Explainable artificial intelligence (XAI)

Why Explainable AI (XAI)?

- Provides transparency in AI models, especially in critical fields like healthcare, finance, …
- Helps uncover biases in black-box models, increasing trust and accountability
- Ensures compliance with regulations (e.g., GDPR) requiring understandable decision-making
- Improves user trust by offering insight into how AI decisions are made

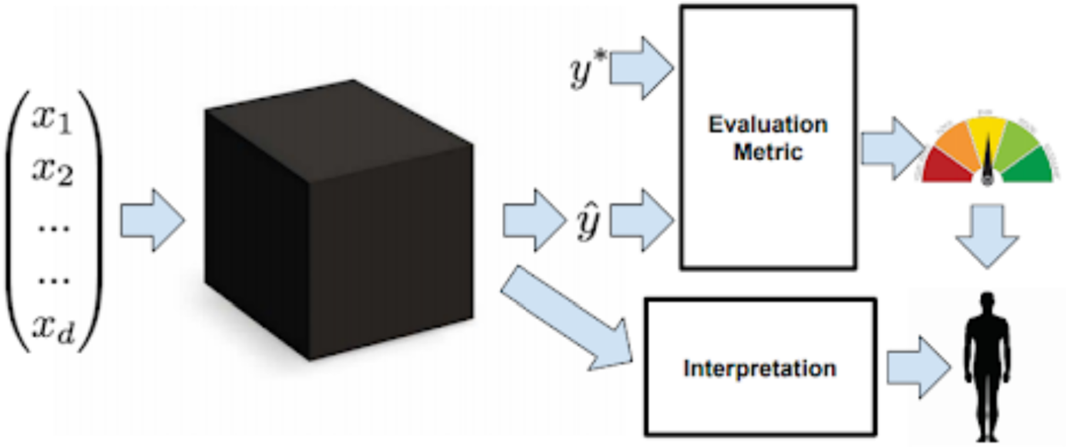
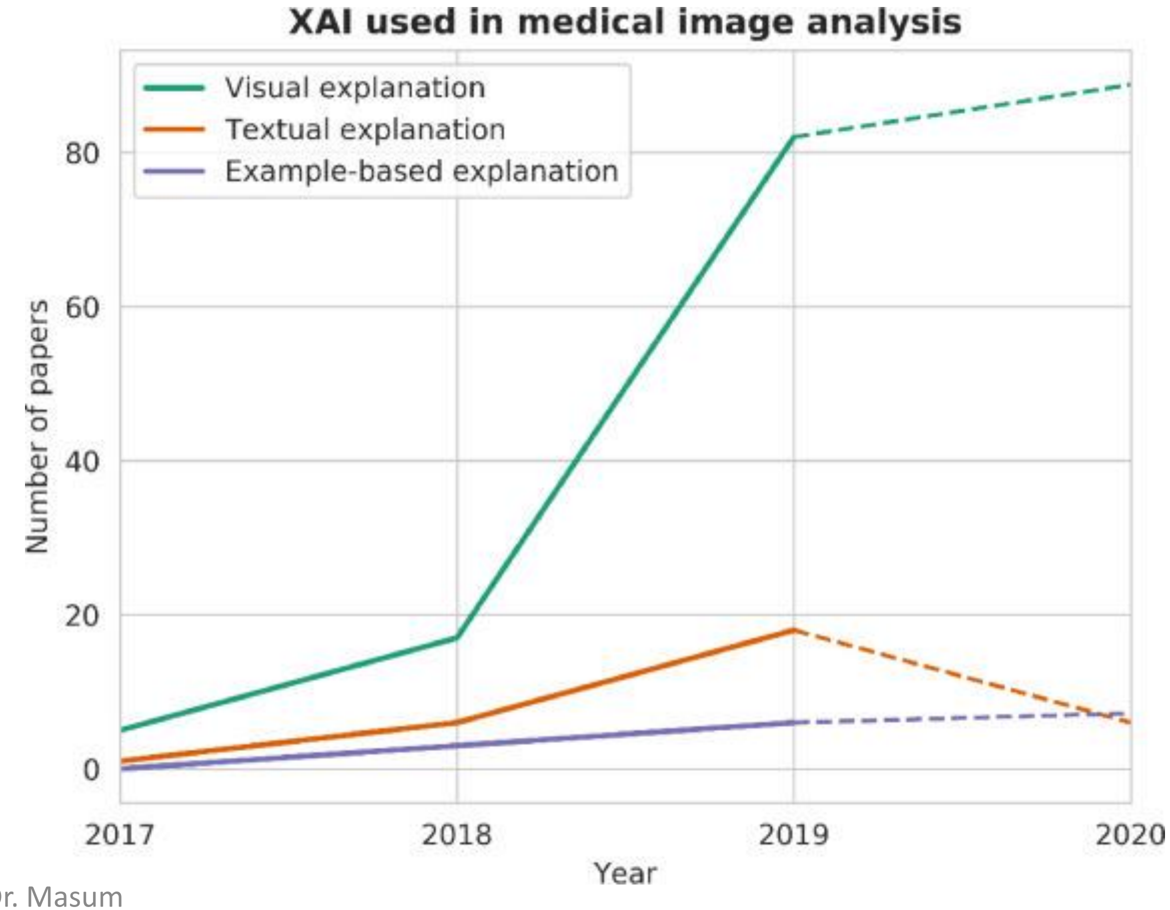


Figure 2: Stakeholders demand interpretability when the evaluation metric does not reflect the real cost of a model in deployment. The interpretation of the model and its decisions confer information about the true desiderata. (Lipton 2016)

Explainable artificial intelligence (XAI)

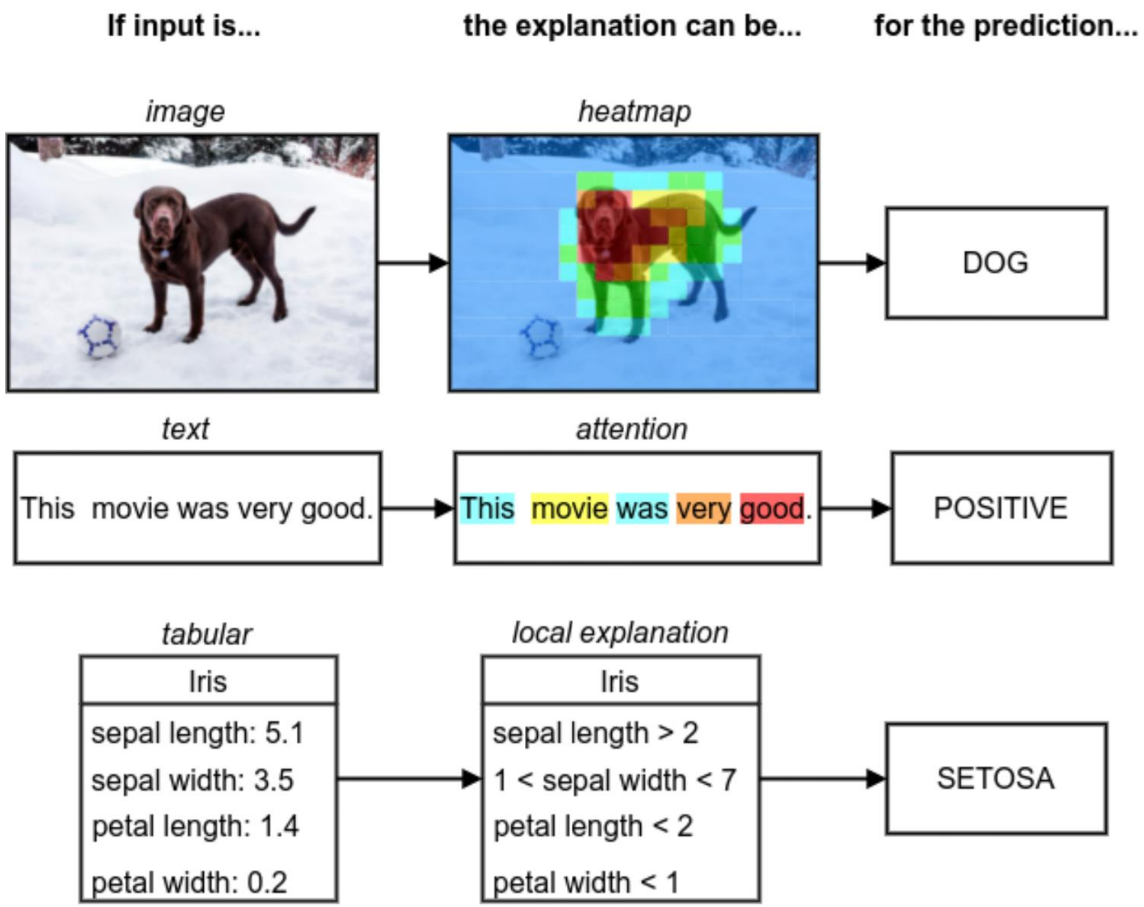
Explainable AI (XAI) makes AI model decisions interpretable and understandable by providing insights into the internal workings of models, such as deep learning, and highlighting the key factors influencing their decisions.

- Visual explanation
- Textual explanation
- Example-based explanation



Van der Velden, B. H., Kuijf, H. J., Gilhuijs, K. G., & Viergever, M. A. (2022). Explainable artificial intelligence (XAI) in deep learning-based medical image analysis. *Medical Image Analysis*, *79*, 102470.

Explainable artificial intelligence (XAI)



Ras, G., Xie, N., Van Gerven, M., & Doran, D. (2022). Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, *73*, 329-396.

Explainable artificial intelligence (XAI)

- **Visual Explanation (Saliency Mapping)**: Most common form of XAI, specially in medical image analysis.
- **Saliency Maps**: Highlight important parts of an image that contribute to a decision.
- **Backpropagation-based Techniques**: The most widely used approach for generating saliency maps, tracing gradients back to input pixels.
    - Class Activation Maps (CAM)
    - Gradient-weighted class activation mapping (Grad-CAM)
    - Layer-wise relevance propagation (LRP)
    - Deep SHapley Additive exPlanations (Deep SHAP)
- **Perturbation-based Techniques**: Alter parts of the image to observe the impact on model decisions, providing insight into important regions.

Explainable artificial intelligence (XAI)

- Convolutional Layers in CNNs act as object detectors, even without supervision of object location
  - Localization ability occurs naturally within the convolutional units across layers
  - However, when FC layers are applied for classification, the object localization ability reduced significantly

Explainable artificial intelligence

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

- Proposed Class Activation Maps using Global Average Pooling in CNNs
- GAP acts as a structural regularizer, helping to prevent overfitting during training.
  - Beyond regularization, global average pooling enhances the network's localization ability.
  - With slight adjustments, the network can maintain its strong localization ability up to the final layer.
    - Identifying discriminative image regions in a single forward pass
- Despite the simplicity, GAP achieves impressive results: 37.1% top-5 error in object localization on ILSVRC 2014

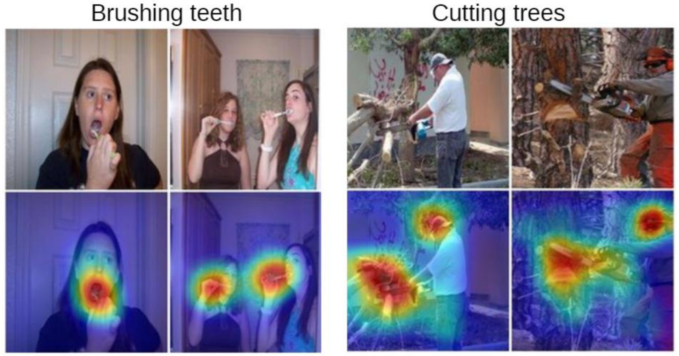


Figure 1. A simple modification of the global average pooling layer combined with our class activation mapping (CAM) technique allows the classification-trained CNN to both classify the image and localize class-specific image regions in a single forward-pass e.g., the toothbrush for *brushing teeth*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

Explainable artificial intellig

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

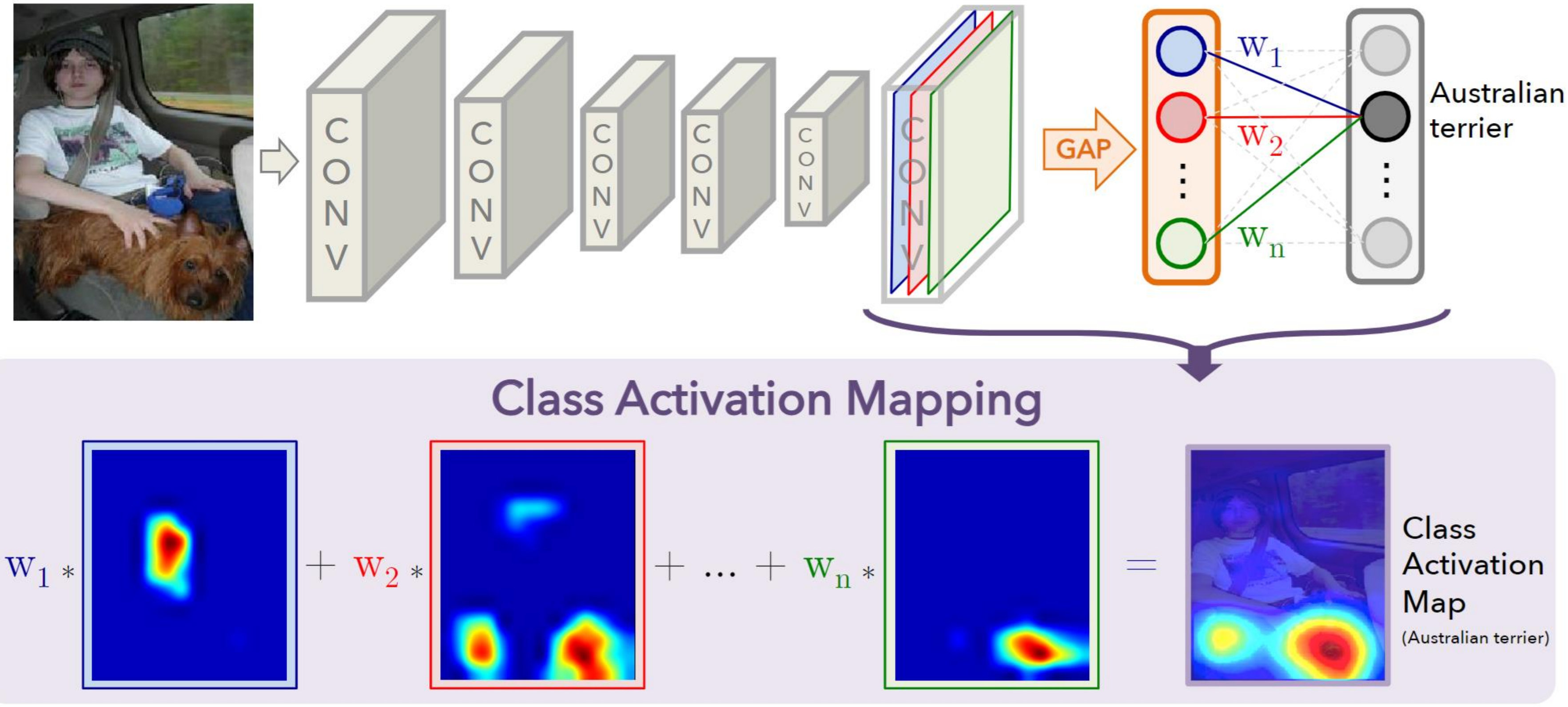


Figure 2. Class Activation Mapping: the predicted class score is mapped back to the previous convolutional layer to generate the class activation maps (CAMs). The CAM highlights the class-specific discriminative regions.

Explainable artificial intellig

Learning Deep Features for Discriminative Localization

Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, Antonio Torralba
Computer Science and Artificial Intelligence Laboratory, MIT
{bzhou,khosla,agata,oliva,torralba}@csail.mit.edu

Example Workflow

1. **Input**: Image of a cat
2. **Final Conv Layer Output**: Feature maps $(F_1, F_2, \ldots, F_n)$
3. **GAP Layer**: Averages each feature map to create a vector
4. **Fully Connected Layer**: Produces class scores based on weighted sums of the GAP output
5. **Extract Weights for "Cat"**: Use weights corresponding to the "Cat" class from the fully connected layer
6. **CAM Calculation**: Weighted sum of the feature maps, using the "Cat" class weights
7. **Resize CAM**: Resize the CAM to match the original image size
8. **Overlay**: Overlay the CAM on the original image to visualize the important regions for the "Cat" prediction

Explainable artificial intelligence (XAI)

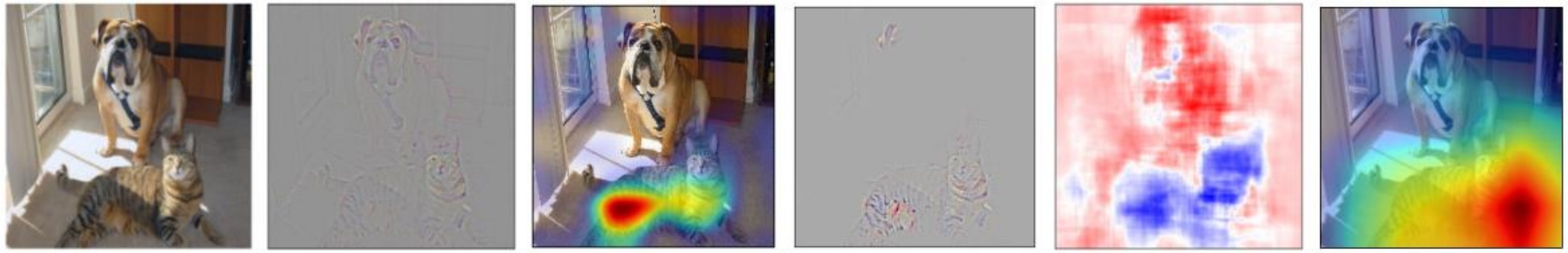
Limitations

- CAM requires specific CNN architectures with GAP before SoftMax layer → limiting its use case
    - This may lead to inferior accuracies on tasks like image classification
    - Unsuitable for image captioning or visual question answering
- Grad-CAM extends CAM to work with a wider range of architecture, including FC layers

Explainable artificial intelligence (XAI)

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

Ramprasaath R. Selvaraju · Michael Cogswell · Abhishek Das · Ramakrishna Vedantam · Devi Parikh · Dhruv Batra



(a) Original Image (b) Guided Backprop 'Cat' (c) Grad-CAM 'Cat' (d)Guided Grad-CAM 'Cat' (e) Occlusion map 'Cat' (f) ResNet Grad-CAM 'Cat'

(g) Original Image (h) Guided Backprop 'Dog' (i) Grad-CAM 'Dog' (j)Guided Grad-CAM 'Dog' (k) Occlusion map 'Dog' (l)ResNet Grad-CAM 'Dog'

Fig. 1: (a) Original image with a cat and a dog. (b-f) Support for the cat category according to various visualizations for VGG-16 and ResNet. (b) Guided Backpropagation [53]: highlights all contributing features. (c, f) Grad-CAM (Ours): localizes class-discriminative regions, (d) Combining (b) and (c) gives Guided Grad-CAM, which gives high-resolution class-discriminative visualizations. Interestingly, the localizations achieved by our Grad-CAM technique, (c) are very similar to results from occlusion sensitivity (e), while being orders of magnitude cheaper to compute. (f, l) are Grad-CAM visualizations for ResNet-18 layer. Note that in (c, f, i, l), red regions corresponds to high score for class, while in (e, k), blue corresponds to evidence for the class. Figure best viewed in color.

Explainable artificial intelligence (XAI)

Our approach – Gradient-weighted Class Activation Mapping (Grad-CAM), uses the gradients of any target concept (say 'dog' in a classification network or a sequence of words in captioning network) flowing into the final convolutional layer to produce a coarse localization map highlighting the important regions in the image for predicting the concept.

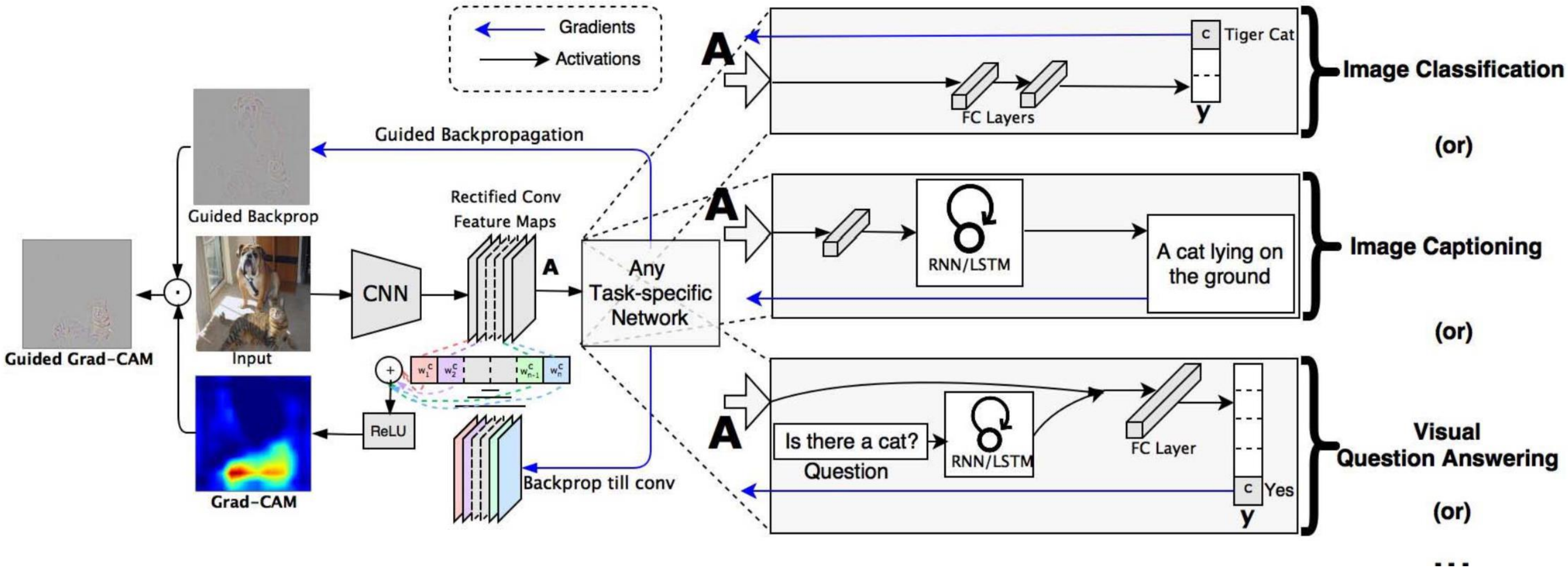
Grad-CAM is applicable to various CNN architecture without architectural changes or re-training:

- CNNs with FC layers (e.g. VGG)
- CNNs used for structured outputs (e.g. captioning)
- CNNs used in tasks with multimodal inputs (e.g. visual question answering) or reinforcement learning

Explainable artificial intelligence (XAI)

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

Ramprasaath R. Selvaraju[1*] Michael Cogswell[1] Abhishek Das[1] Ramakrishna Vedantam[1*]
Devi Parikh[1,2] Dhruv Batra[1,2]
[1]Georgia Institute of Technology [2]Facebook AI Research
{ramprs, cogswell, abhshkdz, vrama, parikh, dbatra}@gatech.edu



Explainable artificial intelligence (XAI)

Step 1: compute the gradient of y^c w.r.t. the feature map activation A^k of final conv layer

$$\frac{\partial y^c}{\partial A^k}$$

Step 2: Calculate GAP of the feature maps

$$\alpha_k^c = \overbrace{\frac{1}{Z}\sum_i \sum_j}^{\text{global average pooling}} \underbrace{\frac{\partial y^c}{\partial A_{ij}^k}}_{\text{gradients via backprop}}$$

Step 3: Calculate final grad-CAM localization map

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

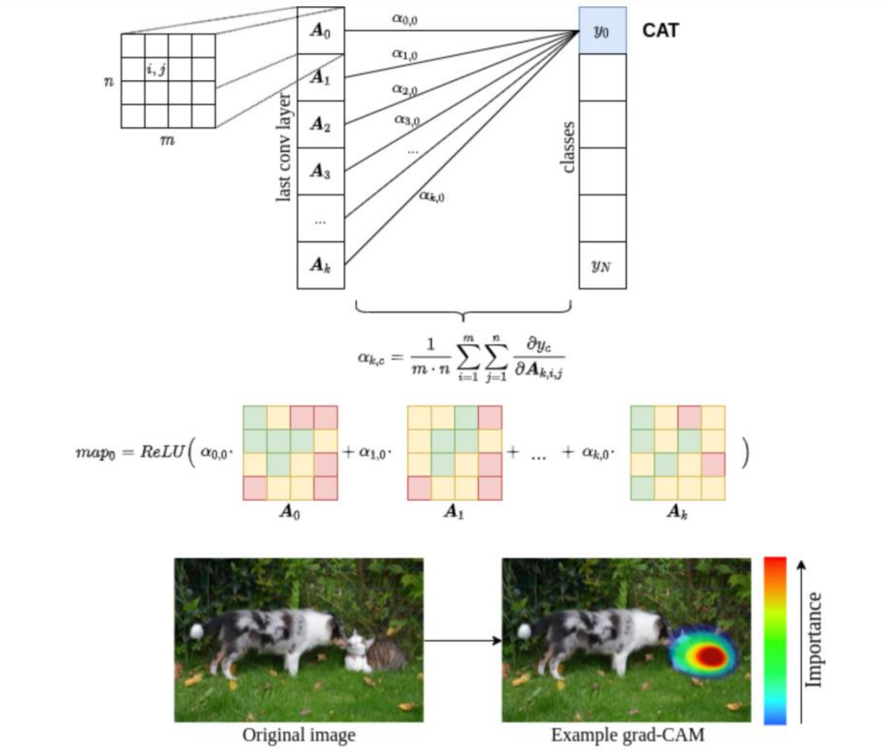


Figure 5: Visual explanation of how grad-CAM works. **Top**: Visualization of Equation 5 for calculating the importance scores $\alpha_{i,j}$ for each feature map $\boldsymbol{A}_k$. **Middle**: The heatmap for a specific class is computed by multiplying the importance score with each feature map and taking the sum. Afterwards the heatmap is upsampled and overlaid on the original image. **Bottom**: Heatmap for the prediction "cat".

Ras, G., Xie, N., Van Gerven, M., & Doran, D. (2022). Explainable deep learning: A field guide for the uninitiated. *Journal of Artificial Intelligence Research*, *73*, 329-396.

Explainable artificial intelligence (XAI)

```python
def make_gradcam_heatmap(img_array, model, last_conv_layer_name, pred_index=None):
    # First, we create a model that maps the input image to the activations
    # of the last conv layer as well as the output predictions
    grad_model = keras.models.Model(
        model.inputs, [model.get_layer(last_conv_layer_name).output, model.output]
    )

    # Then, we compute the gradient of the top predicted class for our input image
    # with respect to the activations of the last conv layer
    with tf.GradientTape() as tape:
        last_conv_layer_output, preds = grad_model(img_array)
        if pred_index is None:
            pred_index = tf.argmax(preds[0])
        class_channel = preds[:, pred_index]

    # This is the gradient of the output neuron (top predicted or chosen)
    # with regard to the output feature map of the last conv layer
    grads = tape.gradient(class_channel, last_conv_layer_output)

    # This is a vector where each entry is the mean intensity of the gradient
    # over a specific feature map channel
    pooled_grads = tf.reduce_mean(grads, axis=(0, 1, 2))
```

```python
# We multiply each channel in the feature map array
# by "how important this channel is" with regard to the top predicted class
# then sum all the channels to obtain the heatmap class activation
last_conv_layer_output = last_conv_layer_output[0]
heatmap = last_conv_layer_output @ pooled_grads[..., tf.newaxis]
heatmap = tf.squeeze(heatmap)

# For visualization purpose, we will also normalize the heatmap between 0 & 1
heatmap = tf.maximum(heatmap, 0) / tf.math.reduce_max(heatmap)
return heatmap.numpy()
```


https://keras.io/examples/vision/grad_cam/