Visual Explanations for Deep Neural Nets

Siddhant Midha
Sophomore, Electrical Engg.
IIT Bombay
siddhantm@iitb.ac.in

Prateek Garg
Sophomore, Electrical Engg.
IIT Bombay
prateekgarg@ee.iitb.ac.in

***Abstract*—With the evolution of Machine Learning and Deep Learning, we have come up with more and more complex networks which perform very well on various tasks. Through this development, one caveat remains. Increasing complexity of the network leads to decreasing interpretability. In this report, we review few such methods and implement them.**

***Index Terms*—Saliency, Occlusion Sensitivity, Class Activation Maps**

I. Introduction

There are two broad areas where an interpretation of the workings of neural networks proves useful - in the researcher's understanding, and as a proof of working to the stakeholders. We survey methods that facilitate the visualisation of the network, namely **Saliency Maps**, **Occlusion Sensitivity**, **Class Activation Maps** and **Deconvolution**. For some more insight, we also plot the learned **feature maps** at different locations in the network. The code can be found here.

II. Methods

A. Saliency Maps

Let us review the method used by the authors in the original paper [1]. Given a specific class, we query a Convolutional net about the spatial support of that particular class in the image. Formally, let I be the image and let $\mathfrak{S}$ be the image space. We have the vectorizing function

$$L : \mathfrak{S} \rightarrow \mathbb{R}^{k \times 1}$$

Thus $L(I)$ is the vectorized version of the image. With abuse of notation, let us denote this as I. Further for a class c we have the score function

$$S_c : \mathbb{R}^{k \times 1} \rightarrow \mathbb{R}$$

We consider the linear approximation,

$$S_c(I) \approx w_c^T I + b_c \text{ , where, } w_c^T \in \mathbb{R}^{k \times 1}, b_c \in \mathbb{R}$$

Elementary calculus tells us that,

$$w_c = \frac{\delta S_c}{\delta I}$$

Now the authors claim that this matrix gives us an idea of which spatial locations the network is looking at while predicting the class to be c. This is backed by the intuition that the larger the derivative for a particular pixel, more the output is sensitive to that pixel.

Computationally, we arrive at the map as follows. Assume the input shape is $m \times n$.

1) Firstly, the derivative vector is found by backpropagation.
2) Then, the vector is arranged in the same shape as the input image. Denote this $w_{i,j,c}$, where $i \in \{1, 2 \dots m\}, j \in \{1, 2 \dots n\}$ correspond to the x-y directions, and c to the channel.
3) For all $i \in \{1, 2 \dots m\}, j \in \{1, 2 \dots n\}$, define

$$\mathfrak{M}_{i,j} := max_c |w_{i,j,c}|$$

Then, $\mathfrak{M}$ is the desired saliency map.
Positive gradients corresponds to the location of target object and Negative image gradients corresponds to the other objects of competing classes. Since for the final output we use absolute value, saliency maps are not class discriminative which means it cannot localize the category in the image.

B. Occlusion Sensitivity Maps

To answer the question of whether the model is really identifying the object while classifying it, this [2] paper uses occlusion maps. This serves to be a coarser technique which does not require backpropagation at all. Here, we perform a systematic occlusion of the input image with an occluding object of some fixed size and monitor the output of the classifier. If and when some main features of the object being identified are occluded, the probability of classification of that object drops sharply. On the other hand, if a competing object is occluded partially or completely, probability of classification increases sharply.

More formally, let our input image be $I \in \mathbb{R}^{m \times n \times c}$. Further, let $p \times q$ be the dimensions of the occlusion. We describe occlusion at the $(j, k)^{th}$ position as

$$I[j : j + p - 1][k : k + q - 1][:] \leftarrow 0$$

Finally, we prepare a map showing the probability of classification as a function of position of the occluder. This process is justified from natural intuition and it shows that the visualization genuinely corresponds to the image structure that stimulates that feature map.

C. Class Activation Map (CAM)

A class activation map for a particular category indicates the discriminative image regions used by the CNN to identify that category. The authors of the paper [3] define this in the following way. For a given image, let $f_k(x,y)$ represent the activation of unit k in the last convolutional layer at spatial location (x,y). Then, for the unit k global average pooling is given as

$$F_k := \sum_{(x,y)} f_k(x,y)$$

Now say we have the final softmax layer. Then, for some class c, the input to the softmax S_c is

$$S_c := \sum_k w_k^c F_k$$

Intuitively, the importance of F_k in computing S_c is given by the weight w_k^c. We have,

$$S_c = \sum_k w_k^c \left(\sum_{(x,y)} f_k(x,y) \right) \quad (1)$$

$$S_c = \sum_{(x,y)} \sum_k w_k^c f_k(x,y) \quad (2)$$

Then we define $\mathfrak{K}_c$, the class activation map for class c as

$$\mathfrak{K}_c(x,y) := \sum_k w_k^c f_k(x,y)$$

Thus we claim that $\mathfrak{K}_c(x,y)$ indicates the importance of the activation at the spatial grid point (x,y) leading to the activation of class c. This is because

$$S_c = \sum_{(x,y)} \mathfrak{K}_c(x,y)$$

Thus, the class activation map is simply a weighted linear sum of the presence of these visual patterns at different spatial locations. By simply upsampling the class activation map to the size of the input image, we can identify the image regions most relevant to the particular category

III. Results

A. Saliency Maps

We trained a simple classifier to distinguish between dogs and wolves, using a [4] backbone network, on top of which we put a $2048 \rightarrow 1$ linear layer followed by a sigmoid. For this, we used the [8] dataset on Kaggle.
Let us take a look at an image and its corresponding saliency.

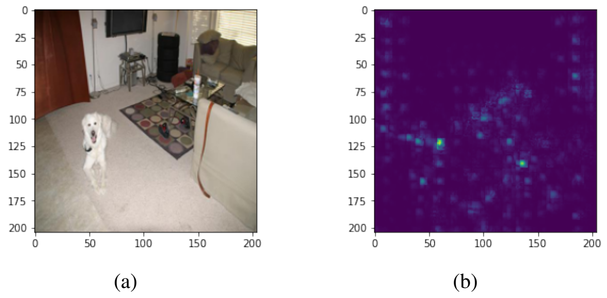

(a) (b)

Fig. 1. Image and its Saliency Map

Through the previous result, we suspect a dependence on the background. Now, we look at an image from the test set which was classified as a wolf.

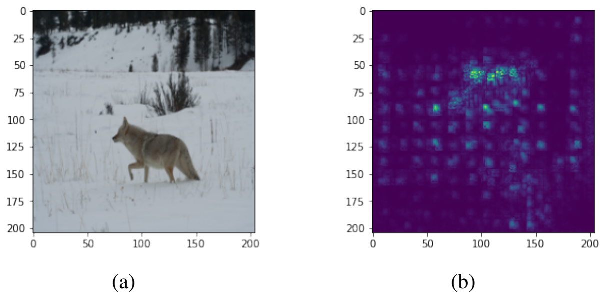

(a) (b)

Fig. 2. Wolf and its Saliency Map

We deliberately found an image of a dog with a white background, and fed it to the model to observe the following.

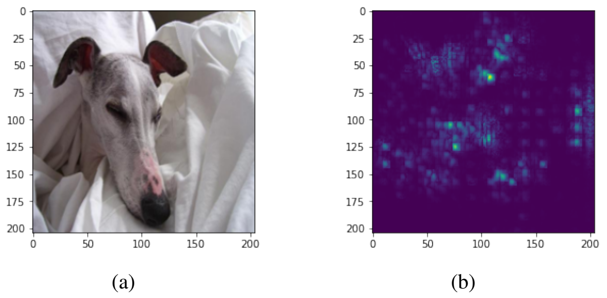

(a) (b)

Fig. 3. A dog classified as a wolf

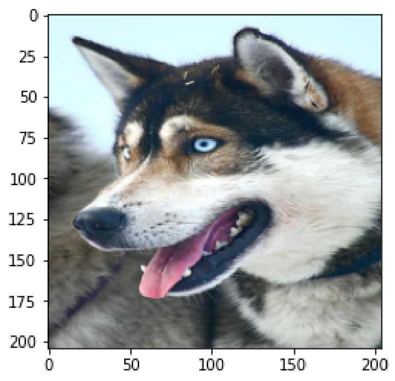
Thus, this technique enables us to comment on why certain classifications and particularly, some misclassifications are made.

B. Occlusion Sensitivity Maps

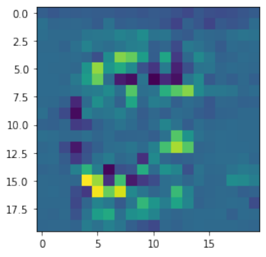
For testing occlusion sensitivity maps, we have used the model trained in the previous section. Results are given below. Note that the tuple in the caption is

$$(\text{Size of occluder}, \text{Stride of Occlusion})$$

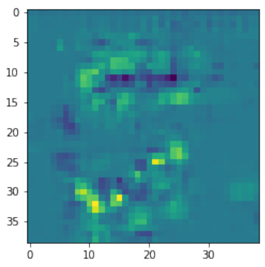
Let us begin with looking at which areas of the image fire for classifying the image below as a wolf.



(a)



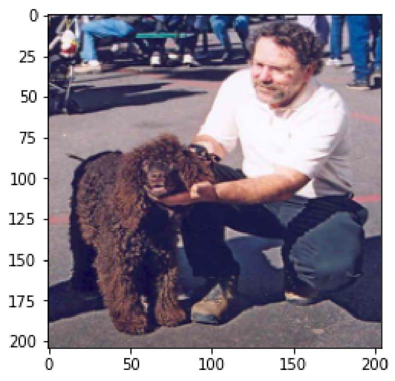
(b)



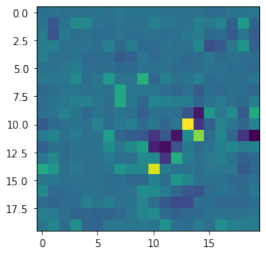
(c)

Fig. 4. Occlusion - (10,10) & (10,5)

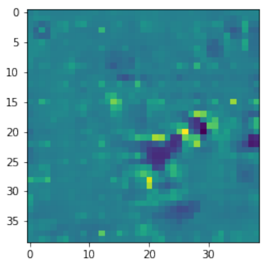
Let us look at another image of a dog with a man, and see which regions of the image indicate it is a dog.



(a)



(b)



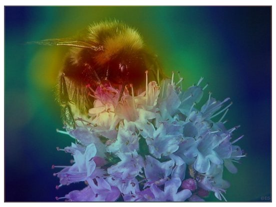
(c)

Fig. 5. Occlusion - (10,10) & (10,5)

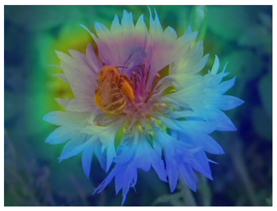
We can see that the model does not seem to be ”looking” at the dog, and this is backed by the fact that probability of classification as a dog in this case is 67.96%.

C. *Class Activation Maps*

We used pretrained network ResNet18 [4] from the torch library. Since it already has a Global Average Pooling Layer after convolutional Layer, we directly create a hook at the forward pass. The heatmap superposed with image is created using opencv. Other Networks which already have a GAP layer are Inception [5], SqueezeNet [6], and DensNet [7]. For the labels, we used [9] dataset, a subset of ImageNet dataset available on Kaggle.



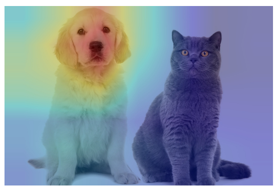
(a)



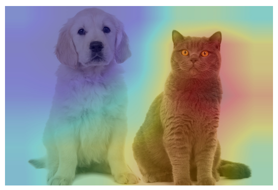
(b)

Fig. 6. Some Examples with images of Bees

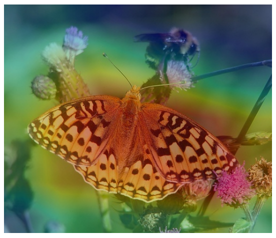
This method showed remarkable results even when the softmax output scores of competing classes were really skewed.



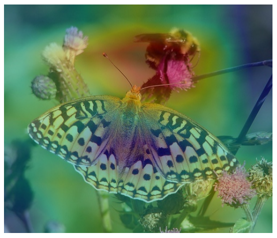
(a) Golden Retriever Score=0.715



(b) Egyptian Mau Score=0.0021



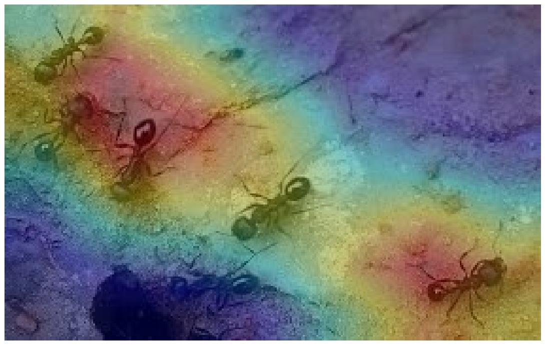
(c) Ringlet, Score=0.539



(d) Bee, Score=0.005

Fig. 7. CAM is class discriminative

In case of multiple instances of target object, the performance suffers. It is not able to localise well on each instance.


(a)

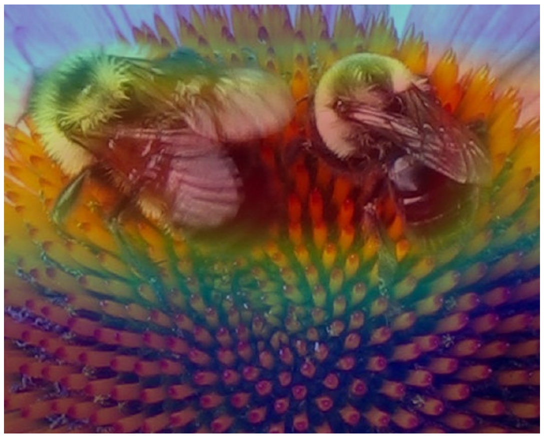

(b)

Fig. 8. The CNN can't localise well in case of multiple objects

CAM likely suffered from the use of Global Average Pooling to compute weights. This might be improved by using a conventional pooling layer.

ACKNOWLEDGMENT

We thank the `Analytics Club, IITB` for giving us the opportunity to work on this project, and we are grateful to our mentor Saikiran for his help.

REFERENCES

[1] Karen Simonyan, Andrea Vedaldi and Andrew Zisserman, "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps" arXiv:1312.6034v2 [cs.CV] 19 Apr 2014
[2] Matthew D. Zeiler and Rob Fergus, "Visualizing and Understanding Convolutional Networks" arXiv:1311.2901v3 [cs.CV] 28 Nov 2013
[3] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba "Learning Deep Features for Discriminative Localization" Computer Science and Artificial Intelligence Laboratory, MIT
[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, 'Deep Residual Learning for Image Recognition' arXiv:1512.03385 [cs.CV]
[5] Wei Liu Et. al, Going deeper with convolutions arXiv:1409.4842v1 [cs.CV]
[6] N. Iandola Et. al,'SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and ¡0.5MB model size' arXiv:1602.07360 [cs.CV]
[7] Huang Et. al,"Densely Connected Convolutional Networks" arXiv:1608.06993 [cs.CV]
[8] Kaggle Dogs vs Wolves Dataset
[9] ImageNet Large Scale Visual Recognition Challenge (ILSVRC)