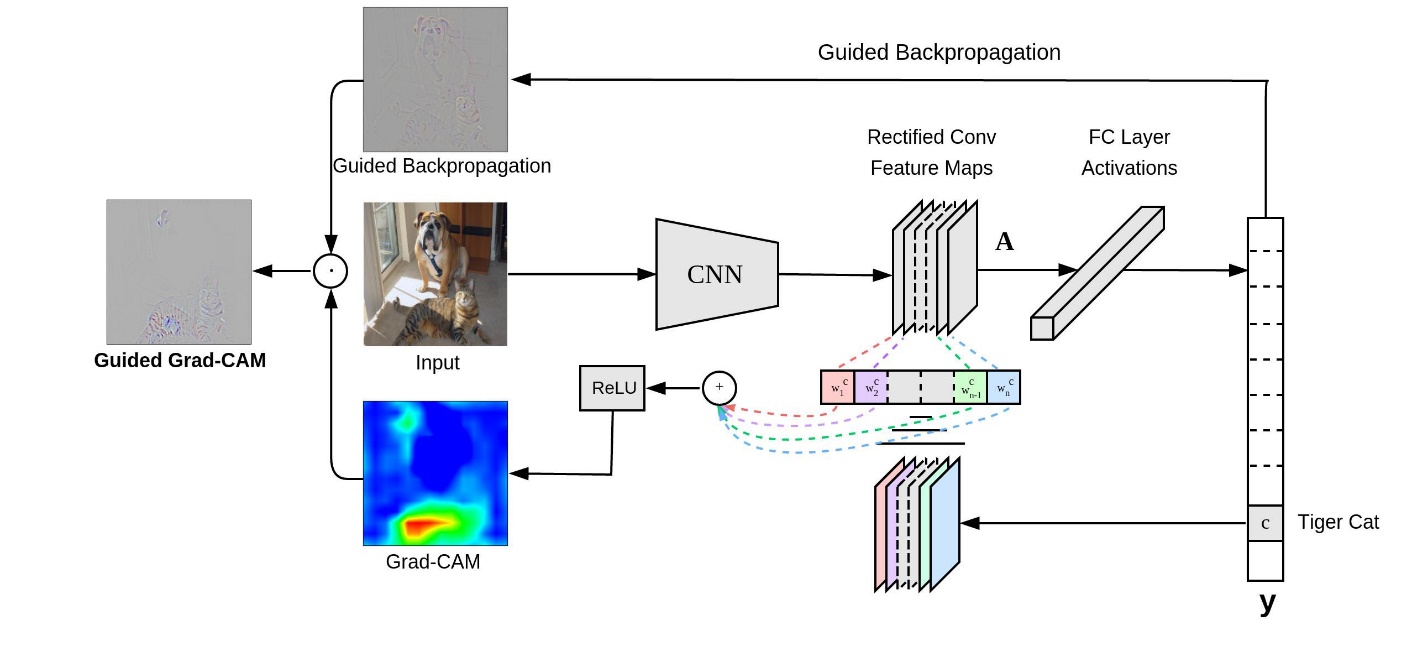
**Grad-Cam**

1. Giới thiệu

Đây là 1 kĩ thuật dựa trên thuật toán cnn(Convolutional Neural Network) làm cho mô hình trở nên minh bạch hơn, bằng cách trực quan hóa hình ảnh ( visual explain).

Phương pháp này dùng mô hình đã được đạo tọa khi thay đổi kiến trúc không cần phải đào tạo lại

1. Phương pháp

Note: Lan truyền ngược(Backpropagation): có nghĩa là sự mất mát của các pixel đầu vào.(gradient)

Tài liệu tham khảo:

<https://medium.com/@mohamedchetoui/grad-cam-gradient-weighted-class-activation-mapping-ffd72742243a>

**CNN**

Các bước thực hiện:

1. Đọc hình ảnh đầu vào (Reading the input image)
2. Chuẩn bị bộ lọc (preparing filters)

Lưu ý: là bộ lọc phải là ma trận vuông tức và có kích thước lẻ tức là cột và hàng bằng nhau và bằng 1 con số lẻ. Số chiều của ảnh đầu vào phải khớp với chiều bộ lọc.

Ở đây ta tạo ra 2 bộ lọc có kích thước 3x3 (num\_filters=2, num\_row\_filter =3, num\_colum\_filters = 3), do hình ảnh đầu vào là 2D không có chiều sâu nên ta chỉ 2 bộ lọc( tức 2D), neeys là ảnh RBG với 3 chiều chanel, kích thược bộ lọc là (3,3,3).

ở đây mình tạo bộ lọc để phải phát hiện cạnh dọc và ngang.

**l1\_filter [0,:,:] = numpy.array([[[- 1, 0, 1],**

**[-1, 0, 1],**

**[-1, 0, 1]]])**

=>Đây là phát hiện cạnh dọc

**l1\_filter [1,:,:] = numpy.array([[[1, 1, 1],**

**[0, 0, 0],**

**[-1, -1, -1]]])**

=> Đây là phát hiện cạnh ngang

1. Lớp chuyển đổi (Conv Layer)

Đây là bước biến đổi hình ảnh đầu vào, chuyển đổi hình ảnh được bắt đầu bằng cách khởi tạo một mảng để giữu đầu ra của tích chập( feature map) bằng cách chỉ định kích thước nó theo sau:

**feature\_maps = numpy.zeros ((img.shape [0] -conv\_filter.shape [1] +1, img.shape [1] -conv\_filter.shape [1] +1, var\_filter.shape [0]))**

Vòng lặp bên ngoài lặp lại tưng bộ lọc và trả về ở

**curr\_filter = convert\_filter [filter\_num,:], (nhận bộ lọc)**

Việc thực hiện tích chập nằm ở hàm conv\_()