing the correct channels plays a vital role in visualizing the model's decision.

Table 3: We directly evaluate the weighting of different CAM-based methods and find that gScoreCAM is the best among them. The uniform baseline simply averages all the activation maps. The total variation of the heatmap provides information about how noisy the heatmap is. The lower total variation means that the heatmap tends to be less noisy.

	Mean weighted AUC	Mean Total Variation
Baseline	0.0380	1745 ± 268
GradCAM [22]	0.0390	885 ± 484
xGradCAM [44]	0.0421	1090 ± 657
GradCAM++ [23]	0.0357	1500 ± 458
ScoreCAM [27]	0.0881	1363 ± 391
gScoreCAM (ours)	**0.0936**	1301 ± 422

4.3 gScoreCAM Is Less Noisy Compared to ScoreCAM

Experiment We compute the mean total variation of the heatmaps generated by different methods.

Results The total variation provides a statistical view of the noise level of the resulting heatmap from different methods. We find that gScoreCAM is statistically less noisy than ScoreCAM in Table 3. An interesting result is that GradCAM is the least noisy method. Although we want the resulting heatmap to be less noisy, the noise level itself can not guarantee better localization performance.

Why GradCAM is the least noisy method? As discussed above, GradCAM provides a less noisy, lower coverage heatmap than other methods. We study the weighted activation maps of GradCAM over 4,000 random samples and find that the average resulting heatmap is entirely negative. Statistically, about 47% of the weights in GradCAM are negative. On the other hand, all the weights of gScoreCAM and ScoreCAM are positive, leading to higher total variation and a noisier heatmap. Note that the activation maps are after ReLU; therefore, all the activation maps are positive.

5 Experiments & Results

To directly compare performance between different methods, we use Zero-Shot object Detection (ZSD) to evaluate the heatmaps generated by different methods (Sec. 5.1). This evaluation provides information on how accurate the heatmap is with respect to the object. Our experiments find that gScoreCAM is the best method in COCO and PartImageNet (see Table 4 for details). To better understand the ZSD results, we further study how they perform in different scenarios, e.g., different object sizes and the number of objects in the test image (Sec. 5.2). Lastly, we conduct the same experiment as in Sec. 5.1 but for a different model (RN50x4), which reveals that our method is better than others without extra hyperparameter tuning (Sec. 5.3).

5.1 Zero-Shot Object Detection Results

Since CAM-based methods aim to detect the corresponding area of a given class or prompt, measuring the performance of ZSD will be a direct measurement of visualization results.

Table 4: CLIP zero-shot object detection results with different CAM variances. ScoreCAM and gScoreCAM are similar overall, while gScoreCAM is faster. The gScoreCAM on RN-50x16 is substantially better than HilaCAM [31], a state-of-the-art method for CLIP ViT-B/32. In contrast, gScoreCAM performs on par with HilaCAM on ViT-B/32. The results are for CLIP RN50x16 unless noted (ViT-B/32).

	ImageNet-v2 [38] (MaxBoxAccV2)	COCO [29] (BoxAcc)	PartImageNet [30] (BoxAcc)	Run time (s)	Number of Forward passes	Number of Backward passes
GradCAM [22]	38.90	11.59	10.91	0.21	1	1
xGradCAM [44]	24.24	5.60	2.93	0.24	1	1
GradCAM++ [23]	44.15	9.68	6.57	0.39	1	1
LayerCAM [45]	43.70	9.19	12.42	0.82	1	1
GroupCAM [46]	50.85	13.06	6.16	1.99	96	1
RISE [24]	41.39	7.26	8.69	166.57	8001	0
HilaCAM [17] (ViT-B/32)	47.79	12.82	11.80	0.26	1	1
gScoreCAM (ViT-B/32)	45.26	12.73	10.67	0.84	301	1
ScoreCAM [27]	**57.78**	20.43	15.76	55.75	3073	0
gScoreCAM (ours)	56.61	**20.83**	**16.34**	7.40	301	1

ScoreCAM and gScoreCAM are the best methods among the tests ScoreCAM and gScoreCAM outperform other methods in the object localization tests (Table 4). In particular, they are about 1.5 to 4× better in COCO and 1.2 to 2× better in PartImageNet compared to other methods.

gScoreCAM runs 8 times faster than ScoreCAM Since the computation of the visualization methods is model-dependent, we measure the computation overhead by its approximate elapsed time and the forward and backward passes required by these methods. We measure the average run-time (in seconds) for each image-prompt pair under different CAM methods and report in Table 4. The average run-time is measured by averaging the elapsed time of 200 samples on a single Nvidia 1080Ti GPU.

The required forward passes of ScoreCAM are 10× more than gScoreCAM, which means the run-time is 10× longer in theory. In our approximate experiments, the actual run-time of gScoreCAM is about 8× less than ScoreCAM.

gScoreCAM performs better on complex tasks The object localization task on ImageNet-v2 is relatively "simple" because most test images are object-centric, and a center-gaussian baseline reaches 52.5% accuracy, as reported in [39]. However, the same tasks on COCO and PartImageNet are much harder due to the variety of target sizes, shapes, and locations. Interestingly, gScoreCAM performs better on these more complex tasks compared to ScoreCAM.

Apply gScoreCAM to ViT-based CLIP Although gScoreCAM is designed for the CNN-based model, we also apply it to the ViT-based model by reshaping the embedding as discussed in Sec. 3.5. As shown in Table 4, we achieve a similar performance as HilaCAM, which is the state-of-the-art method in ViT visualization. One interesting note is that HilaCAM uses only attention in generating heatmaps, and our method uses only activation. Despite the enormous difference in the approaches, both techniques end up showing similar results.

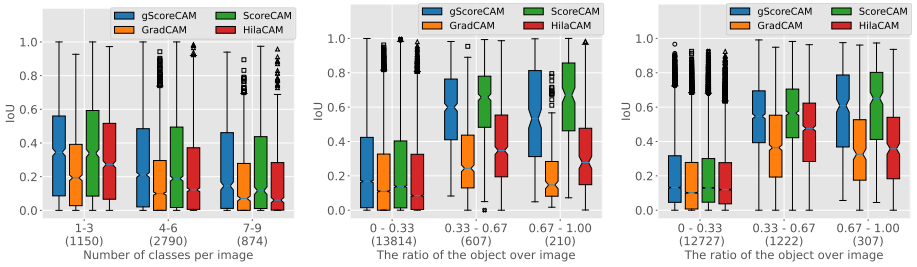
5.2 Why Does gScoreCAM Perform Better in COCO and PartImageNet?

Table 4 shows that gScoreCAM is slightly worse in ImageNet-v2 but better in COCO and PartImageNet than ScoreCAM. We conduct two sets of controlled experiments to find out why gScoreCAM is better in these two datasets.

5.2.1 gScoreCAM Performs Better in Multi-object Scenes

Experiment We conduct a controlled experiment based on the number of objects in an image. Specifically, we measure the average IoU of different methods when the number of classes varies. For diversity, we select images that only have one instance per class. This experiment uses a union of the COCO and LVIS labels because it provides more labels for each image. We measure the mean IoU because it directly measures the level of overlap with the object. Based on the number of classes in the images, we split the test images into three groups: (1-3, 4-6, 7-9) classes with (1150, 2790, 874) samples correspondingly.

Results We find that as the number of classes per image increases, the IoU of all methods decreases. gScoreCAM has the lowest IoU drop, resulting in better



(a) Controlled experiment results of object number on COCO.

(b) Controlled experiment results of object ratio on COCO.

(c) Controlled experiment results of object ratio on PartImageNet.

Fig. 2: Controlled experiments on COCO and PartImageNet. Fig. 2a shows how IoU changes with different methods when the number of classes per image is different. Figs. 2b and 2c show how the object ratio affects the methods' IoU. The number in parenthesis of x-axis on each plot is the number of samples in that group. In sum, **gScoreCAM is more accurate than other methods when a scene contains more objects (a) and object size (measured as the ratio between the object size and the image size) is smaller (b–c)**.

performance when the number of classes per image is greater than or equal to 4. The median IoU of gScoreCAM is approximately 0.03 higher than ScoreCAM and 0.07 to 0.10 higher than GradCAM and HilaCAM, as shown in Fig. 2a. This advantage makes gScoreCAM performs better in COCO because about 61% of the COCO validation images have more than three objects.

5.2.2 gScoreCAM Can Better Localize Small Objects

Experiment Similar to Sec. 5.2.1, we conduct another controlled experiment on COCO and PartImagenet based on the size of the target part or object. This experiment divides the test samples into three groups according to the target ratio. The target ratio is measured by the area of the target part or object over the full image. We divide the images into three groups: small (0-0.33), medium (0.33-0.67), and large (0.67-1). COCO and PartImageNet have samples (13811, 607, 210) and (12727, 1222, 307) in the corresponding group.

Results As Figs. 2b and 2c show that gScoreCAM consistently has a higher IoU in the small object groups, while ScoreCAM always has a higher IoU in the large groups. Combined with the results in Sec. 4.3, we find that ScoreCAM tends to generate a large and possibly noisy heatmap, resulting in better performance in the ZSD task when the object is large. On the other hand, gScoreCAM can better locate small objects. It is a critical capability in interpretability since we want the resulting heatmap to be as accurate as possible. Interestingly, when the target is a large scene (e.g., road, sea), gScoreCAM is still the best method (see Sec. A4 for details).

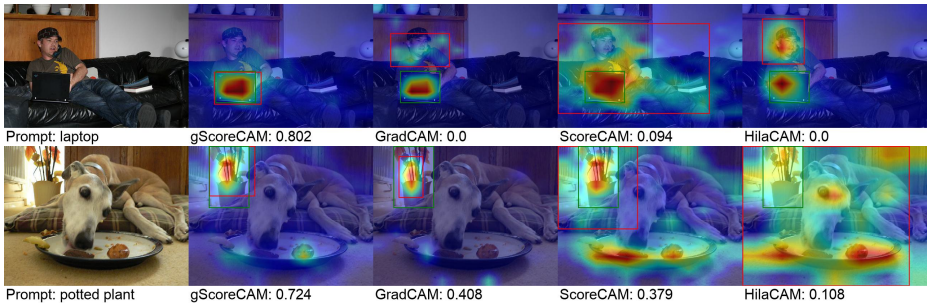


Fig. 3: In complex scenes (i.e. not ImageNet-v2), gScoreCAM outperforms other methods, yielding more precise localization and cleaner heatmaps. IoU scores between the ground truth (□) and inferred box (□) are shown next to each method name. More examples in Figs. S2 and S3.

5.3 gScoreCAM Performs Better on Different CLIP Models

We repeat the ZSD experiments in Sec. 5.1 for RN50x4 to confirm that our proposed method can generally be applied to different CLIP variations. For generality (i.e., without further hyperparameter tuning), we use the same hyperparameter $k = 300$ as in Sec. 5.1.

Results Our proposed method has the best ZSD performance, as shown in Table 5. The accuracy of gScoreCAM is around 2 to 4× higher than other methods (except ScoreCAM). It suggests that gScoreCAM can be applied to other CLIP variations without hyperparameter tuning.

Table 5: For both CLIP RN50x16 and RN50x4, gScoreCAM has the best overall ZSD performance in the CAM-based family.

	ImageNet-v2 [38] (MaxBoxAccV2)		COCO [29] (BoxAcc)		PartImageNet [30] (BoxAcc)	
	RN50x16	RN50x4	RN50x16	RN50x4	RN50x16	RN50x4
GradCAM [22]	38.9	32.61	11.59	9.86	10.91	9.60
xGradCAM [44]	24.24	18.94	5.6	6.11	6.57	5.01
GradCAM++ [23]	44.15	46.23	9.68	10.68	2.93	8.00
LayerCAM [45]	43.7	47.01	9.19	9.87	12.42	13.36
GroupCAM [46]	50.85	22.77	13.06	1.29	6.16	3.01
ScoreCAM [27]	**57.78**	57.99	20.43	21.31	15.67	15.39
gScoreCAM (ours)	56.61	**58.76**	**20.83**	**22.17**	**16.34**	**16.22**

5.4 Qualitative Study via CLIP

We first study a progressing plot that shows how the heatmaps change with the number of channels used by gScoreCAM. We then visually study the CLIP

heatmap from different methods. We find that our proposed method provides a more accessible model explanation from the visual studies.

The heatmap is getting noisier as the number of channels used by gScoreCAM increasing Fig. 1 shows a progressing plot when the number of channels used by gScoreCAM increases from 300 to 3072 (ScoreCAM) and the heatmap generated by RISE (last column). We find a clear trend that the gradient-guided ranking successfully ranks the channels by the heatmaps' contribution to the target. This visualization further confirms the result in Sec. 4.1 that we only need the top k channels to localize the target.

Visual comparison of gScoreCAM to other methods It turns out that gScoreCAM can generate a very accurate heatmap when the target object is tiny, as shown in Fig. 3. But other methods result in noisier or incorrect heatmaps. These accurate heatmaps allow us to study what the model is looking at and can be a helpful tool for studying the model. See Figs. S4 to S10 for more examples.

6 Discussion & Conclusions

Limitations One major limitation of our proposed method is that although it is 10× faster than its predecessor; it is still not comparable to methods that do not require multiple forward passes. Our proposed method is CNN-based; it does not generalize well on popular transformer-based networks. One last thing is that our proposed method introduces a hyperparameter (k).

Fig. 4: While Goh et al. [16] reported that CLIP is easily fooled by *typographic attacks*, our gScoreCAM visualizations reveal interesting insights that CLIP indeed was able to distinguish the objects between apple, iPod and even the background. The misclassification was merely due to the fact that there are multiple objects in the scene (i.e., ill-posed, single-label, image classification task).

Conclusions In this paper, we propose gScoreCAM, a gradient-guided CAM method to visualize and explain multimodal models. Our design is generic such that it can be easily applied to visualize and explain other CNN-based networks. In a systematic analysis of different visualization methods, our method performs the best in explaining and visualizing CLIP. We also find that our method can solve a common problem, as shown in Fig. 4: the text in the image misleads CLIP while giving the prediction. To the best of our knowledge, our proposed gScoreCAM is the best method to visualize and explain current large CNN-based models like CLIP. Therefore, we believe that gScoreCAM can help the community better understand recent foundation models and make improvements.

References

1. Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., et al.: On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021)
2. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning, PMLR (2021) 8748–8763
3. yzhuoning: yzhuoning/awesome-clip: Awesome list for research on clip (contrastive language-image pre-training). https://github.com/yzhuoning/Awesome-CLIP (2022) (Accessed on 05/18/2022).
4. Patashnik, O., Wu, Z., Shechtman, E., Cohen-Or, D., Lischinski, D.: Styleclip: Text-driven manipulation of stylegan imagery. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. (2021) 2085–2094
5. nerdyrodent: nerdyrodent/vqgan-clip: Just playing with getting vqgan+clip running locally, rather than having to use colab. https://github.com/nerdyrodent/VQGAN-CLIP (2022) (Accessed on 05/18/2022).
6. Kim, G., Ye, J.C.: Diffusionclip: Text-guided image manipulation using diffusion models. arXiv preprint arXiv:2110.02711 (2021)
7. Luo, H., Ji, L., Zhong, M., Chen, Y., Lei, W., Duan, N., Li, T.: Clip4clip: An empirical study of clip for end to end video clip retrieval. arXiv preprint arXiv:2104.08860 (2021)
8. Lei, J., Li, L., Zhou, L., Gan, Z., Berg, T.L., Bansal, M., Liu, J.: Less is more: Clipbert for video-and-language learning via sparse sampling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 7331–7341
9. Song, H., Dong, L., Zhang, W.N., Liu, T., Wei, F.: Clip models are few-shot learners: Empirical studies on vqa and visual entailment. arXiv preprint arXiv:2203.07190 (2022)
10. Kwon, G., Ye, J.C.: Clipstyler: Image style transfer with a single text condition. arXiv preprint arXiv:2112.00374 (2021)
11. Vinker, Y., Pajouheshgar, E., Bo, J.Y., Bachmann, R.C., Bermano, A.H., Cohen-Or, D., Zamir, A., Shamir, A.: Clipasso: Semantically-aware object sketching. arXiv preprint arXiv:2202.05822 (2022)
12. Sheng, E., Chang, K.W., Natarajan, P., Peng, N.: The woman worked as a babysitter: On biases in language generation. arXiv preprint arXiv:1909.01326 (2019)
13. Verge, T.: What a machine learning tool that turns obama white can (and can't) tell us about ai bias - the verge. https://www.theverge.com/21298762/face-depixelizer-ai-machine-learning-tool-pulse-stylegan-obama-bias (2022) (Accessed on 05/19/2022).
14. Li, Q., Mai, L., Alcorn, M.A., Nguyen, A.: A cost-effective method for improving and re-purposing large, pre-trained gans by fine-tuning their class-embeddings. In: Proceedings of the Asian Conference on Computer Vision. (2020)
15. Phillips, P.J., Hahn, C.A., Fontana, P.C., Broniatowski, D.A., Przybocki, M.A.: Four principles of explainable artificial intelligence. Gaithersburg, Maryland (2020)
16. Goh, G., Cammarata, N., Voss, C., Carter, S., Petrov, M., Schubert, L., Radford, A., Olah, C.: Multimodal neurons in artificial neural networks. Distill **6** (2021) e30

17. Chefer, H., Gur, S., Wolf, L.: Generic attention-model explainability for interpreting bi-modal and encoder-decoder transformers. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). (2021) 397–406
18. Subramanian, S., Merrill, W., Darrell, T., Gardner, M., Singh, S., Rohrbach, A.: Reclip: A strong zero-shot baseline for referring expression comprehension. arXiv preprint arXiv:2204.05991 (2022)
19. Aflalo, E., Du, M., Tseng, S.Y., Liu, Y., Wu, C., Duan, N., Lal, V.: Vl-interpret: An interactive visualization tool for interpreting vision-language transformers. arXiv preprint arXiv:2203.17247 (2022)
20. vijishmadhavan: vijishmadhavan/crop-clip: Crop using clip. https://github.com/vijishmadhavan/Crop-CLIP (2022) (Accessed on 05/23/2022).
21. Kim, W., Son, B., Kim, I.: Vilt: Vision-and-language transformer without convolution or region supervision. In: International Conference on Machine Learning, PMLR (2021) 5583–5594
22. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Grad-cam: Visual explanations from deep networks via gradient-based localization. In: Proceedings of the IEEE international conference on computer vision. (2017) 618–626
23. Chattopadhay, A., Sarkar, A., Howlader, P., Balasubramanian, V.N.: Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In: 2018 IEEE winter conference on applications of computer vision (WACV), IEEE (2018) 839–847
24. Petsiuk, V., Das, A., Saenko, K.: Rise: Randomized input sampling for explanation of black-box models. arXiv preprint arXiv:1806.07421 (2018)
25. Nguyen, A., Yosinski, J., Clune, J.: Multifaceted feature visualization: Uncovering the different types of features learned by each neuron in deep neural networks. arXiv preprint arXiv:1602.03616 (2016)
26. Materzyńska, J., Torralba, A., Bau, D.: Disentangling visual and written concepts in clip. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2022) 16410–16419
27. Wang, H., Wang, Z., Du, M., Yang, F., Zhang, Z., Ding, S., Mardziel, P., Hu, X.: Score-cam: Score-weighted visual explanations for convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops. (2020) 24–25
28. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115** (2015) 211–252
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, Springer (2014) 740–755
30. He, J., Yang, S., Yang, S., Kortylewski, A., Yuan, X., Chen, J.N., Liu, S., Yang, C., Yuille, A.: Partimagenet: A large, high-quality dataset of parts. arXiv preprint arXiv:2112.00933 (2021)
31. Chefer, H., Gur, S., Wolf, L.: Transformer interpretability beyond attention visualization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2021) 782–791
32. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., Torralba, A.: Learning deep features for discriminative localization. In: CVPR. (2016)
33. Ribeiro, M.T., Singh, S., Guestrin, C.: " why should i trust you?" explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. (2016) 1135–1144

34. Agarwal, C., Nguyen, A.: Explaining image classifiers by removing input features using generative models. In: Proceedings of the Asian Conference on Computer Vision. (2020)
35. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. arXiv preprint arXiv:1312.6034 (2013)
36. Nourelahi, M., Kotthoff, L., Chen, P., Nguyen, A.: How explainable are adversarially-robust cnns? arXiv preprint arXiv:2205.13042 (2022)
37. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. (2016) 770–778
38. Recht, B., Roelofs, R., Schmidt, L., Shankar, V.: Do imagenet classifiers generalize to imagenet? In: International Conference on Machine Learning, PMLR (2019) 5389–5400
39. Choe, J., Oh, S.J., Lee, S., Chun, S., Akata, Z., Shim, H.: Evaluating weakly supervised object localization methods right. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. (2020) 3133–3142
40. Gupta, T., Vahdat, A., Chechik, G., Yang, X., Kautz, J., Hoiem, D.: Contrastive learning for weakly supervised phrase grounding. In: European Conference on Computer Vision, Springer (2020) 752–768
41. Gildenblat, J., contributors: Pytorch library for cam methods. https://github.com/jacobgil/pytorch-grad-cam (2021)
42. OpenAI: openai/clip: Contrastive language-image pretraining. https://github.com/openai/CLIP (2022) (Accessed on 07/06/2022).
43. Radin, L., Osher, S., Fatemi, E.: Non-linear total variation noise removal algorithm. Phys D **60** (1992) 259–268
44. Fu, R., Hu, Q., Dong, X., Guo, Y., Gao, Y., Li, B.: Axiom-based grad-cam: Towards accurate visualization and explanation of cnns. arXiv preprint arXiv:2008.02312 (2020)
45. Jiang, P.T., Zhang, C.B., Hou, Q., Cheng, M.M., Wei, Y.: Layercam: Exploring hierarchical class activation maps for localization. IEEE Transactions on Image Processing **30** (2021) 5875–5888
46. Zhang, Q., Rao, L., Yang, Y.: Group-cam: Group score-weighted visual explanations for deep convolutional networks. arXiv preprint arXiv:2103.13859 (2021)