Object-Focused Edge Detection

Scott Lee*
UC Berkeley
scott.lee.3898@berkeley.edu

Alan Rosenthal*
UC Berkeley
amrosenthal@berkeley.edu

Abstract

We describe a general method for altering general algorithms for edge detection in order to produce edge mappings that focus on one or few individual objects in an image. We leverage Class Activation Mappings (CAMs) [13] to infer the important parts of an image and emphasize the edges of the detected objects. Since the general framework allows one to use any pre-trained network for object detection and CAM generation, and works for any edge detection algorithm with a set of intermediate edges (e.g. Canny, HED), our method is highly robust and flexible. When given an image with one or few objects, our method results in higher quality edge mappings that highlight edges from objects present in the image, and de-emphasize edges that are part of the background (i.e. not part of an object).

1. Introduction

Edge detection is a fundamental problem in computer vision, with applications in object detection, 3-D reconstruction, autonomous vehicles, medical analysis, and numerous other fields. As such, it is a widely explored field, and there exist many edge detection algorithms for images; some popular ones include more traditional methods, such as the Canny edge detector [3], and more modern neural-net driven approaches, such as the Holistically-Nested edge detector (HED) [12].

While these methods are usually successful at detecting boundary edges, they often include unnecessary or undesirable edges from the background of an image. Moreover, such algorithms can neglect critical details about the object of focus, such as facial details and individual body parts.

In this paper, we propose a general framework for modifying existing edge detection algorithms in order to make them more suitable for producing better *object-focused edges*—enhancing edges that add important details and reducing edges that are not part of the main subject—in order to augment the boundary edges produced by the original algorithm. In order to do so, we employ Class Activation Mappings (CAMs) [13] to reweight an intermediate set of edges that are used in some edge detection algorithm. Intuitively, CAMs provide a sense of how ”important” a certain pixel is when classifying an object present in the image. Therefore, we can use CAMs to emphasize edges that are part of an object and reducing edges that are instead part of the background of an image, resulting in an edge mapping that focuses on objects contained in the image.

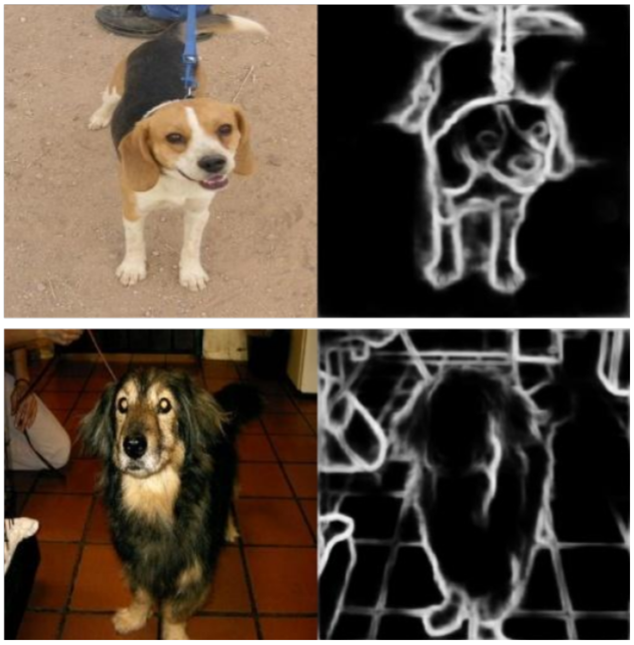
Figure 1. Above: Good object-focused edges. Below: Edges include too much background and not enough facial detail.

1.1. Related Work

Edge detection has its roots in early computational methods, such as the Sobel filter [7], the famous Canny edge detector [3], and zero-crossing feature methods [2]. Another set of edge detection methods relies on using statistics, inference, and information theory, such as the work of Martin *et al.* [9] and Statistical Edge Detection [8].

With the rise of neural nets, we have seen numerous diverse approaches at solving the classic problem,

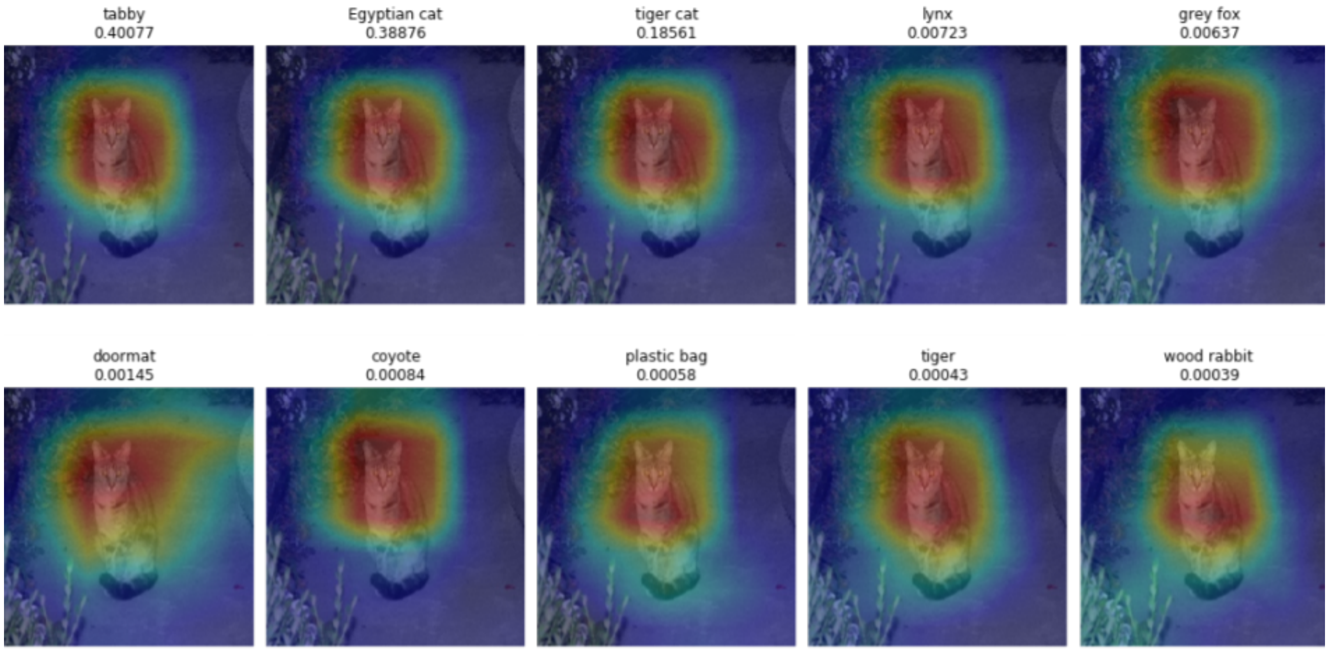

Figure 2. Class activation mappings corresponding to top 10 predicted classes of an image, with predicted class probabilities.

such as Boosted Edge Learning [4] and Structured Forests [5]. Finally, Convolutional Neural Nets (CNNs) have played a major role in recent computer vision developments, including edge detection. Some notable examples include DeepEdge [1], DeepContour [10], and Cost-Sensitive CNNs [6].

1.2. Class Activation Mappings

We leverage Class Activation Mappings (CAMs) [13] to indicate for which parts of an image are more relevant for classifying the content of the image. This method uses a CNN with a Global Average Pooling (GAP) final layer that gives the inputs for softmax to generate class probabilities.

Given an input image, let (x, y) be a spatial location in the image and let $f_k(x, y)$ denote the activation of unit k in the last convolutional layer. Let $F_k = \sum_{x,y} f_k(x, y)$ denote the output of GAP on unit k. For each output class c, let w_k^c be the weight of the linear mapping from unit k in the GAP layer to class c in the softmax input. Then the softmax input is $\sum_k w_k^c F_k$. Let the class activation map for class c be denoted M_c. For a spatial location (x, y), we have $M_c(x, y) = \sum_k w_k^c f_k(x, y)$. That is, the activation map for class c is the sum of the last convolution unit activations f_k, weighted by the "importances" w_k^c of units k for class c.

2. CAM-Weighted Edges

2.1. Core Algorithm

Our approach complements existing algorithms for edge detection. In this paper, we implement our modification on the Canny edge detector [3] and Holistically-Nested Edge Detection (HED) [12] algorithm with non-maximum suppression.

Using one CAM. Given an input image, the first step of our method is to feed the image into a CNN image classifier and obtain the the CAM corresponding to the highest-probability predicted class. Our implementation uses a DenseNet-161 trained on the ImageNet dataset. A vanilla edge detection algorithm, such as Canny or HED, is run on the input image up to an intermediate stage. An appropriate intermediate stage is one where the edges are represented in a "fuzzy" form with variation in intensity, thus capturing more information from the original image, before any thinning and non-maximum suppression is applied. For Canny, we use the gradient intensity representation of the image that is obtained after Gaussian blurring. For HED, we use the output of the algorithm before non-maximum suppression is applied. The CAM is upsampled if necessary to match the dimensions of the intermediate edge output, then the two are multiplied pixel-wise. The weighted edges outputted by this step have higher intensity in the regions where the classifier has highest activation, corresponding to discriminative features for the objects in the

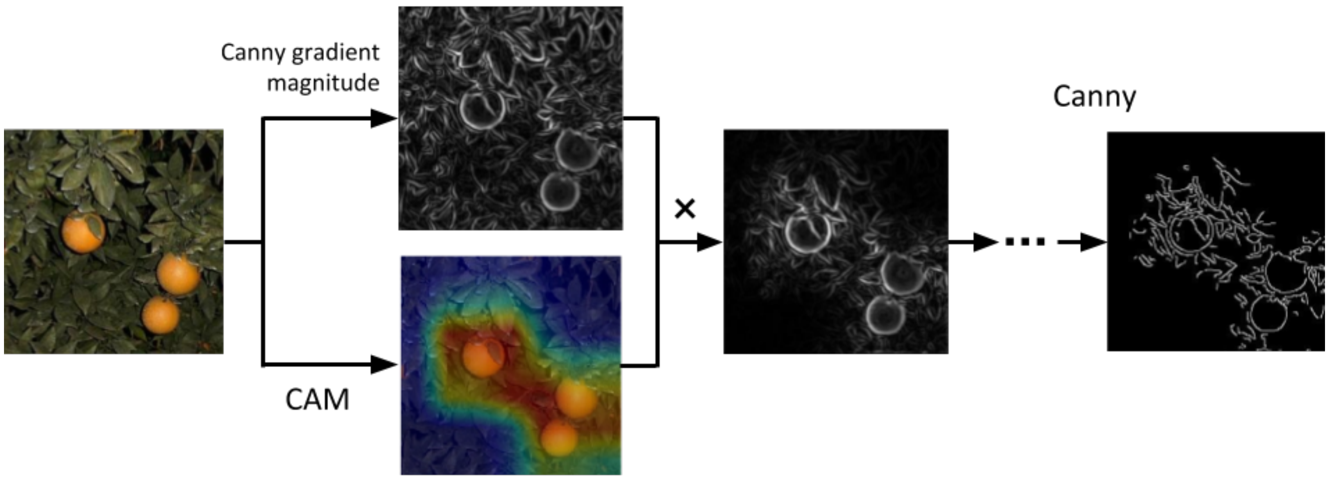

Figure 3. Diagram of intermediate steps, using the Canny edge detector as the base algorithm for modification.

image. The vanilla edge detection algorithm is resumed, using the weighted edges as input, to produce the final thinned and non-maximum suppressed edges.

Using multiple CAMs. The single CAM corresponding to the top predicted class tends to have high activation in only small parts of the image. While weighting with a single CAM produces high detail in these discriminative regions, it generally fails at capturing entire objects. To rectify this, multiple CAMs corresponding to the top k predicted classes may be combined to form an aggregate activation map covering more of the relevant objects, which is then multiplied with the intermediate edges pixel-wise as in the single-CAM case. The top k predicted classes' activation maps tend to focus on different parts of the image, but generally within the same prominent objects. Thus, a combination of these CAMs usually yields better coverage than a single one.

2.2. Tunable parameters

We will discuss several methods for combining the top k CAMs into one aggregate map. These may be used in conjunction with one another. Let $M_1, \dots, M_k$ be the CAMs for the top $1, \dots, k$ predicted classes. Let their predicted probabilities be $p_1 \geq \cdots \geq p_k$. For a spatial location (x, y), the activation for class c at that location is denoted $M_c(x, y)$.

Mean. We take the pixel-wise mean across the top k classes, so $M_{mean}(x, y) = \frac{1}{k} \sum_{c=1}^{k} M_c(x, y)$. This operation "blends" the different CAMs in a straightforward way, but is less effective in cases where many of the CAMs are very similar and only a few focus on different areas of an object.

Max. We take the pixel-wise maximum across the top k classes, so $M_{max}(x, y) = \max\{M_c(x, y) \mid c = 1, \dots, k\}$.

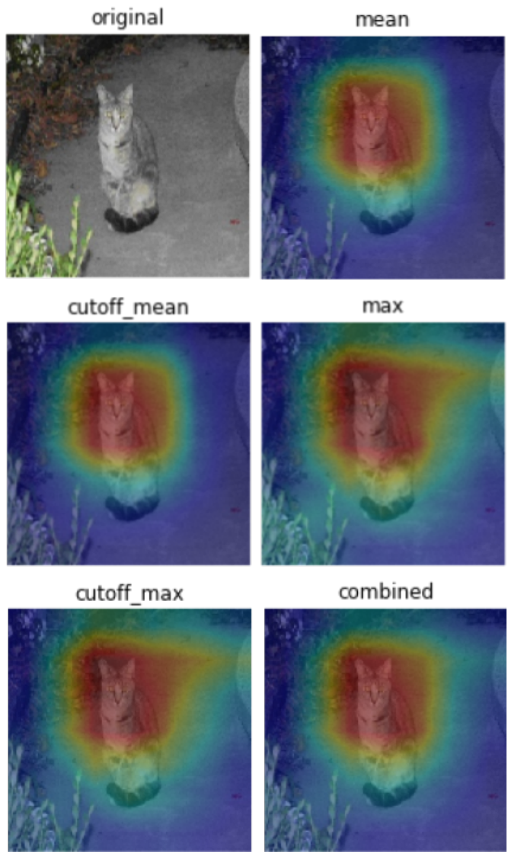

Figure 4. Different functions for aggregating the CAMs from Figure 2.

This method performs well at capturing the relevant object, since a pixel location receives a high weight if any CAM has high activation there. Conversely, it is sensitive to erro-

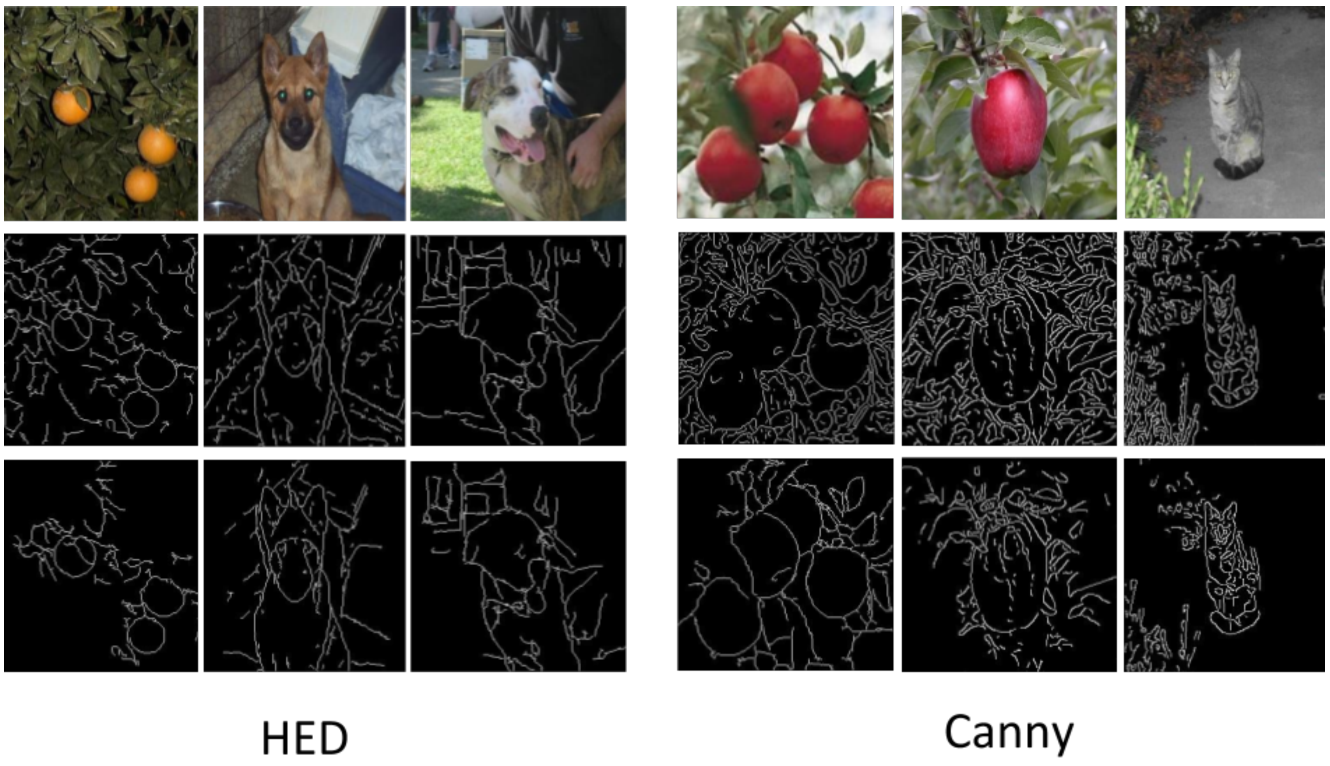

Figure 5. Results. Top row: original images. Middle row: Unmodified edge detector algorithm output. Bottom row: Modified output. For both HED and Canny, modified output includes less background clutter and at least as much detail within relevant objects compared to original output.

neous classification.

Under the reasonable assumption that high-activation regions of higher-probability classes are less likely erroneously focus on unrelated parts of the image, the class probabilities should be incorporated into the aggregate activation map. The following strategies address this issue.

Probability weights. Instead of applying a function like mean or max on the raw pixel values $M_c(x, y)$ of the CAMs, apply it on weighted pixel values $w_c M_c(x, y)$. These weights can be a monotonic function of the probabilities $w_c = f(p_c)$. If f is the identity, so the weights are the probabilities themselves, this scheme can often reduce the benefit of using multiple CAMs since the top class probability is frequently much larger than the rest.

Probability threshold. Instead of using the top k classes, use only the subset whose probabilities exceed some specified threshold. This prevents highly unlikely predictions from having undue influence on the final aggregated activation map.

For our results, we used the average of thresholded mean (with probability threshold = 0.0005) and max, with $k = 10$ classes.

Dilation. Dilation is another strategy to address the issue of CAMs not capturing enough of the image. For each iteration of dilation, each pixel location in the CAM is updated to the max value of its neighbors in a circular region around it. This technique can be applied to a single CAM or to the result of aggregating multiple CAMs as described previously. In some cases, when the activation region is focused only on the interior of an object, dilation produces a better activation map by "expanding" the high-weight area to include the whole object. In most cases, the aggregated CAM is sufficiently able to capture the entire object, and dilation causes background noise to be weighted highly near the border of the object; an negative example of how dilating the CAM includes unwanted background elements is seen in Figure 5. Because of this, we did not include dilation in our final results.

3. Results

We perform our algorithm using the Canny edge detector and the Holistically-Nested edge detector (HED). Our experiments involved trying different numbers of top-k class predictions ($k = 1, 5, 10$), using different functions of generated Class Activation Maps (min, max, average, combination, etc), and adjusting probability weights and cutoffs. Using a DenseNet pretrained on ImageNet as the CAM-

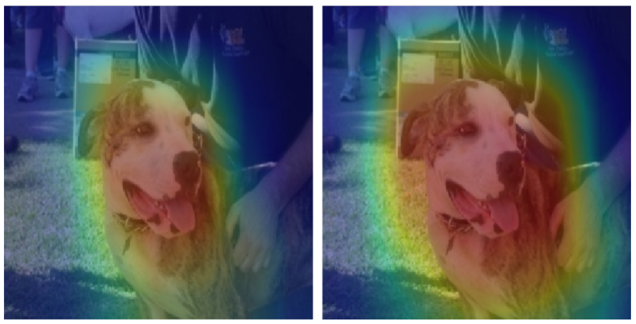

Figure 6. Left, no dilation; right, with dilation. While dilation captures more of the object boundary, it includes undesirable background regions.

generator backbone of our algorithm, we applied the CAM-weighted edge detector to images from a variety of datasets: Fruits360, Dogs vs. Cats, and ImageNet.

As illustrated in Figure 6, our edge detection algorithm produces edge mappings that capture details that are not present in the outputs of the base edge detection algorithms we use, such as fine details on the cat's and dog's noses and ears. Moreover, our method greatly reduces the amount of noise present in the output, yielding a much cleaner edge mapping that allows one to see more details.

Limitations. Because our algorithm depends on CAMs to determine which edges should be emphasized, it is critical that the weighted CAM used in reweighting edges is accurate. In images that do not have a specific focus, or in images that have numerous objects, such as the scenery photo present in Figure [TODO], we see that our algorithm does not provide meaningful improvements over the original edge detection algorithms. This is due to the fact that the top-k prediction CAMs cover most of the image, resulting in a near-uniform distribution of weights that has a very minute effect on the edges when they are reweighted.

Another more serious, but much rarer issue, is when the CAM prediction fails altogether. For the image of a cat in a box (Figure [TODO]), the DenseNet backbone used in CAM-generation gave completely incorrect top-k predictions (carton, cradle, binder, envelope, crate) because it detected the wrong object. As a result, our algorithm emphasized the edges of the wrong object, and the output is in fact worse than the results of the original edge detection algorithms.

4. Future Work

There are two major areas for future development and improvements to our presented algorithm.

Improving CAM Quality. Currently, our algorithm uses Class Activation Maps generated from a single iteration of classification (i.e. we feed our image to the

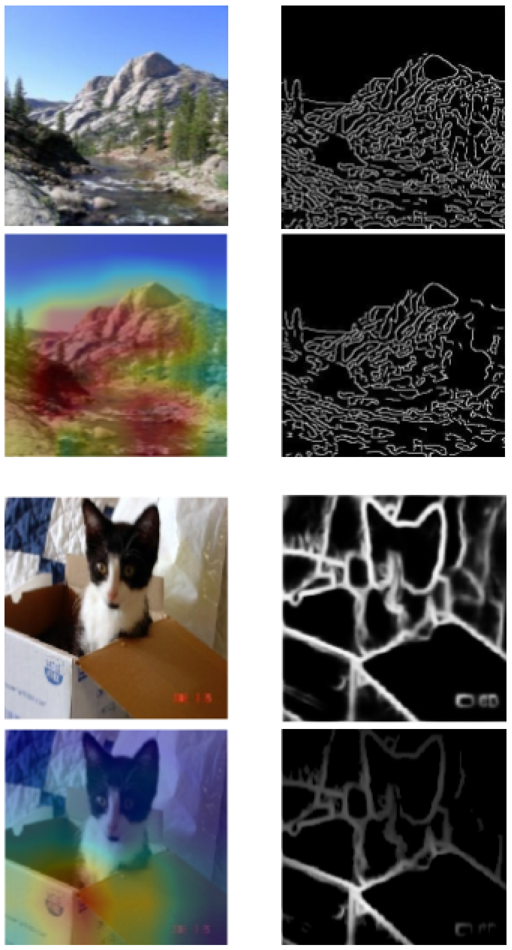

Figure 7. For the landscape images and cat images: top-left is image, top-right is original edge detector output, bottom-left is CAM, bottom-right is modified output. The CAM for the landscape is not very localized, so the modified algorithm fails to improve on the original significantly. For the cat, the classifier mislabeled the image as a box/carton.

CAM-generating backbone once, and use the top-k class CAMs to reweight edges). As we saw in previous examples (Figures 2, 4, 5), the resulting CAMs (and functions of CAMs) often cover the object of interest with an ambiguous circular filter, not binding tightly to the defining border of the object. One reasonable method to combat this issue is to hide random patches of the image when training the CAM-generating backbone so that the learner is forced to learn more robust representations for object recognition [11]. This ”Hide and Seek” method results in CAMs that touch multiple parts of the object instead of a single-center mapping, which yields a CAM that better approximates the boundary of the object, as we see in Figure 8. Furthermore, this augmentation only requires one to generate more samples from existing data, and requires no change to the inner workings of the edge detection algorithm at all.

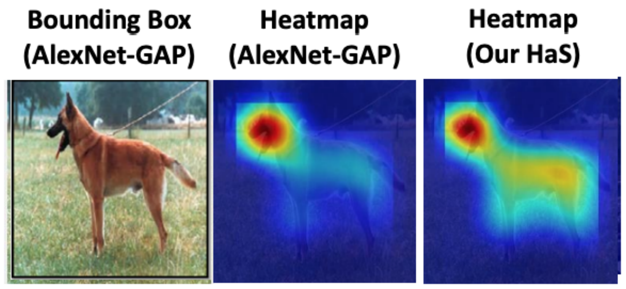

Figure 8. A classifier trained with the Hide-and-Seek (HaS) methodology [11] can produce CAMs that focus on a larger part of the object than a regular classifier.

Using segmentation. A possible evolution of

References

[1] G. Bertasius, J. Shi, and L. Torresani. Deepedge: A multi-scale bifurcated deep network for top-down contour detection. *CoRR*, abs/1412.1123, 2014.

[2] S. Bhardwaj and A. Mittal. A survey on various edge detector techniques. *Procedia Technology*, 4:220226, 12 2012.

[3] J. Canny. A computational approach to edge detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 1986.

[4] P. Dollár, Z. Tu, and S. Belongie. Supervised learning of edges and object boundaries. June 2006.

[5] P. Dollár and C. L. Zitnick. Fast edge detection using structured forests. *PAMI*, 2015.

[6] J.-J. Hwang and T.-L. Liu. Pixel-wise deep learning for contour detection. 04 2015.

[7] J. Kittler. On the accuracy of the sobel edge detector. *Image and Vision Computing*, 1(1):37 – 42, 1983.

[8] S. Konishi, A. L. Yuille, J. M. Coughlan, and S. C. Zhu. Statistical edge detection: Learning and evaluating edge cues. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(1), Jan. 2003.

[9] D. R. Martin, C. C. Fowlkes, and J. Malik. Learning to detect natural image boundaries using local brightness, color, and texture cues. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(5):530–549, May 2004.

[10] W. Shen, X. Wang, Y. Wang, X. Bai, and Z. Zhang. Deepcontour: A deep convolutional feature learned by positive-sharing loss for contour detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[11] K. K. Singh, H. Yu, A. Sarmasi, G. Pradeep, and Y. J. Lee. Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. 2018.

[12] S. "Xie and Z. Tu. Holistically-nested edge detection. In *Proceedings of IEEE International Conference on Computer Vision*, 2015.

[13] B. Zhou, A. Khosla, L. A., A. Oliva, and A. Torralba. Learning Deep Features for Discriminative Localization. *CVPR*, 2016.