Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization

- [[📚 1610.02391.pdf]]
- **Basic Idea**:
  - The gradient of the final output of the network wrt to the final convolutional features should give us a sense of relative importance of different regions in the image in predicting that output.
  - A good visual explanation should have following properties:
    - 🟡 **P2** classdiscriminative (i.e. localize the category in the image)
    - 🟡 **P2** high-resolution (i.e. capture fine-grained detail).

- **Grad-CAM**
  - *Notations*:
    - $L^c_{Grad-CAM} \in \mathbb{R}^{u \times v}$: class-discriminative localization map Grad-CAM of width u and height v for any class c.
    - y^c is the score of the class c predicted by the network before applying softmax
    - A^k is the k^{th} feature map activations of a convolutional layer.
    - A^k_{ij} is the activation at location (i, j) of the feature map A^k.
    - α^c_k: Neuron importance weight for k^{th} feature map for class c.
  - Calculate α^c_k as:

    🟡 **P4**

$$\alpha^c_k = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A^k_{ij}}}_{\text{gradients via backprop}} \qquad (1)$$

  - The final Grad-CAM can be obtained as follows:

    🟡 **P5**

$$L^c_{\text{Grad-CAM}} = ReLU \underbrace{\left(\sum_k \alpha^c_k A^k\right)}_{\text{linear combination}} \qquad (2)$$

  - Few points to note:
    - 🟡 **P5** Notice that this results in a coarse heatmap of the same size as the convolutional feature maps (14 × 14 in the case of last convolutional layers of VGG [52] and AlexNet [33] networks) 3.
      Why last convolutional layer? 🟡 **P4** Furthermore, convolutional layers naturally retain spatial information which is lost in fully-connected layers, so we can expect the last convolutional layers to have the best compromise between high-level semantics and detailed spatial information. 🟡 **P5** We find that Grad-CAM maps become progressively worse as we move to earlier convolutional layers as they have smaller receptive fields and only focus on less semantic local features.
    - *Why ReLU?* 🟡 **P5** We apply a ReLU to the linear combination of maps because we are only interested in the features that have a positive influence on the class of interest, i.e. pixels whose intensity should be increased in order to increase yc. Negative pixels are likely to belong to other categories in the image. As expected, without this ReLU, localization maps sometimes highlight more than just the desired class and perform worse at localization.
    - *Global Average Pooling vs Global Max Pooling*: 🟡 **P4** Empirically we found global-average-pooling to work better than global-max-pooling as can be found in the Appendix
      This result is consistent with the CAM paper as well which reasons the above observation as follows:
      - *Global Average Pooling (GAP) vs Global Max Pooling (GMP)*:
        - 🟡 **P3** We believe that GAP loss encourages the network to identify the extent of the object as compared to GMP which encourages it to identify just one discriminative part.
        - 🟡 **P3** This is because, when doing the average of a map, the value can be maximized by finding all discriminative parts of an object as all low activations reduce the output of 🟡 **P4** the particular map. On the other hand, for GMP, low scores for all image regions except the most discriminative one do not impact the score as you just perform a max.

    - Empirically, they prove that GMP achieves similar classification performance to GAP. However, GAP outperforms GMP for localization.

- *How to make Grad-CAM high-resolution?* (tldr: Use Guided Backpropagation)
    - 🟡 **P6** Guided Backpropagation visualizes gradients with respect to the image where negative gradients are suppressed when backpropagating through ReLU layers. Intuitively, this aims to capture pixels detected by neurons, not the ones that suppress neurons.
    - Upsample $L^c_{Grad-CAM}$ using bilinear interpolation to the input image resolution and then do an element-wise multiplication with the Guided Backpropagation output to obtain the Guided Grad-CAM output.

- *Counterfactual Explanations*:
    - This refers to the regions whose presence in the image can make the network change its prediction (or removal of these regions will make the network more confident about its predictions). In simple words, for a binary classification task these regions would correspond to class opposite to c.
    - 🟡 **P6** Specifically, we negate the gradient of yc (score for class c) with respect to feature maps A of a convolutional layer. Therefore, α^c_k can be calculated as

        🟡 **P6**

$$\alpha^c_k = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{-\frac{\partial y^c}{\partial A^k_{ij}}}_{\text{Negative gradients}} \qquad (12)$$

    - In order to intuitively understand why this works, let's consider binary classification task. In order to highlight the regions important for the opposite class, we should calculate the Grad-CAM wrt to the score $1 - \text{softmax}(y^c)$

$$\begin{aligned}\alpha^c_k &= \frac{1}{Z}\sum_i \sum_j \frac{\partial(1 - \text{softmax}(y^c))}{\partial A^k_{ij}} \\ &= \frac{1}{Z}\sum_i \sum_j -\frac{\partial\, \text{softmax}(y^c)}{\partial A^k_{ij}}\end{aligned}$$

      If we ignore the derivative wrt softmax, then we obtain the above formula.

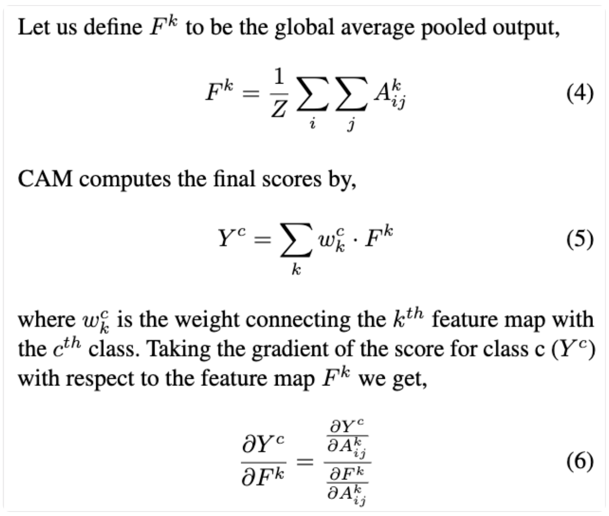
- *Grad-CAM generalizes CAM*
  tldr: Intuitively this should be true, as derivative of a linear function $w^T + b$ is w
    - In CAM the feature maps are globally average pooled and then directly fed to the softmax layer. Therefore, score Y^c for each class c is obtained as

        🟡 **P5**

$$Y^c = \sum_k \underbrace{w^c_k}_{\text{class feature weights}} \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{A^k_{ij}}_{\text{feature map}} \qquad (3)$$

    -

🟡 P5



Let us define F^k to be the global average pooled output,

$$F^k = \frac{1}{Z}\sum_i \sum_j A_{ij}^k \tag{4}$$

CAM computes the final scores by,

$$Y^c = \sum_k w_k^c \cdot F^k \tag{5}$$

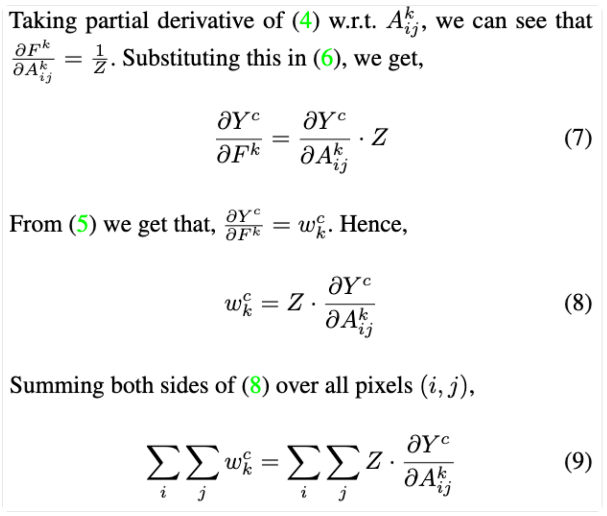
where w_k^c is the weight connecting the k^{th} feature map with the c^{th} class. Taking the gradient of the score for class c (Y^c) with respect to the feature map F^k we get,

$$\frac{\partial Y^c}{\partial F^k} = \frac{\frac{\partial Y^c}{\partial A_{ij}^k}}{\frac{\partial F^k}{\partial A_{ij}^k}} \tag{6}$$

The Eq. (6) is true because of the chain-rule i.e.,

$$\frac{\partial Y^c}{\partial A_{ij}^k} = \frac{\partial Y^c}{\partial F^k} \cdot \frac{\partial F^k}{\partial A_{ij}^k}$$

🟡 P5



Taking partial derivative of (4) w.r.t. A_{ij}^k, we can see that $\frac{\partial F^k}{\partial A_{ij}^k} = \frac{1}{Z}$. Substituting this in (6), we get,

$$\frac{\partial Y^c}{\partial F^k} = \frac{\partial Y^c}{\partial A_{ij}^k} \cdot Z \tag{7}$$

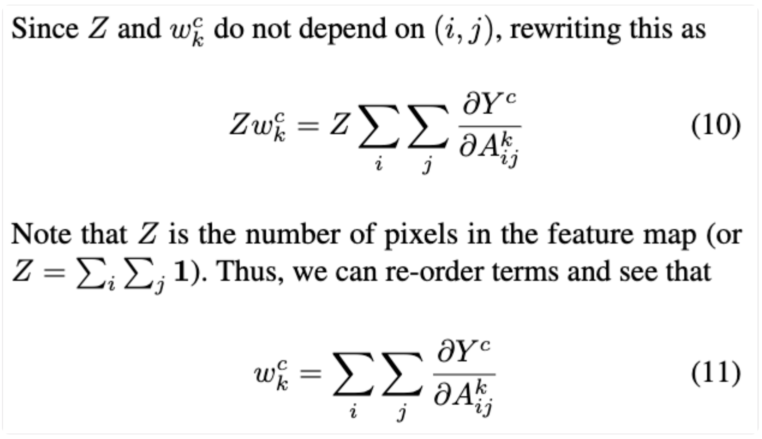
From (5) we get that, $\frac{\partial Y^c}{\partial F^k} = w_k^c$. Hence,

$$w_k^c = Z \cdot \frac{\partial Y^c}{\partial A_{ij}^k} \tag{8}$$

Summing both sides of (8) over all pixels (i, j),

$$\sum_i \sum_j w_k^c = \sum_i \sum_j Z \cdot \frac{\partial Y^c}{\partial A_{ij}^k} \tag{9}$$

🟡 P5



Since Z and w_k^c do not depend on (i, j), rewriting this as

$$Zw_k^c = Z\sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k} \tag{10}$$

Note that Z is the number of pixels in the feature map (or $Z = \sum_i \sum_j \mathbf{1}$). Thus, we can re-order terms and see that

$$w_k^c = \sum_i \sum_j \frac{\partial Y^c}{\partial A_{ij}^k} \tag{11}$$

🟡 P5 Up to a proportionality constant (1/Z) that gets normalizedout during visualization, the expression for wc k is identical toαc k used by Grad-CAM (1) . Thus, Grad-CAM is a strict generalization of CAM.