# Class Activation Map

# Preface

- Convolutional layers of CNNs actually behave as object detectors (i.e., to localize objects) Despite no supervision on the location of the object is provided.

- In other words, convolutional layers naturally retain spatial information.

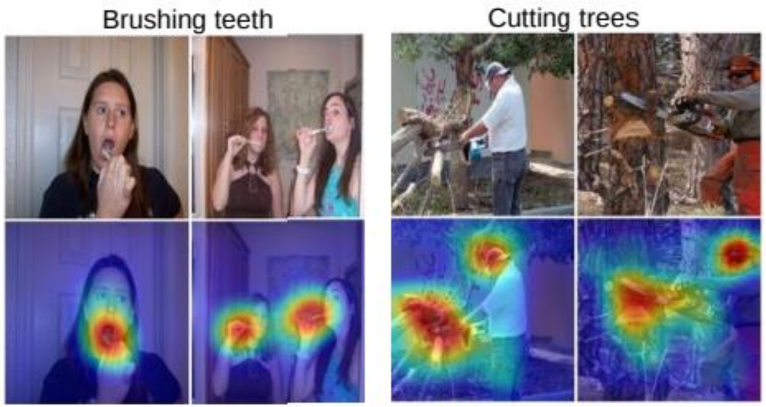
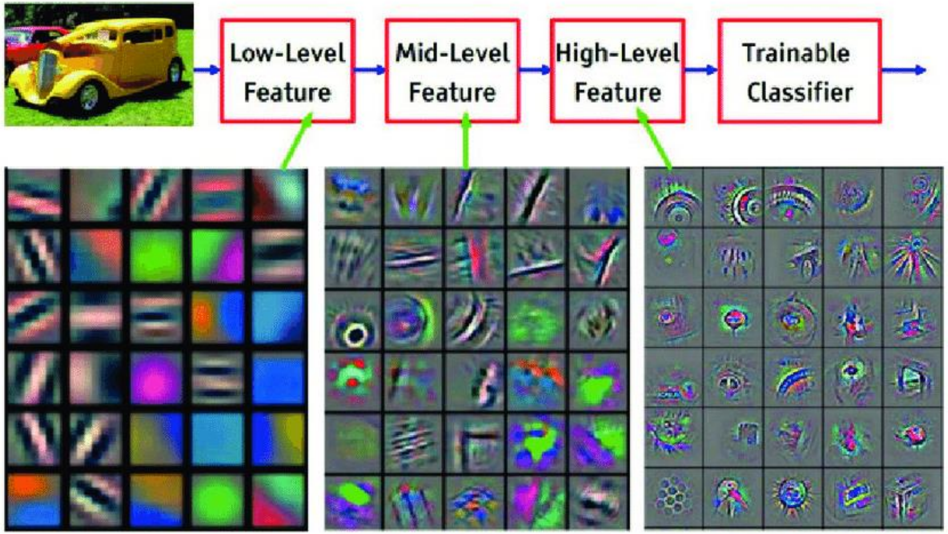



Figure 1. A simple modification of the global average pooling layer combined with our class activation mapping (CAM) technique allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass e.g., the toothbrush for *brushing teeth* and the chainsaw for *cutting trees*.

E.g., for classification, CNN is able to localize the discriminative region.

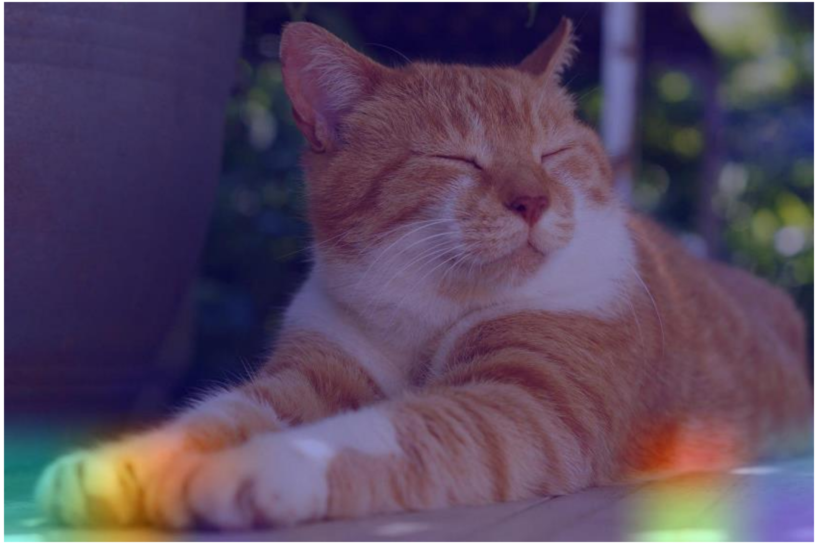
# Preface

- However, this ability (spatial information) is lost in fully-connected layers.
- So we expect the last convolution layer have the most detailed spatial information. The higher the convolutional-layers are, the higher level pf semantics are extracted.



# Why we need CAM?

- AI explainability
- Training quality



# CAM (Class Activation Map)

- For a particular category, a Class Activation Map (CAM) indicates the discriminative image regions used by the CNN to identify that category.
- Replace fully-connected layers with global average pooling (GAP) layers.

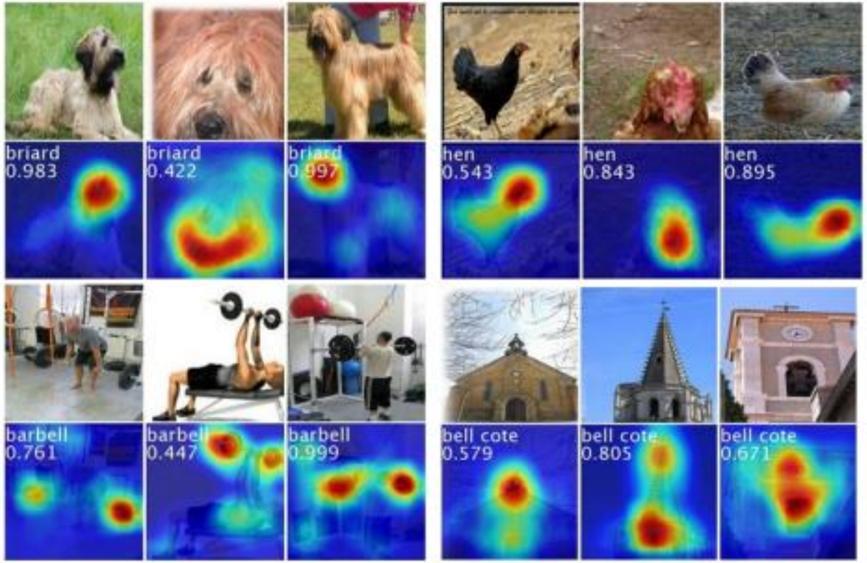



Figure 3. The CAMs of four classes from ILSVRC [20]. The maps highlight the discriminative image regions used for image classification e.g., the head of the animal for *briard* and *hen*, the plates in *barbell*, and the bell in *bell cote*.

Paper

# CAM

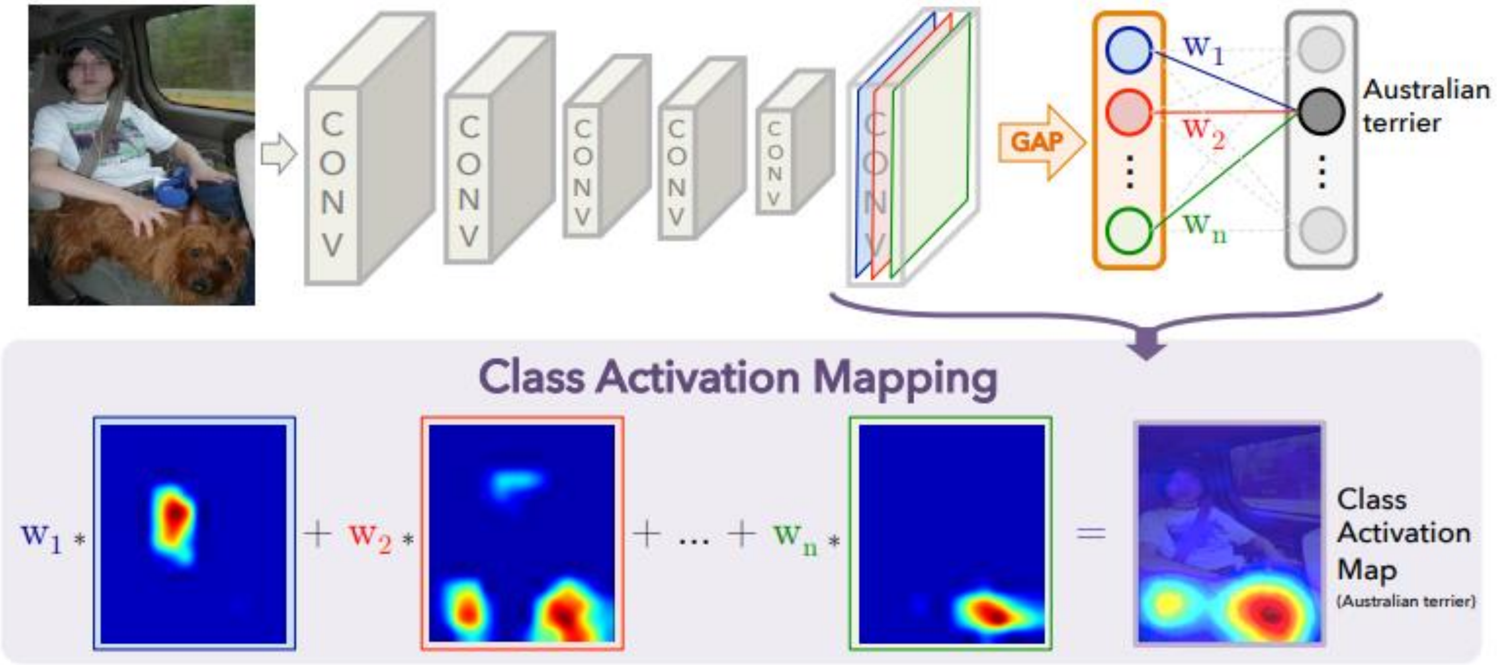



Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

# CAM

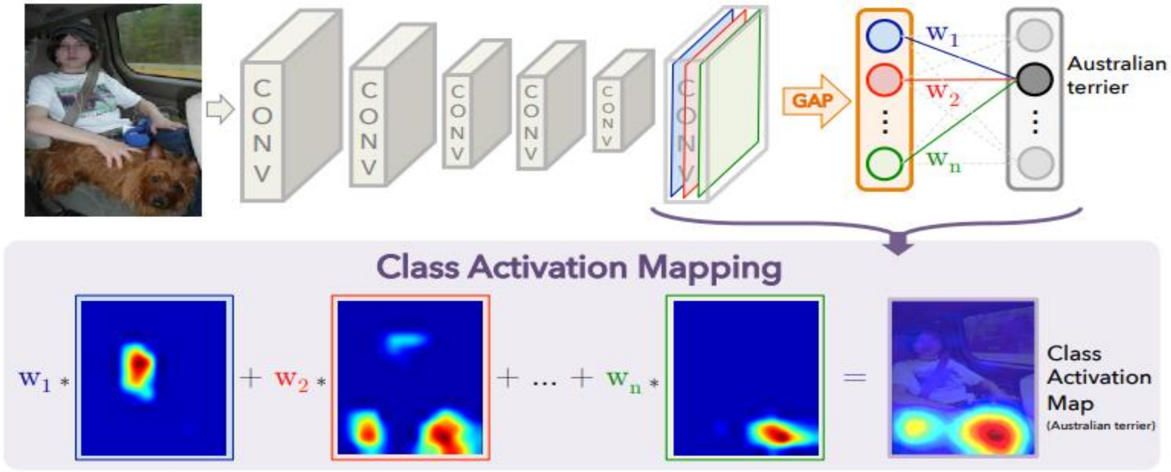



Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

1. For each feature map ($f_k(x, y), k = 1, \dots, n$) at the last convolution layer, GAP outputs the spatial average of each feature map

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

2. For a given class c, the input for output layer: $S_c = \sum_k w_k^c F_k$

3. Output score for class c: $P_c = \frac{\exp(S_c)}{\sum_c \exp(S_c)}$ (e.g., softmax)

# CAM

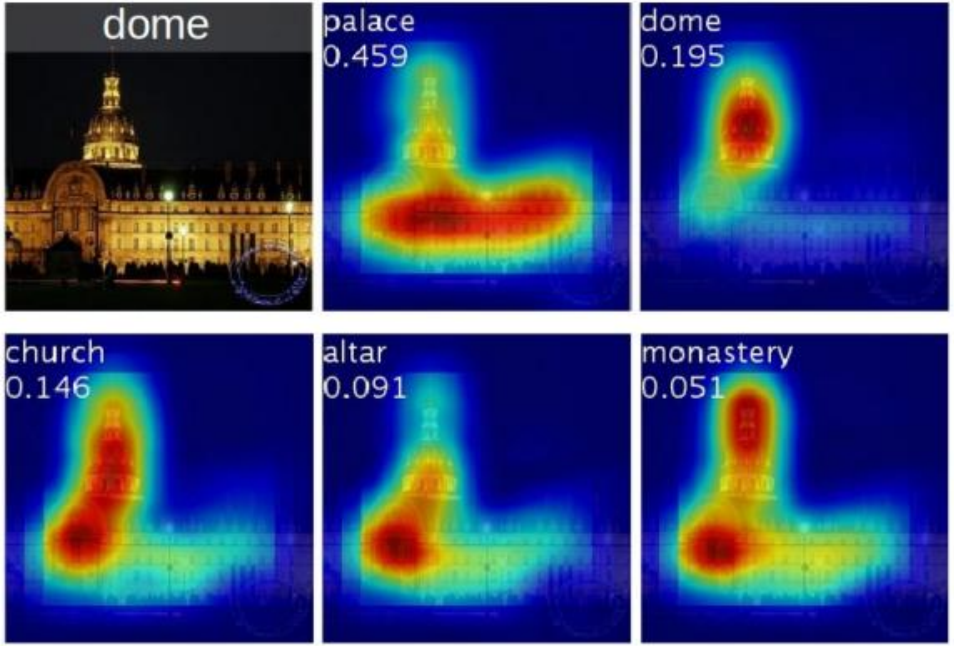



Figure 4. Examples of the CAMs generated from the top 5 predicted categories for the given image with ground-truth as dome. The predicted class and its score are shown above each class activation map. We observe that the highlighted regions vary across predicted classes e.g., *dome* activates the upper round part while *palace* activates the lower flat part of the compound.

# CAM

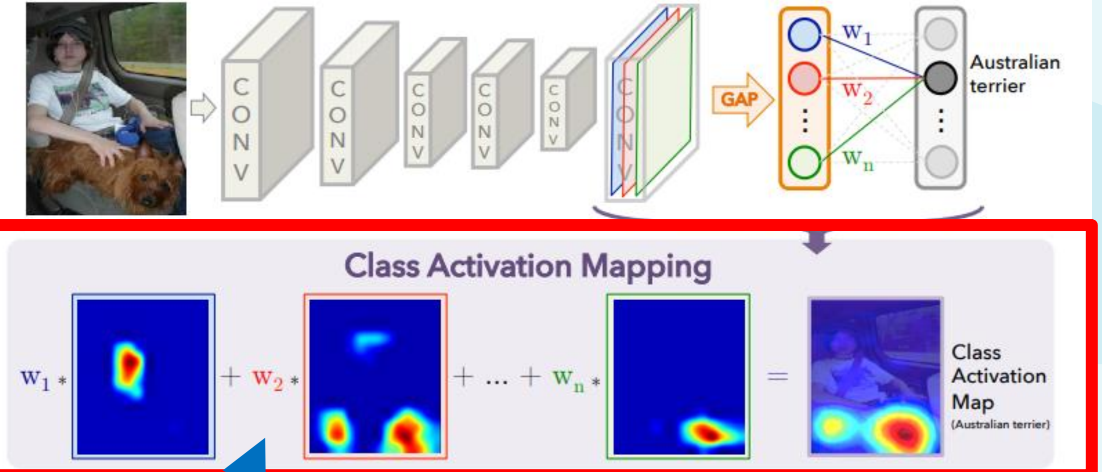



Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

Compute CAM:

$$M_c(x, y) = \sum_k w_k^c f_k(x, y)$$

Up-sampling to the shape of input image.

# GRAD - CAM

- Gradient-weighted Class Activation Mapping (Grad - CAM) generalizes CAM for a wide variety of CNN-based architectures. i.e., without requiring architectural changes or re-training.
- Without GAP layers, we need a way to define weights - $W_k^c$
- Grad – CAM uses the gradients of any target concept flowing into the final convolutional layer, and derive summary statistics out of it to represent the weights.

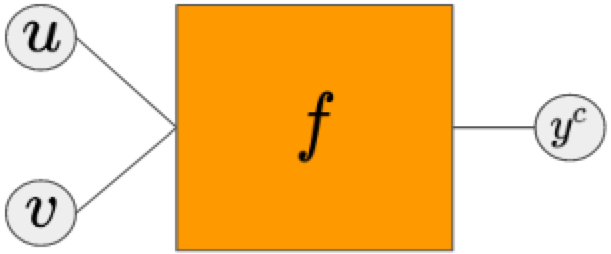
Paper

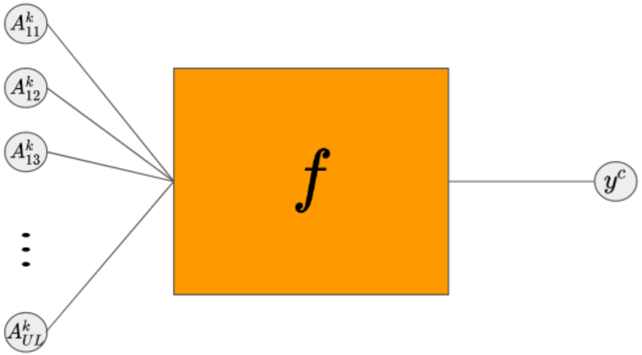
$$S_c = \sum_k w_k^c F_k$$

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

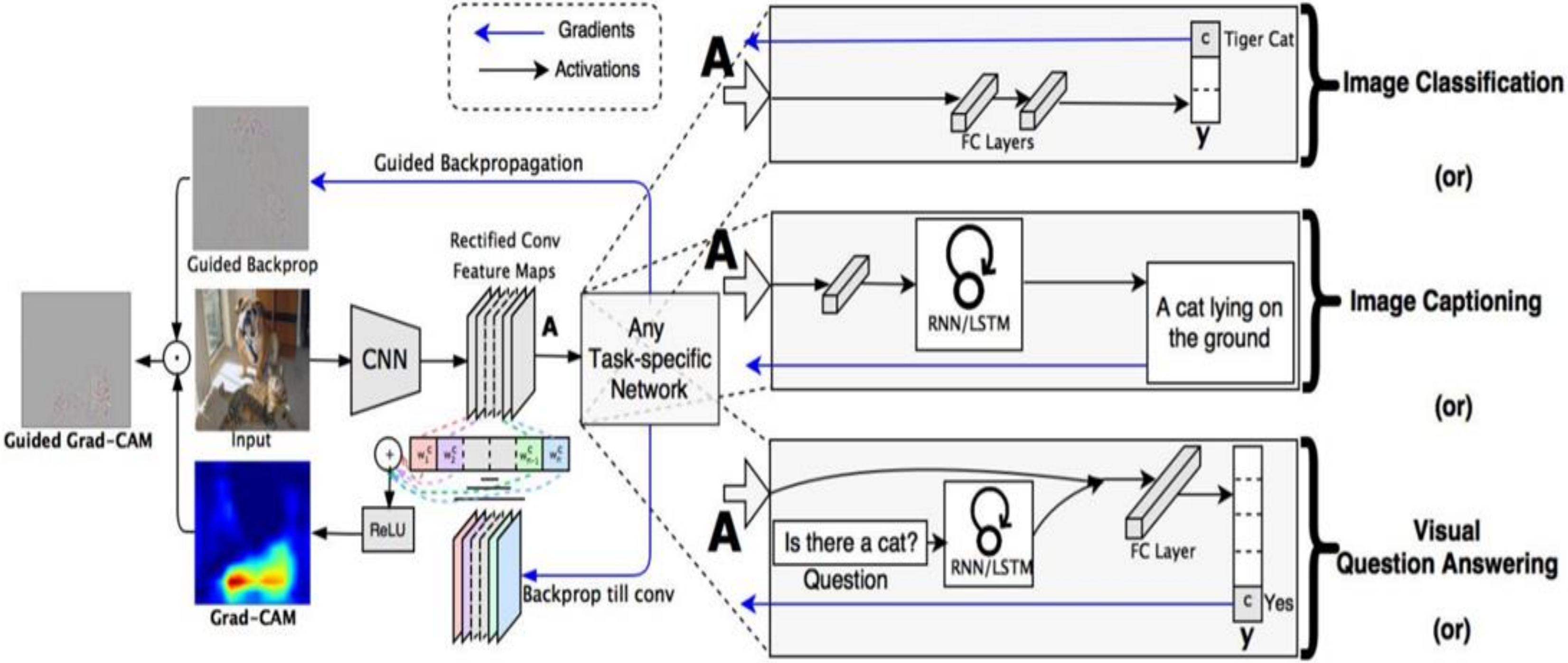
$$Y^c = \underbrace{\sum_k w_k^c}_{\text{class feature weights}} \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{A_{ij}^k}_{\text{feature map}}$$

$$w_k^c = \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k}$$




$$
\begin{aligned}
y^c &= f(u,v) \\
&\approx f(u_0,v_0) + \frac{\partial y^c}{\partial u}\Big|_{u=u_0,v=v_0}(u-u_0) + \frac{\partial y^c}{\partial v}\Big|_{u=u_0,v=v_0}(v-v_0) \\
&= \underbrace{f(u_0,v_0) - \frac{\partial y^c}{\partial u}u_0 - \frac{\partial y^c}{\partial v}v_0}_{\text{Bias term}} + \frac{\partial y^c}{\partial u}u + \frac{\partial y^c}{\partial v}v
\end{aligned}
$$


Gradients
Activations
Guided Backpropagation
Guided Backprop
Rectified Conv
Feature Maps
Guided Grad-CAM
Input
CNN
A
Any
Task-specific
Network
ReLU
Grad-CAM
Backprop till conv
A
Tiger Cat
FC Layers
y
Image Classification
(or)
A
RNN/LSTM
A cat lying on
the ground
Image Captioning
(or)
A
Is there a cat?
Question
RNN/LSTM
FC Layer
Yes
y
Visual
Question Answering
(or)

# GRAD – CAM

- For a given class c, compute the gradient of its score - $y^c$ ( before the softmax ), w.r.t each feature map activations $A_k \in \mathbb{R}^{u*v}, \mathrm{k} = 1, \ldots, \mathrm{n}$ of a convolutinal layer,

$\mathrm{i.e.}, \frac{\partial y^c}{\partial A_k} \in \mathbb{R}^{u*v}$ ← Influence of $A_k$( x, y ) to $y^c$

- Define the importance weights of feature map k via GAP :

$$\alpha_k^c = \frac{1}{Z} \sum_{i \in x} \sum_{j \in y} \frac{\partial y^c}{\partial A_{ij}^k}$$

Paper

# GRAD - CAM

- Compute Grad – CAM :

$$L^{c}_{Grad-CAM}(x,y) = \underline{ReLU}\left(\sum_k \alpha_k^c \, A^k(x,y)\right) \in \mathbb{R}^{u\times v}$$

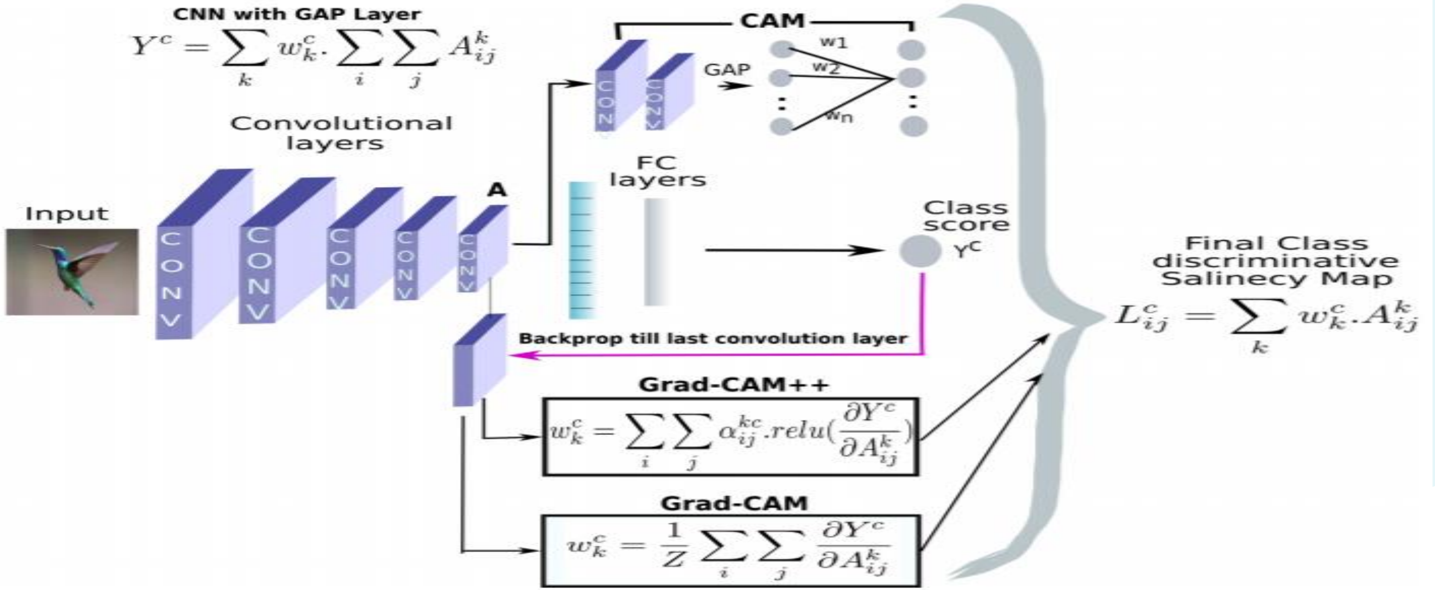
cut off non-positive values

- ReLu is applied because we are only interested in the features that have a positive influence on the class of interest

# GRAD - CAM ++

$$w_k^c = \sum_i \sum_j \alpha_{ij}^{kc} . relu\left(\frac{\partial Y^c}{\partial A_{ij}^k}\right)$$

where $$\alpha_{ij}^{kc} = \frac{\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2}}{2\frac{\partial^2 Y^c}{(\partial A_{ij}^k)^2} + \sum_a \sum_b A_{ab}^k \{\frac{\partial^3 Y^c}{(\partial A_{ij}^k)^3}\}}$$



# Score - CAM

梯度是可能會存在 noise ，且會有梯度飽和的問題，因此Score-CAM改用模型運算結果來求得 score

## Phase 1:

將影像輸入模型進行推論，將特徵upsample，做normalization 成為 0 ~1 範圍

$$H_l^k = s(Up(A_l^k))$$

## Phase 2:

當成遮罩，做為各點權重並和原圖進行點運算，再扣除 baseline 得到之結果

Increase of Confidence $$C(A_l^k) = f(X \circ H_l^k) - f(X_b)$$

結果softmax normalized to 0~1，作為 $\alpha_k^c$

**Algorithm 1:** Score-CAM algorithm

**Input:** Image $X_0$, Baseline Image $X_b$, Model $f(X)$, class $c$, layer $l$
**Output:** $L^c_{Score-CAM}$
initialization;
// get activation of layer $l$;
$M \leftarrow [], A_l \leftarrow f_l(X)$
$C \leftarrow$ the number of channels in $A_l$
**for** $k$ *in* $[0, ..., C-1]$ **do**
  $M_l^k \leftarrow \text{Upsample}(A_l^k)$
  // normalize the activation map;
  $M_l^k \leftarrow \text{s}(M_l^k)$
  // Hadamard product;
  $M.\text{append}(M_l^k \circ X_0)$
**end**
$M \leftarrow \text{Batchify}(M)$
// $f^c(\cdot)$ as the logit of class $c$;
$S^c \leftarrow f^c(M) - f^c(X_b)$
// ensure $\sum_k \alpha_k^c = 1$ in the implementation;
$\alpha_k^c \leftarrow \frac{\exp(S_k^c)}{\sum_k \exp(S_k^c)}$
$L^c_{Score-CAM} \leftarrow ReLU(\sum_k \alpha_k^c A_l^k)$

# Application

- Analyze training quality and training explainability
- Utilize Class Activation Map to generate a pseudo mask.