Real-Time Image Saliency for Black Box Classifiers

Authors - Piotr Dabkowski and Yarin Gal
Submitted by Rishab Khantwal 180100095

Abstract:

In this work we implement a fast saliency detection method paper that can be applied to any differentiable image classifier. The main objective includes studying the paper and understand the method it applies. This also includes implementing the paper code and verifying results and suggest and prove any improvement that could be made.

Introduction:

In this paper, the authors develops a fast saliency detection method that can be applied to any differentiable image classifier. They train a masking model to manipulate the scores of the classifier by masking salient parts of the input image. This model generalizes well to unseen images and requires a single forward pass to perform saliency detection, therefore suitable for use in real-time systems. The author introoduces some new changes to the saliency objective and masking model.

Methodology:

Fighting the introduced evidence

Applying a mask M to image X to obtain the edited image E. In the simplest case, we can simply multiply X and M element-wise

$$E = X \odot M$$

The mask M, in the worst case, can be used to introduce a large amount of additional evidence by generating adversarial artifacts.

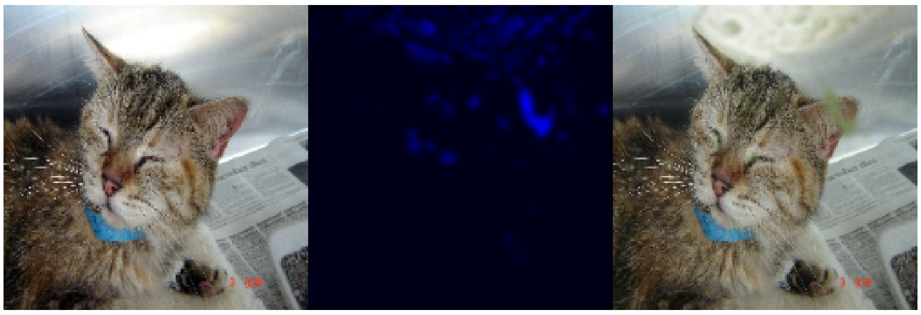

Figure 1: Cat image(left) when applied with Mask(middle) produced the image with an adversarial artifact which resulted in the model being unable to detect it as a cat.

We apply a mask to reduce the amount of unwanted evidence due to specifically-crafted masks:

$$E = X \odot M + A \odot (1 - M)$$

A can be chosen to be for example a highly blurred version of X. In such case mask M simply selectively adds blur to the image X and therefore it is much harder to generate high-frequency-high-evidence artifacts

Saliency metric

This paper also introduces a new saliency metric, it proposes to find the tightest rectangular crop that contains the entire salient region and to feed that rectangular region to the classifier to directly recognize the requested class We describe saliency matric as:

$$s(a, p) = \log(\tilde{a}) - \log(p)$$

Good saliency detectors will be able to significantly reduce the crop size without reducing the classification probability, and therefore a low value for the saliency metric is a characteristic of good saliency detectors.

Saliency objective

Given class c of interest, and an input image X, to find a saliency map M for class c, our objective function L is given by:

$$L(M) = \lambda_1 TV(M) + \lambda_2 AV(M) - \log(f_c(\Phi(X, M))) + \lambda_3 f_c(\Phi(X, 1 - M))^{\lambda_4}$$

where fc is a softmax probability of the class c of the black box image classifier

TV(M) is the total variation of the mask defined simply as:

$$TV(M) = \sum_{i,j} (M_{ij} - M_{ij+1})^2 + \sum_{i,j} (M_{ij} - M_{i+1j})^2,$$

AV(M) is the average of the mask elements, taking value between 0 and 1, and all λ_i are regularisers.

The function Φ removes the evidence from the image

$$\Phi(X, M) = X \odot M + A \odot (1 - M).$$

In total, the objective function is composed of 4 terms. The first term enforces mask smoothness, the second term encourages that the region is small. The third term makes sure that the classifier is able to recognise the selected class from the preserved region. Finally, the last term ensures that the probability of the selected class, after the salient region is removed, is low.

Masking Model

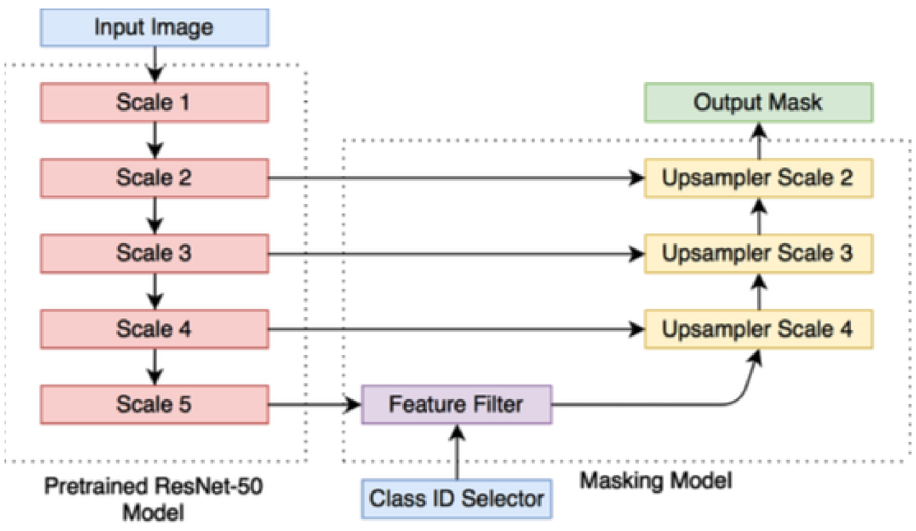

Figure 2: Architecture design of masking model

Architecture: In order to make our masks sharp and precise, we adapt a U-Net architecture so that the masking model can use feature maps from multiple resolutions. This is the diagram for our model. For the encoder part of the U-Net, we use ResNet-50 pre-trained on ImageNet.

Feature Filter: It would take X as input from the 5th layer of Resnet and class id and then would generate output given by:

$$Y_{ij} = X_{ij}\sigma(X_{ij}^T C_s)$$

Upsampler: The upsampler blocks take the lower resolution feature map as input and upsample it by a factor of two using transposed convolution, afterwards they concatenate the upsampled map with the corresponding feature map from ResNet and follow that with three bottleneck blocks.

Finally, to the output of the last upsampler block (Upsampler Scale 2) we apply 1x1 convolution to produce a feature map with just two channels — C0, C1. The mask Ms is obtained from:

$$M_s = \frac{\text{abs}(C_0)}{\text{abs}(C_0) + \text{abs}(C_1)}$$

The mask Ms has resolution four times lower than the input image and has to be upsampled by a factor of four with bilinear resize to obtain the final mask M.

Desired results

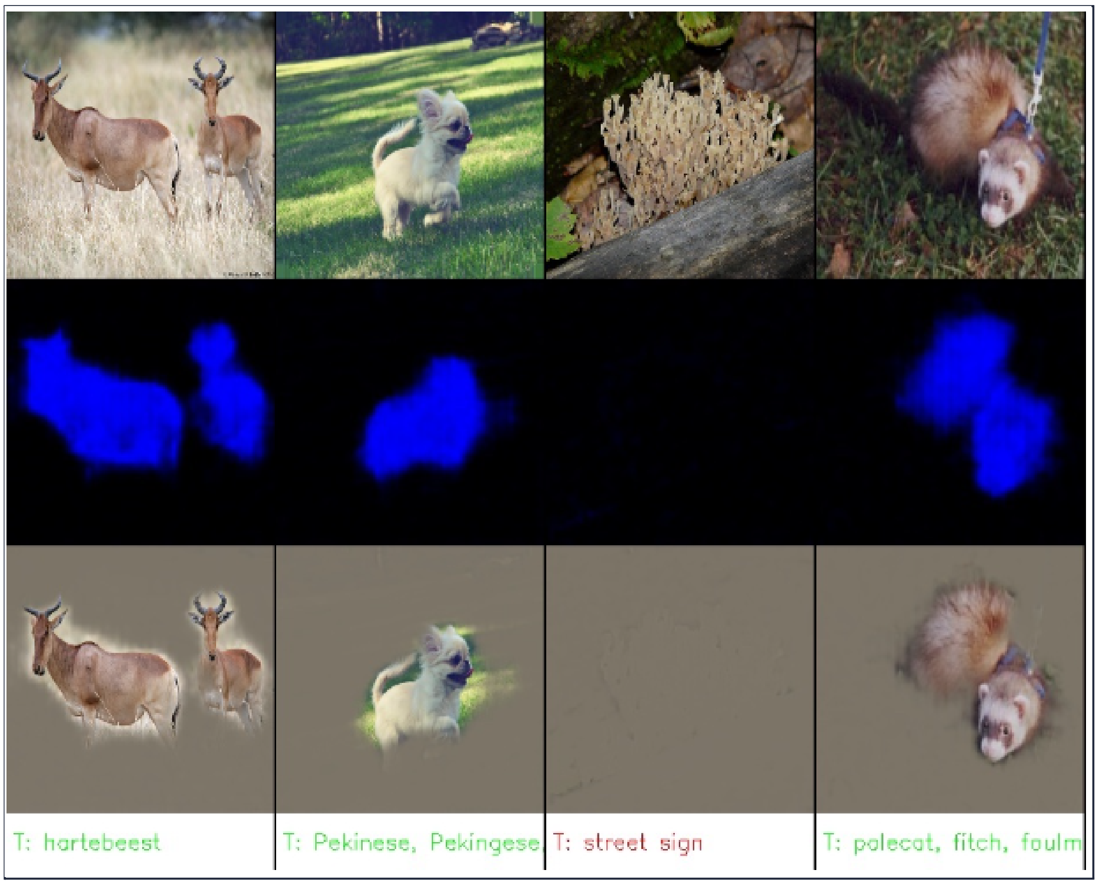

Figure 3: Image Net dataset examples

	Localisation Err (%)	Saliency Metric
Ground truth boxes (baseline)	0.00	0.284
Max box (baseline)	59.7	1.366
Center box (baseline)	46.3	0.645
Grad [12]	41.7	0.451
Exc [18]	39.0	0.415
Masking model (this work)	**36.9**	**0.318**

Figure 4: ImageNet localisation error and the saliency metric for GoogLeNet

Implementation:

Step 1: Deciding the training dataset and the black-box model.

I have chosen CIFAR10 (https://www.cs.toronto.edu/~kriz/cifar.html), The CIFAR-10 dataset consists of 60000 color images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images.

Step 2: Training the Black box model on the training set.

I have trained the Resnet50 on CIFAR10 taking CrossEntropy as a loss function for 6 epochs. I achieved a training loss of **0.064229** and an accuracy of **0.909544**.

Step 3: Developing the saliency model for the black-box model using the upsampler layer and feature filter function. Defining the loss function for the saliency model.

Step 4: Training the Saliency model on defined loss function keeping Black box model weights as frozen.

I trained the Saliency model for 6 epochs in steps 3 and 3. It achieved the training loss of **1.746939** at the end of 6 epochs.

Step 5: Visualize the results of the saliency model on the validation dataset.

Results and Discussions:

All the testing and training results are stored in the output and output_train folder

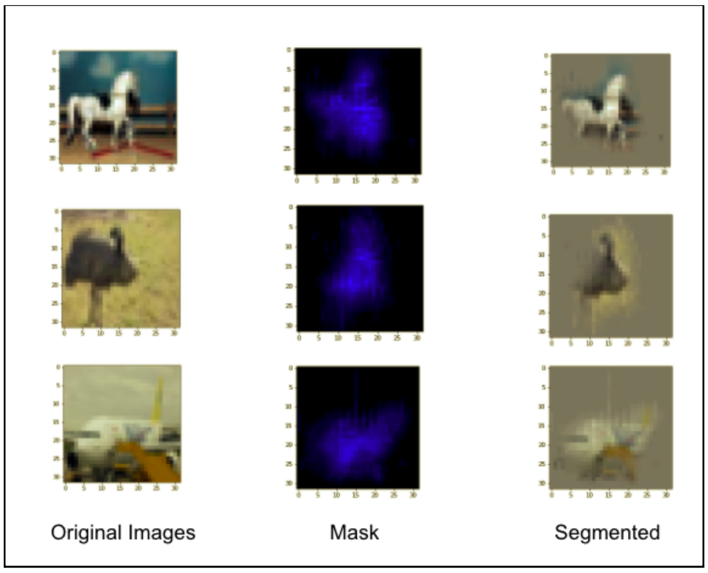

Figure 5: CIFAR10 examples of our implementation

- We tested our results on Nvidia K80 / T4 with 12GB Memory and a training time of 31 minutes for each epoch, taking a total time of **186 minutes to train**.
- The time for predicting mask **1000 images is 20.137 seconds**.
- Our model predicts at **fps of 49.657**

Conclusion:

This method for generating a saliency map is efficient and fast compared to the iterative saliency method. The method used is easy to generalize over different datasets. The model gives good results even after training on a single GPU for just 6 epochs. The training took about 3 hours for the training masking model.

However, for different black-box classifiers, different masking models would be needed to build, and hence generalizing over different black-box classifiers is tedious but possible.

Extra things tried:

- The original code was on python2, but since many functions have been depreciated in python2, we have written our code in python3.
- The original paper used Fitnet to train on the CIFAR10 dataset but we have used Resnet50 original model.

References:

The paper demonstrated
Real Time Image Saliency for Black Box Classifiers

The written code is inspired by the original source code
PiotrDabkowski/pytorch-saliency: Real-time image saliency 🌠 (NIPS 2017)

The code for the training of the Resnet model and masking model was written by me.
The code for defining the masking model is inspired by the main code file but was edited and modified.
The loss function code and Resnet code were directly taken from the source code.

Repository and Presentation link

My Github repository: https://github.com/rishav1122/CS689_RealtimeSaliency
My Presentation link: Rishab Khantwal presentation 2