Task Writeup

Himanchal Chandra
31/10/2020

What is Explainable AI?

Explainable AI are the set of methods and techniques in the application of artificial intelligence such that the result of the solution can be be understood by humans. It helps us to solve the problem of Black-Box in machine learning where it is very difficult to explain how a model has approached some specific decision.

Why Explainable AI?

Sometimes its not enough to just make blind predictions. You need to have some justification regarding why the model is predicting some specific decision. For example:

1. If a self-driving car makes a bad decision and kills a person, if we aren't able to quantify the reason for this bad decision, then we won't be able to rectify it, which could lead to even more disasters.

2. If some image recognition system is trained to detect a tumour in images and performs very well in terms of accuracy both on validation and test set. But when you present the results to the stakeholders they question from what parts of the image your model is learning or what is the main cause of this output and your most probable answer would be "I don't know" and no matter how perfect your model is, the stakeholders won't accept it because human life is at stake.

With the increased research in the field of Machine Learning especially Deep Learning various efforts are being made to solve the problem of interpretability and reach the stage of interpretable AI.

Although there are many interpretation methods, for keeping this writeup small and precise we will focus on 2 widely used methods namely:

1. Saliency Maps:

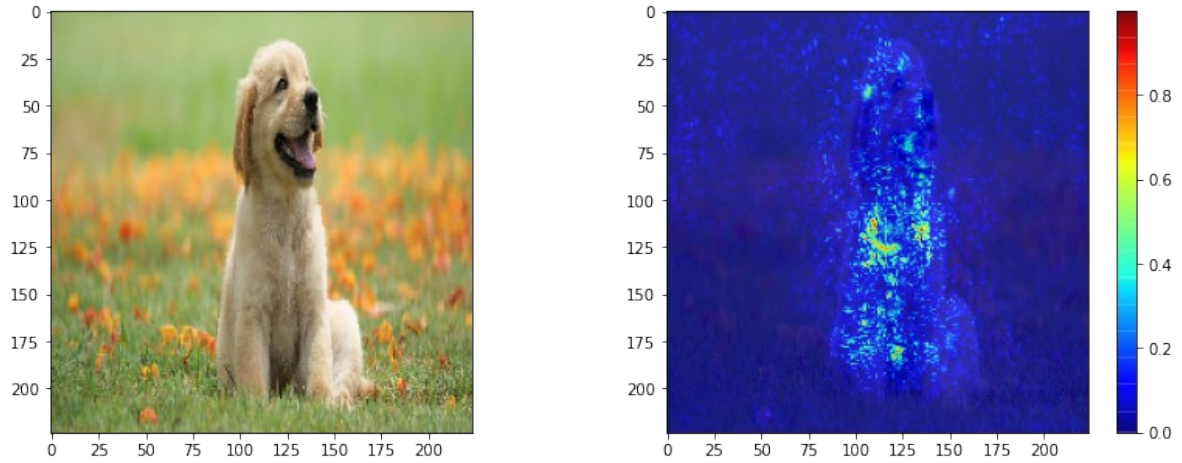
So what is Saliency, Suppose that all the training images of bird class contains a tree with leaves. How do we know whether the CNN is using bird-related pixels, as opposed to some other features such as the tree or leaves in the image? This actually happens more often than you think and you should be especially suspicious if you have a small training set.

So the idea is pretty straightforward, We compute the gradient of output category with respect to input image. This should tell us how output category value changes

with respect to a small change in input image pixels. All the positive values in the gradients tell us that a small change to that pixel will increase the output value.

∂Output/∂Input

This should tell us how the output value changes with respect to a small change in inputs. We can use these gradients to highlight input regions that cause the most change in the output. Intuitively this should highlight salient image regions that most contribute towards the output.



2. Gradient Class Activation Map (GRADCAM):

Gradient Class Activation Map (Grad-CAM) for a particular category indicates the discriminative image regions used by the CNN to identify that category.

GRAD-CAM utilizes the last convolutional layer feature maps beacause it retain spatial information (which is lost in fully-connected layers) and use it to visualize the decision made by CNN.

Some feature maps would be more important to make a decision on one class than others, so weights should depend on the class of interest.

$$L^{c}_{Grad-CAM} \sim= \sum_{k=1}^{K} \alpha^{c}_{k} A^{k} \in \mathbb{R}^{u \, x \, v}$$

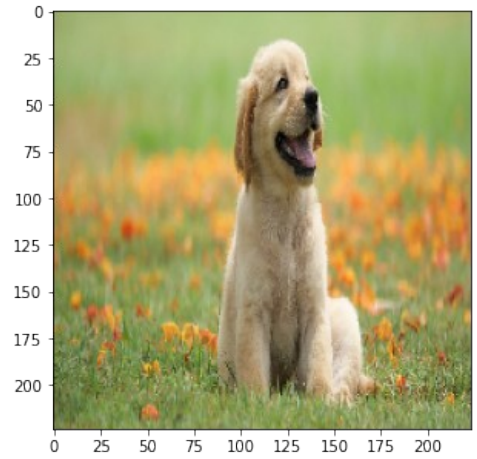
But what should be the weight α_{ck}?

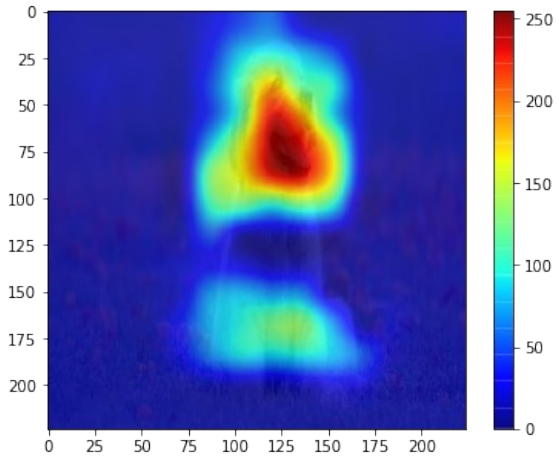
The gradient of the cth class score with respect to feature maps A_k measures linear

$$\frac{dy^c}{dA^k_{i,j}}$$

effect of (i,j)th pixel point in the kth feature map on the cth class score. Finally, Relu is applied to the linear combination of maps because we are only interested in the features that have a positive influence on the class of interest.

$$L^c_{Grad-CAM} = ReLU\left(\sum_{k=1}^{K} \alpha^c_k A^k\right)$$





Code: Training Part:

I used Google Colab for the coding part. I downloaded the Dog Vs Cat dataset from Kaggle and trained it on the VGG16 network (Transfer Learning) keeping the convolutional layers frozen.

I used 19000 images for training and 6000 images for testing and achieved almost 100 percent accuracy on the 8th epoch because of the virtue of transfer learning.

I used early stopping to stop training when the monitored metric which was Validation accuracy has stopped improving, and Model checkpoint to save the best weights.

Code: Visualization Part:

Saliency Maps: I used the Keras-vis toolkit for visualizing Saliency maps and Matplotlib for plotting.

GRAD-CAM: For visualizing grad-cam, I manually calculated the gradients with the help of Yumi’s tutorial on Grad-Cam.