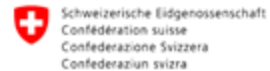


DIGITAL FINANCE

This project has received funding from the Horizon Europe research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 101119635



State Secretariat for Education,
Research and Innovation SERI



**Funded by
the European Union**



Deep Learning xAI

Faizan Ahmed



Funded by
the European Union

Deep Learning

- A type of machine learning based on artificial neural networks in which multiple layers of processing are used to extract progressively higher-level features from data.
- Unlike traditional models, deep learning doesn't need manually engineered features: it learns from raw input, layer by layer.
 - Learns filters and features from scratch (e.g., from image pixels).
 - Handles high-dimensional and unstructured data like images, audio, and text.
- Therefore, how to process the pixels of an image, which filters to use, which feature is important, etc. all of this is learnt from scratch.
- Some examples:
 - Dense Neural Networks
 - Convolution Neural Networks - (Too many flavours to list them all)
 - (Variational) Auto Encoders
 - Generative Adversarial Networks (GAN) - (Too many flavours to list them all)
 - Graph Neural Networks



XAI for Deep Learning

- Input output mapping is beyond human perception
 - Many layers of multiplication
 - Millions of weights
 - Multiple non-linear transformations
- The main idea:
 - The methods visualize features and concepts learned by a neural network, explain individual predictions and simplify neural networks.

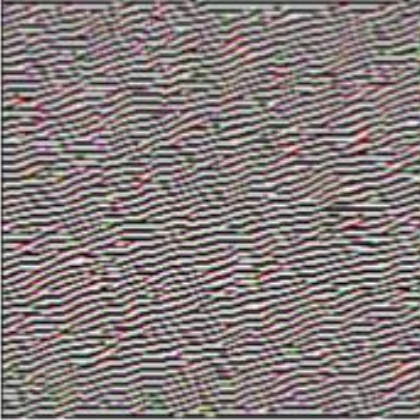


XAI for Deep Learning

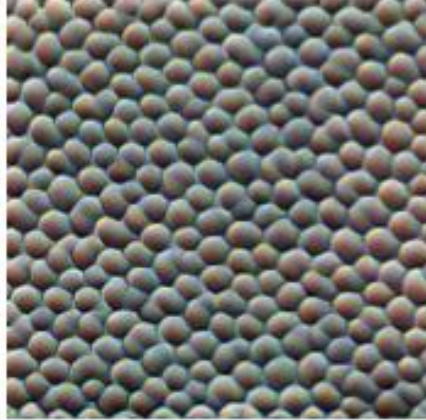
- Most model agnostic methods such as local models or partial dependence plots are applicable
- Why special XAI methods for deep learning
- neural networks learn features and concepts in their hidden layers, and we need special tools to uncover them
- the gradient can be utilized to implement interpretation methods that are more computationally efficient than model-agnostic methods that look at the model “from the outside”



Edges



Textures



Patterns



Parts



Objects



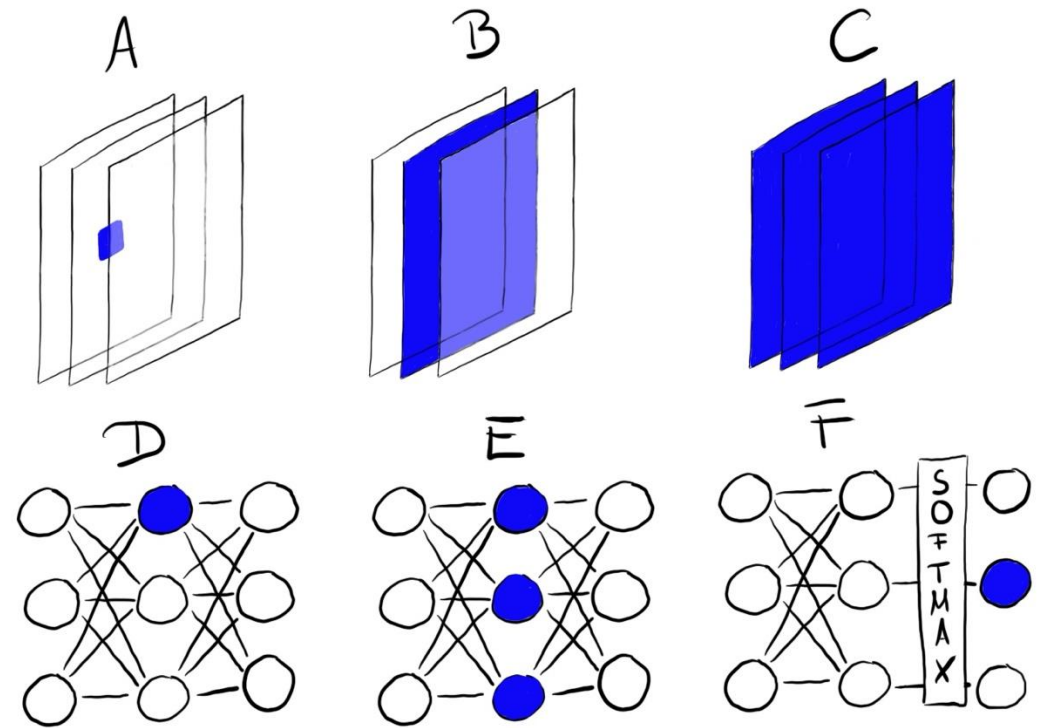
Learned Features

- The first convolutional layer(s) learn features such as edges and simple textures.
- Later convolutional layers learn features such as more complex textures and patterns.
- The last convolutional layers learn features such as objects or parts of objects.
- The fully connected layers learn to connect the activations from the high-level features to the individual classes to be predicted.



Feature Visualization

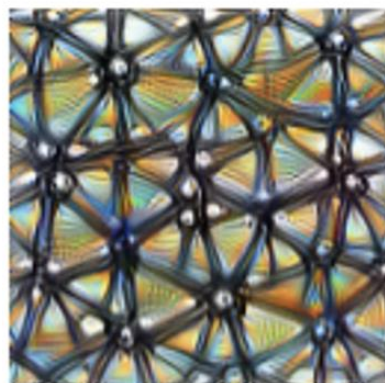
- Feature visualization reveals learned features by finding the input that maximally activates a specific neural network “unit”
 - Convolution neuron,
 - Convolution channel, Convolution layer, Neuron,
 - Hidden layer,
 - Class probability neuron (or corresponding pre-softmax neuron)
- Mathematically: Give a new image that maximizes the (mean) activation of a unit





Neuron

`layern[x,y,z]`



Channel

`layern[:, :, z]`



Layer/DeepDream

`layern[:, :, :]2`



Class Logits

`pre_softmax[k]`



Class Probability

`softmax[k]`

Feature Visualization

Give a new image that maximizes the (mean) activation of a unit

- **Individual Neurons:** Offer the most detailed insights but impractical to analyze.

$$img^* = \arg \max_{img} h_{n,x,y,z}(img)$$

- **Channels (Feature Maps):** Useful for feature visualization. Good balance between useability and computation efforts

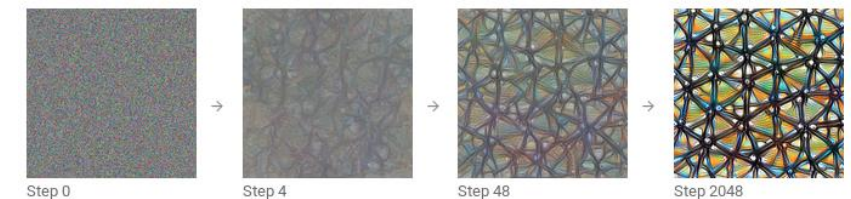
$$img^* = \arg \max_{img} \sum_{x,y} h_{n,x,y,z}(img)$$

- **Alternatively:** Use weighted sum
- **Entire Layers:** Used in applications like Google's DeepDream, enhancing the input image with layer-specific features for a dream-like effect.



Feature Visualization

- **Positive vs Negative activation**
 - Exploring how maximizing and minimizing activations affects a neural network unit can reveal distinct features that either strongly activate or deactivate that particular unit.
- Which images to use:
 - **Using Existing Images:** Search through training images to find those that maximize activation, Limitation: this might not fully reveal what the network focuses on due to correlations in the data.
 - **Generating New Images:** Start from random noise,
 - applying constraints to ensure only subtle changes, utilizing techniques like jittering, rotation, scaling, and regularization (e.g., frequency penalization).



Feature Visualization

Advantages

- **Qualitative insight** into what individual units respond to (edges, textures, object parts)
- **Hypothesis generation** for which features a network might use
- **Guides further analysis** (e.g. suggests concepts to test with Network Dissection)

Disadvantages

- **No proof-of-concept learning:** seeing “skyscraper-like” patterns doesn’t guarantee the unit actually detects skyscrapers
- **Lacks a quantitative score** for how reliably a unit detects a given concept
- **Entanglement risk:** real concepts often spread across many channels, so single-unit visualizations can be misleading



Network Dissection

- **Disentangled feature:** A single unit/channel \rightarrow a single real-world concept
 - e.g. channel 394 \rightarrow skyscrapers, 121 \rightarrow dog-snouts, 12 \rightarrow 30° stripes
- **Entangled features:** concepts spread across many channels

Assumption Feature Visualization: Units of a neural network (like convolutional channels) learn disentangled concepts.

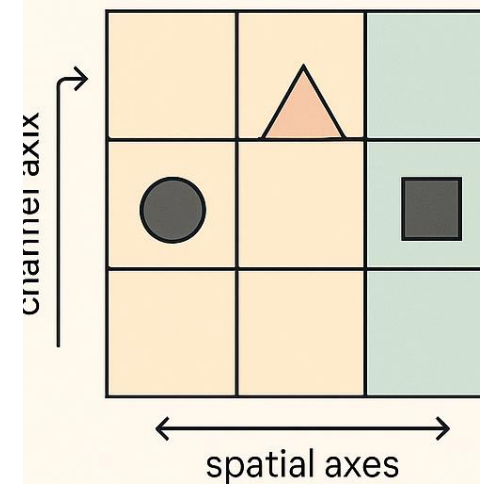
Problem: Convolutional neural networks are not perfectly disentangled.

A way to go: quantify the interpretability of a unit of a convolutional neural network.

- links highly activated areas of CNN channels to human-understandable concepts like objects, colors, and textures.

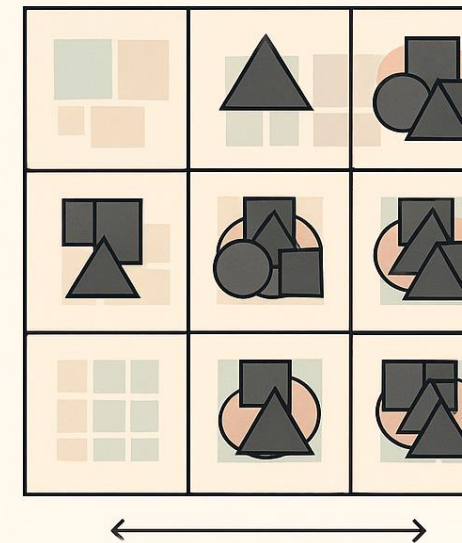


Disentangled feature maps



CNN feature map slice

Entangled feature maps



CNN feature map slice





Network Dissection

Get	Get images with human-labeled visual concepts, from stripes to skyscrapers
Measure	Measure the CNN channel activations for these images.
Quantify	Quantify the alignment of activations and labeled concepts.



DIGITAL



-  = Human annotated ground truth
-  = Top activated area
-  = Area of Intersection
-  = Area of Union

Network Dissection

- **Step 1: (Get)** Broden dataset
- **Step 2: (Measure)** Retrieve network activations
- **Step 3: (Quantify)** Activation-concept alignment



Algorithm 1: Network Dissection Process

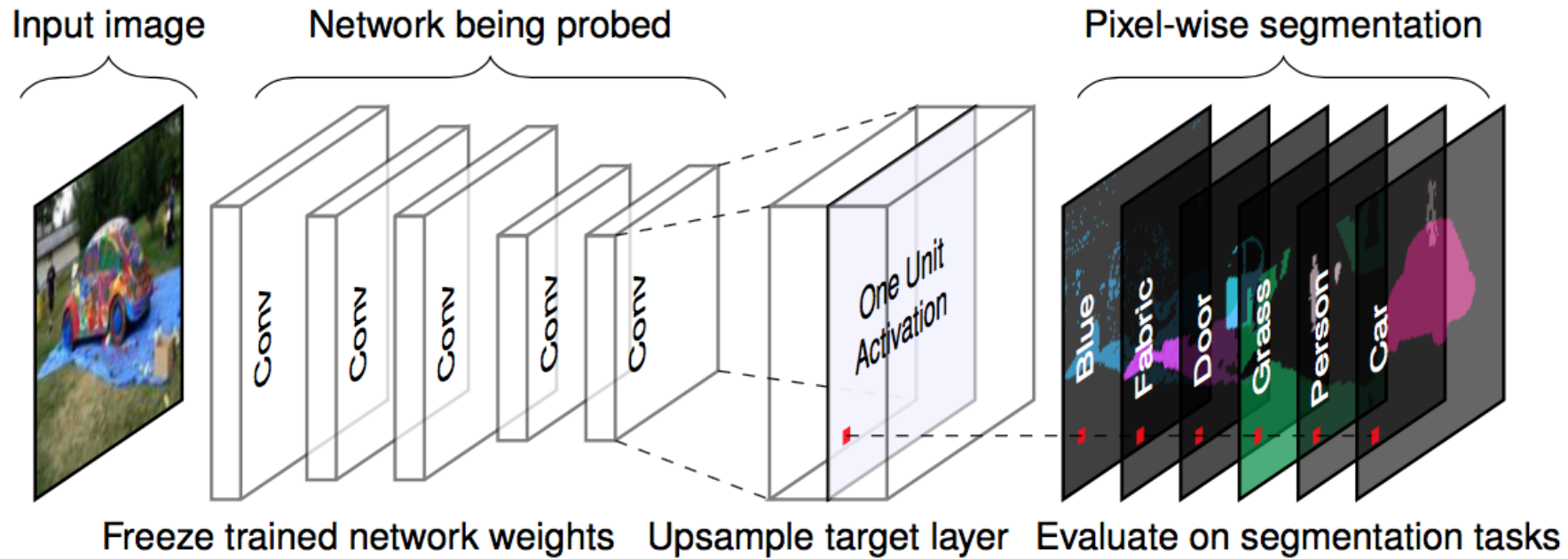
```
/* Get - Data collection */
1
2 Collect images with human-labelled visual concepts.
3 Use the Broden dataset for maximum concept diversity.

/* Measure - Retrieve network activations */
4
5 foreach convolutional channel  $k$  do
6   foreach image  $x$  in Broden do
7     Forward-propagate  $x$  to the target layer containing  $k$ ;
8     Record pixel activations  $A_k(x)$ .
9   Compute the global activation distribution  $\alpha_k$  over all images;
10  Let  $T_k$  be its 99.5-percentile threshold.
11  foreach image  $x$  in Broden do
12    Upsample  $A_k(x)$  to the resolution of  $x$ ;
13    Binarise:  $M_k(x) \leftarrow 1(A_k(x) \geq T_k)$ .

/* Quantify - Activation-concept alignment */
14
15 foreach channel  $k$  do
16   foreach concept mask  $c$  do
17     Compute  $IoU_{k,c} = \frac{|M_k(x) \cap L_c(x)|}{|M_k(x) \cup L_c(x)|}$ 
18     if  $IoU_{k,c} > 0.04$  then
19       Mark unit  $k$  as a detector for concept  $c$ .
```

Network Dissection

Quantifies the interpretability of a unit of a convolutional neural network



Feature Visualization and Network Dissection



Feature Visualization Complexity

Many feature visualizations are abstract, lacking clear links to understandable human concepts.

Displaying feature visualizations alongside training data often provides limited insight, indicating only general attributes like color presence (e.g., "requires yellow").



Volume of Data

High volume of units makes comprehensive analysis impractical:

- Over 5000 channels across nine layers in Inception V1.
- Visualizing both positive and negative activations, plus training images, could require displaying over 50,000 images.



Interpretability Issues

Feature visualizations can create an illusion of understanding neural network operations.

Complex interactions and lack of clear concept linkage make true interpretability elusive:

- Many units do not correspond to any human-recognizable concept.
- Positive and negative activations often indicate unrelated features.



Network Dissection Limitations

Requires extensively labeled datasets, with each pixel annotated—resource-intensive.

Traditionally aligns only with positive activations, potentially missing insights from negative activations.

Even in detailed architectures like ResNet or Inception, units may respond to the same concept or none, with moderate Intersection over Union (IoU) scores.



Feature Attribution

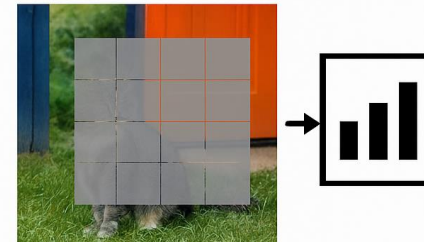
- Many Names: Sensitivity Map, Saliency Map, Pixel Attribution Map, Gradient-Based Attribution Methods, Feature Relevance, Feature Attribution, Feature Contribution.
- General Idea:
 - Given a NN that output $S \in \mathbb{R}^C$ [for regression $C = 1$]
 - For image I we have $S(I) = [S_1(I), \dots, S_C(I)]$
 - Input to feature attribution $x \in \mathbb{R}^p$ (pixel, tabular data, words, etc) with p features
 - Output: Relevance score for each feature $R^c = [R_1^c, \dots, R_p^c]$, where c indicates the relevance for the c^{th} output $S_c(I)$



Pixel Attribution

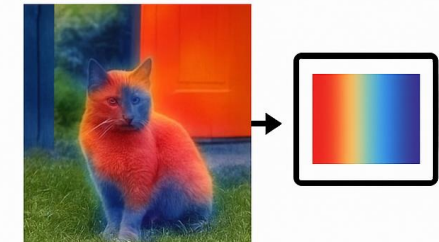
- Feature (pixel) attribution: Explains predictions by attributing relevance to each input feature (pixels, tabular data, words).
- Two major classes(pixel attribution):
 - **Occlusion / perturbation methods** (e.g., SHAP, LIME) explain a model by hiding or altering small regions of the image and observing how the output changes.
 - **Gradient-based methods** (e.g., saliency maps, Guided BP, Grad-CAM) explain a prediction by back-propagating the score to the input pixels and visualising the resulting gradients.

Occlusion / Perturbation



Occlusion / Perturbation

Gradient-based

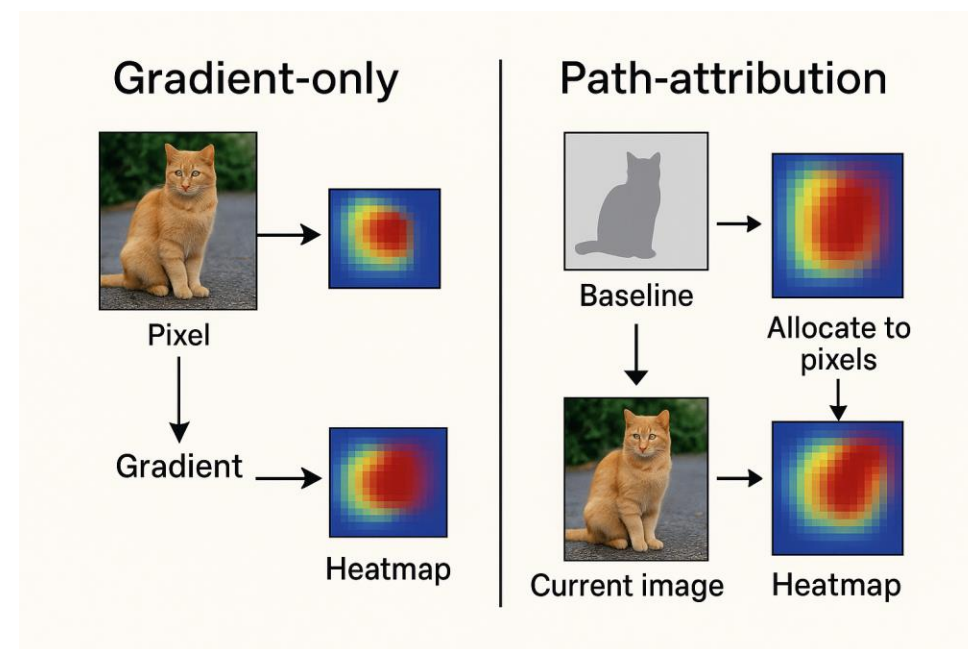


Pixel Attribution

- **Gradient-only** : *Backpropagate once*. The heat-map shows how an infinitesimal change in each pixel would raise (warm colours) or lower (cool colours) the class score.
 - Examples – Vanilla Gradients, Grad-CAM.
 - local sensitivity
- **Path-attribution**: *Compare to a baseline*. Subtract the score of a reference image (often a grey “zero” image) from the current score and divide that difference among the pixels.
 - Examples – Integrated Gradients, Deep Taylor, LIME, SHAP.
 - If the method is *complete*, the pixel scores sum exactly to the score gap.
 - contribution relative to a reference.



Both produce a same-size heat-map you can overlay on the image.



Vanilla Gradient (Saliency Maps)

- Calculates the gradient of the loss function for the desired class relative to input pixels.
- Produces a map showing pixel influence on class prediction, with values ranging from negative to positive.
- Three steps:

- **Forward Pass:** Process the image through the neural network to activate outputs.
- **Gradient Computation:** Calculate the gradient

$$E_{grad}(I_0) = \left. \frac{\delta S_c}{\delta I} \right|_{I=I_0}$$

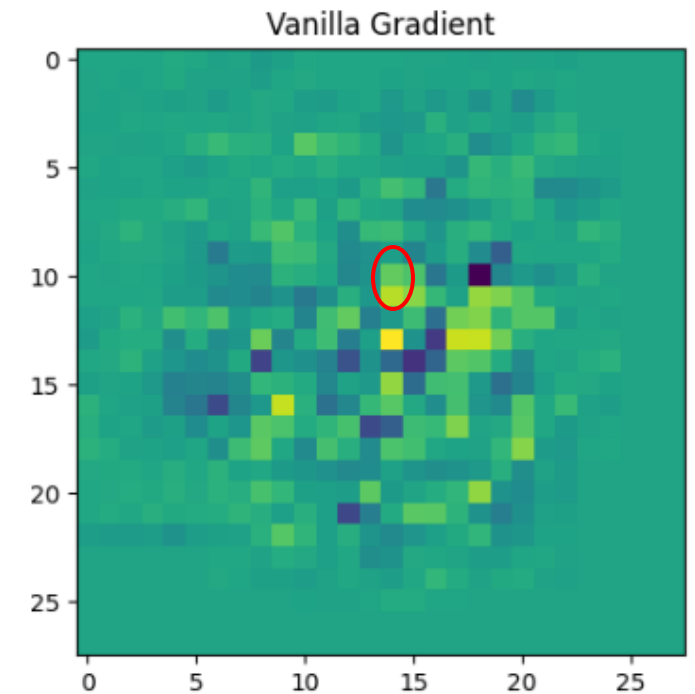
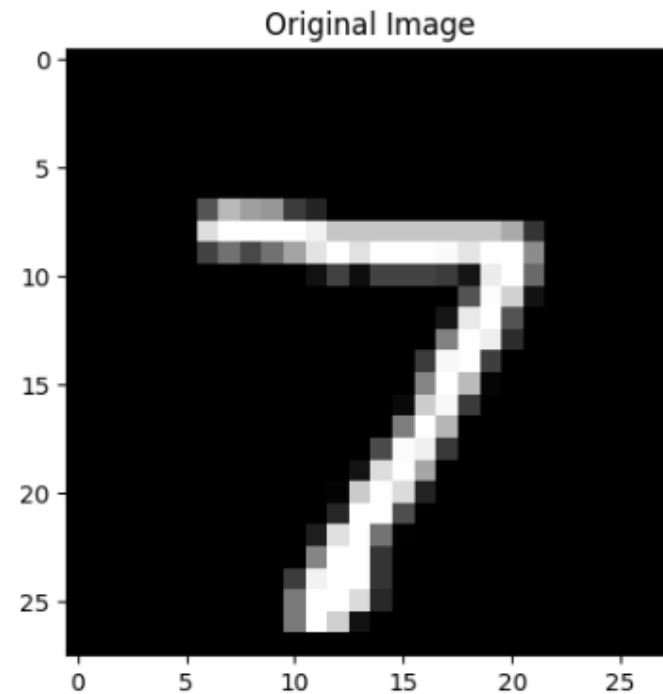
for the class score relative to input pixels.

- This indicates how each pixel's change would affect the class score.
- **Visualization:**
 - Display the gradient map.
 - Options include showing absolute values or highlighting positive and negative influences separately.



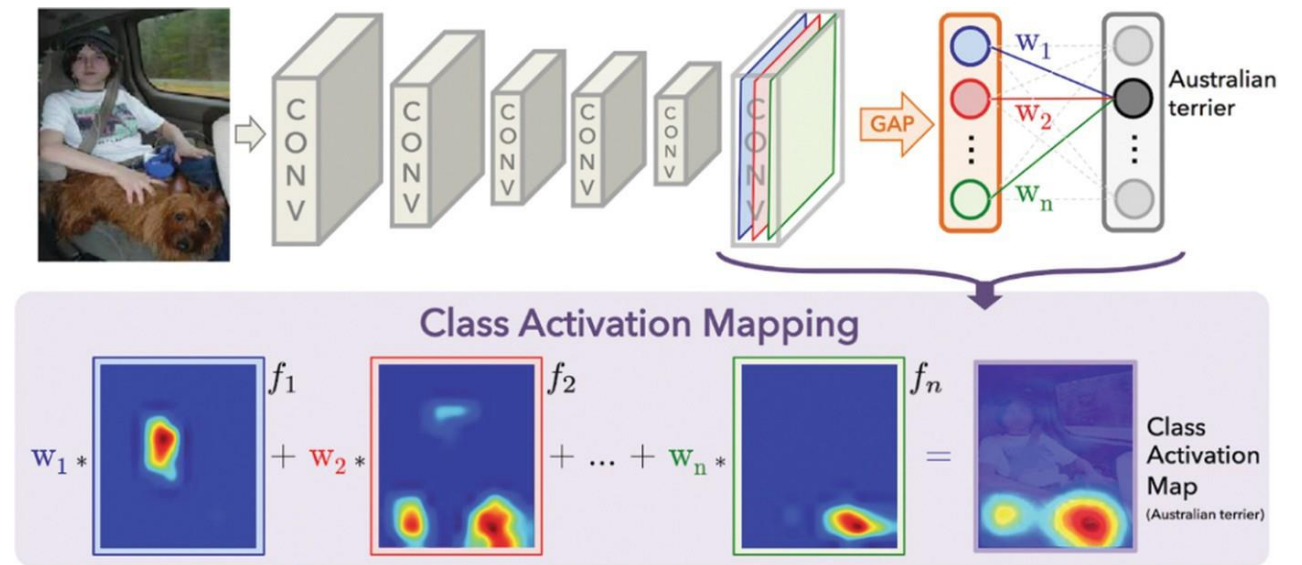
Vanilla Gradient (Saliency Maps)

- Score Approximation: For image I with a score $S_c(I)$ for class c .
 - $S_c(I)$ is nonlinear function. Its Taylor's Approximation
$$S_c(I) \approx \left. \frac{\delta S_c}{\delta I} \right|_{I_0} + b = w^T I + b$$



Class Activation Map (CAM)

- Class Activation Maps (CAM) help us understand which regions of an image affect the output of a convolutional neural network.
- Based on a heatmap that highlights the pixels of the image that push a model to associate a certain class, a certain label, to the image.



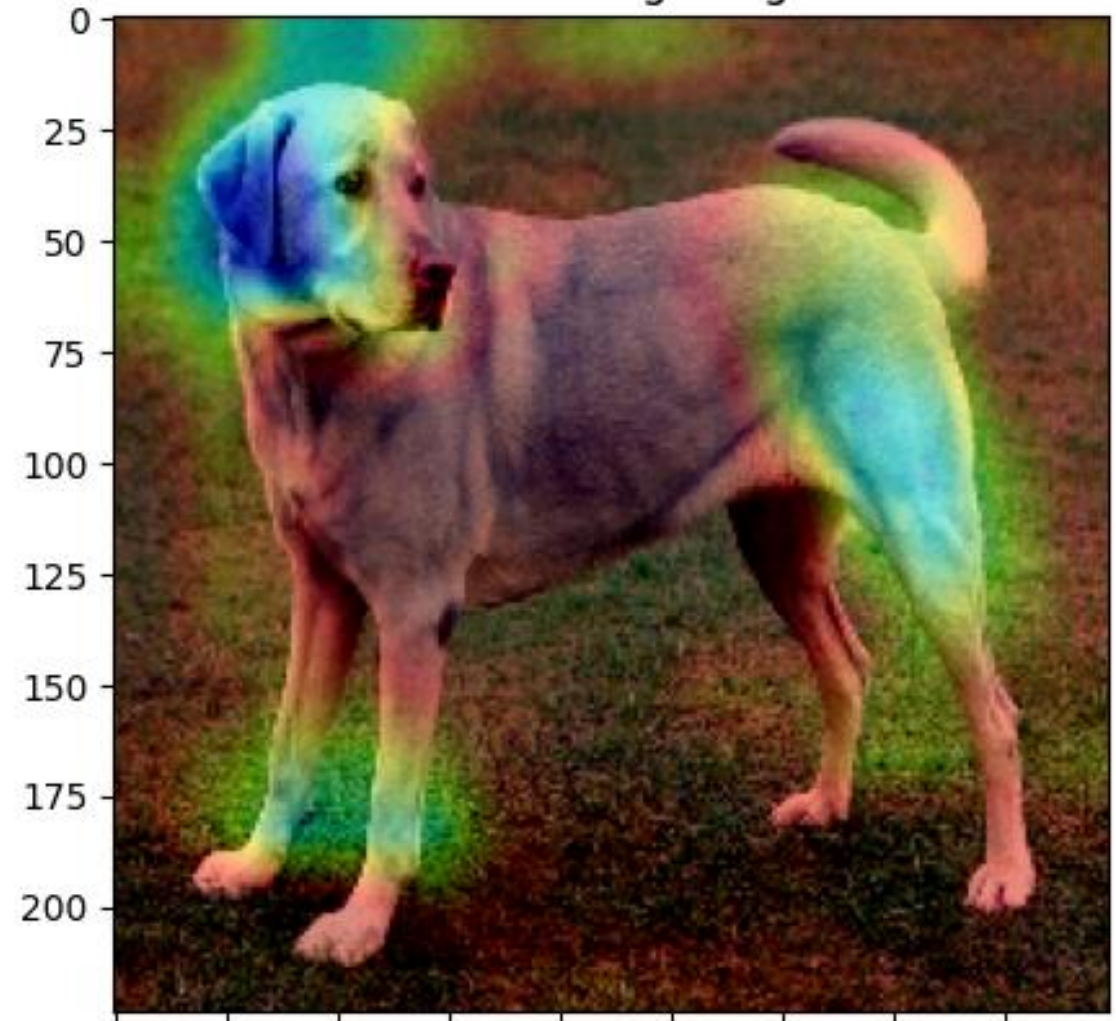
Class Activation Map (CAM)

- CAM procedure
 - remove the last fully connected layers,
 - apply a Global Average Pooling (GAP) to the last convolutional layer
 - Train the weights from the reduced layer to the classes.
- The dimensions of the last convolution layer are (width and height) are same as the original image.
- The linear combination allows us to have the final visualisation.
- Drawback:
 - change the structure of the network and retraining it;
 - it can only be used by applying it only starting from a convolutional layer and therefore is not applicable in all architecture



DIGITAL

CAM on Dog Image



Gradient Class Activation Map (Grad-CAM)

Highlights the regions of an input image that drive a CNN's prediction for a chosen class.

- Look at the *last convolutional layer* (retains spatial layout).
 - Weight each feature-map by how strongly its gradient influences the class score.
 - Sum, apply ReLU → a coarse *heat-map* showing class-specific evidence.
- Works with almost any CNN architecture (no retraining).
 - Fast: one forward + one backward pass.
 - Visually intuitive → great for demos, debugging, reports.



Grad-CAM

Algorithm 2: Gradient-weighted Class Activation Mapping

Input: Image I , trained CNN, target class c

Output: Heat-map L_{GradCAM}

```
1 ; /* Forward pass */
2  $A^k \leftarrow$  feature-maps of the last conv layer;
3  $S_c \leftarrow$  predicted score for class  $c$ ;

4 ; /* Backward pass */
5  $\frac{\partial S_c}{\partial A^k} \leftarrow$  gradients w.r.t. each map;

6 ; /* Channel importance */
7  $\alpha_k = \frac{1}{Z} \sum_{i,j} \frac{\partial S_c}{\partial A_{ij}^k}$  *[r]global average pooling

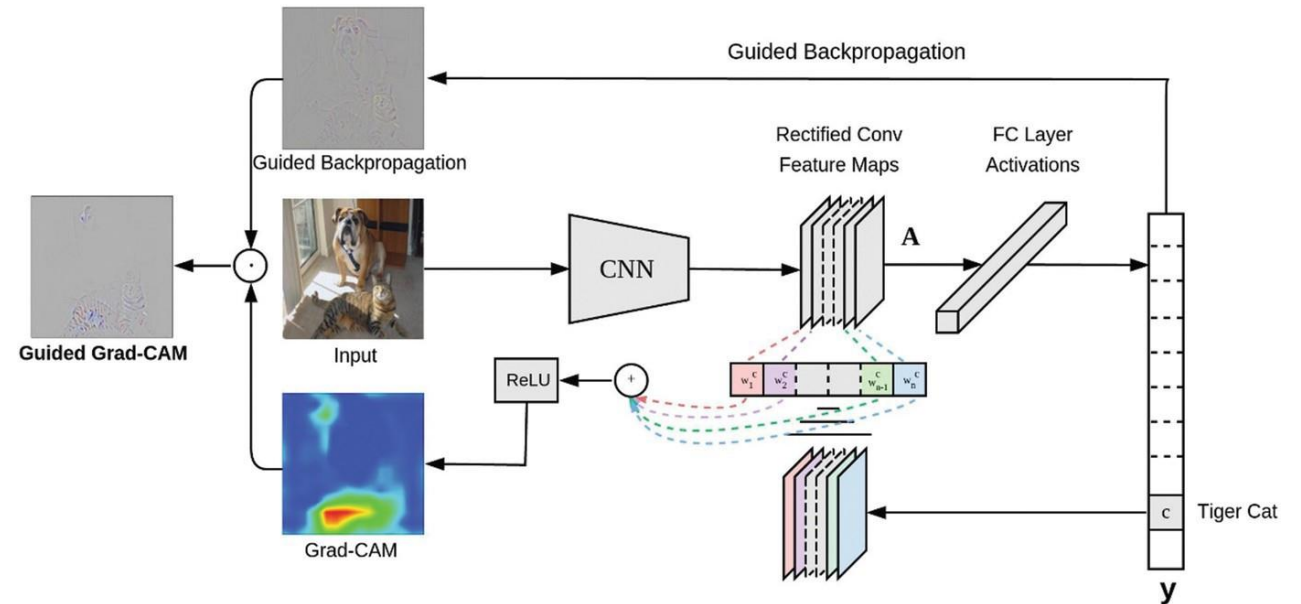
8 ; /* Linear combination & ReLU */
9  $L_{\text{GradCAM}} = \text{ReLU}\left(\sum_k \alpha_k A^k\right)$ ;

10 ; /* Upsample */
11 Resize  $L_{\text{GradCAM}}$  to the resolution of  $I$  and overlay;
```



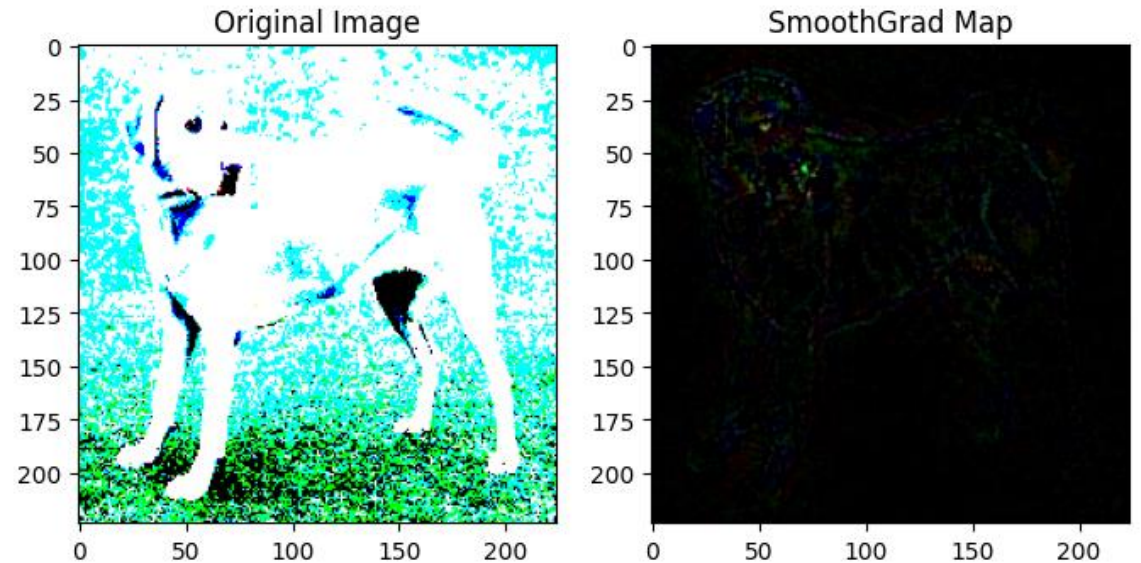
Gradient Class Activation Map (Grad-CAM)

- An evolution of the CAM is the Grad-CAM.
- Grad-CAM does not retrain the network.
- Grad-CAM's explanations can suffer from gradient problems.

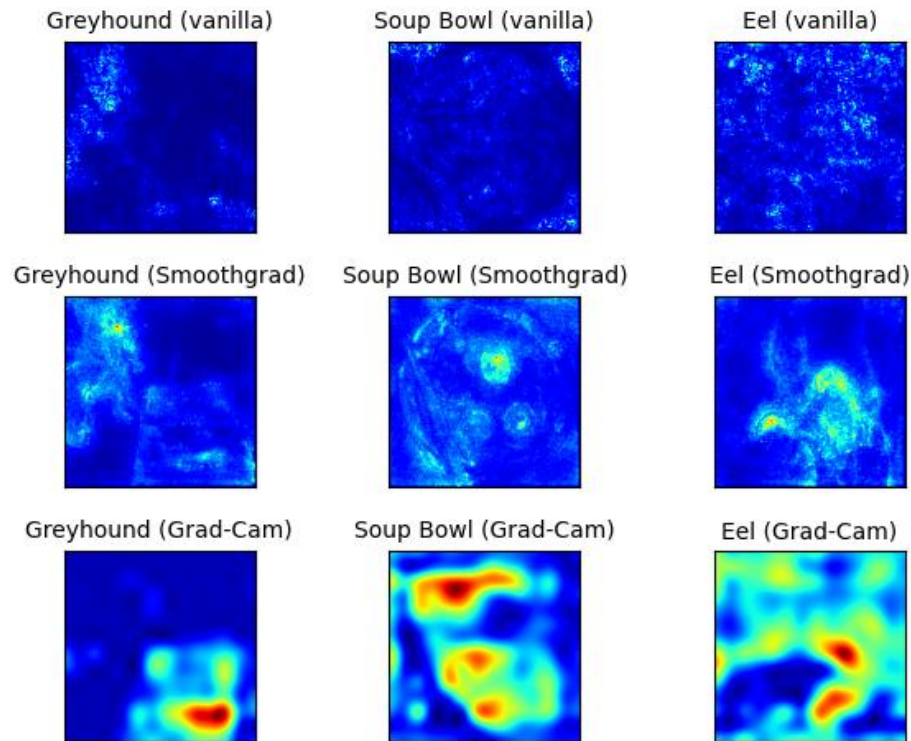


Smooth Gradient

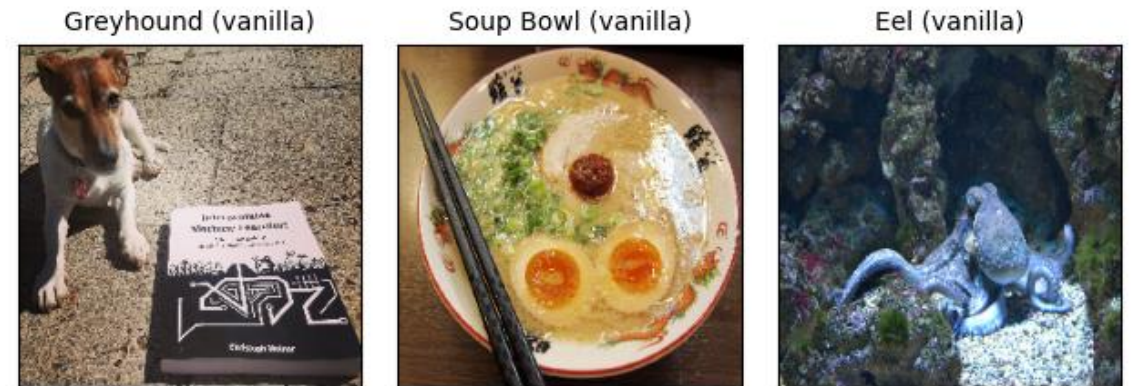
- **A generic method**
 - **Noise Addition:** Generate multiple versions of the target image by adding Gaussian noise.
 - **Pixel Attribution:** Create pixel attribution maps for each noisy image.
 - **Averaging:** Average these maps to produce a smoother, less noisy gradient visualization.
- Derivatives in neural networks can be noisy due to small-scale fluctuations.
- Averaging over several noisy samples reduces these fluctuations, leading to clearer insights.



Can we trust these explanations?



- Image on the left classified as “Greyhound” with a probability of 35%.
- The middle image Correctly identified as “Soup Bowl” with a probability of 50%.
- The right image incorrectly classified as “Eel” with a high probability of 70%.



Definition of Gradient of a Function of Two Variables

Let $z = f(x, y)$ be a function of x and y such that f_x and f_y exist. Then the **gradient of f** , denoted by $\nabla f(x, y)$, is the vector

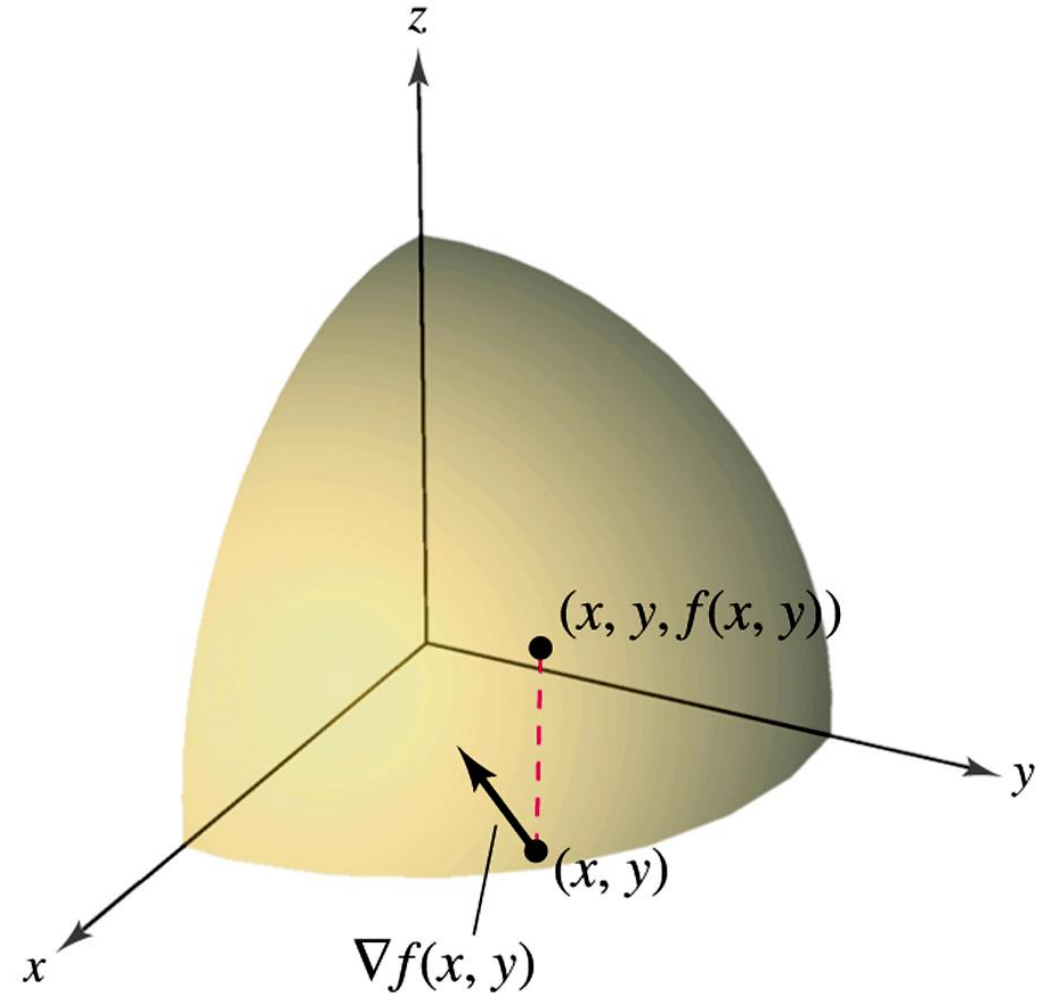
$$\nabla f(x, y) = f_x(x, y)\mathbf{i} + f_y(x, y)\mathbf{j}.$$

∇f is read as “del f .” Another notation for the gradient is **grad** $f(x, y)$. In Figure 13.48, note that for each (x, y) , the gradient $\nabla f(x, y)$ is a vector in the plane (not a vector in space).

THEOREM 13.10 Alternative Form of the Directional Derivative

If f is a differentiable function of x and y , then the directional derivative of f in the direction of the unit vector \mathbf{u} is

$$D_{\mathbf{u}}f(x, y) = \nabla f(x, y) \cdot \mathbf{u}.$$



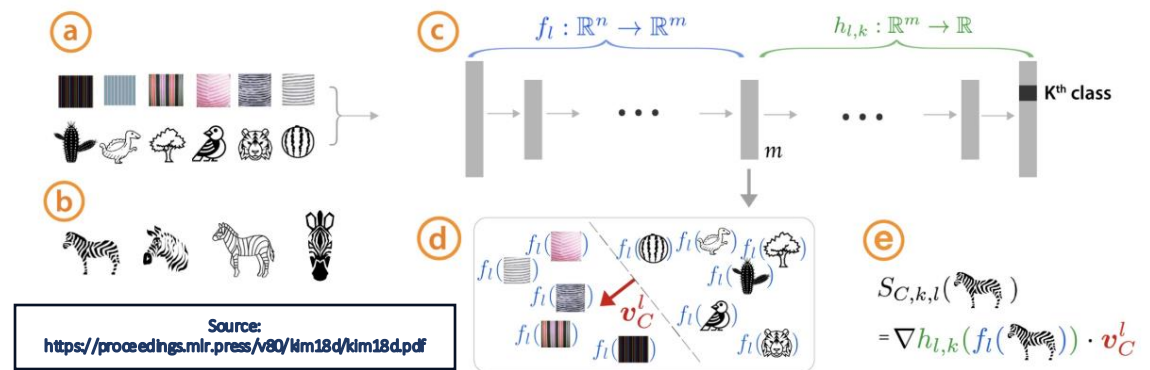
Concept Activation Vectors

- Mapping data to human interpretable concepts via vectors.
- For features/pixels

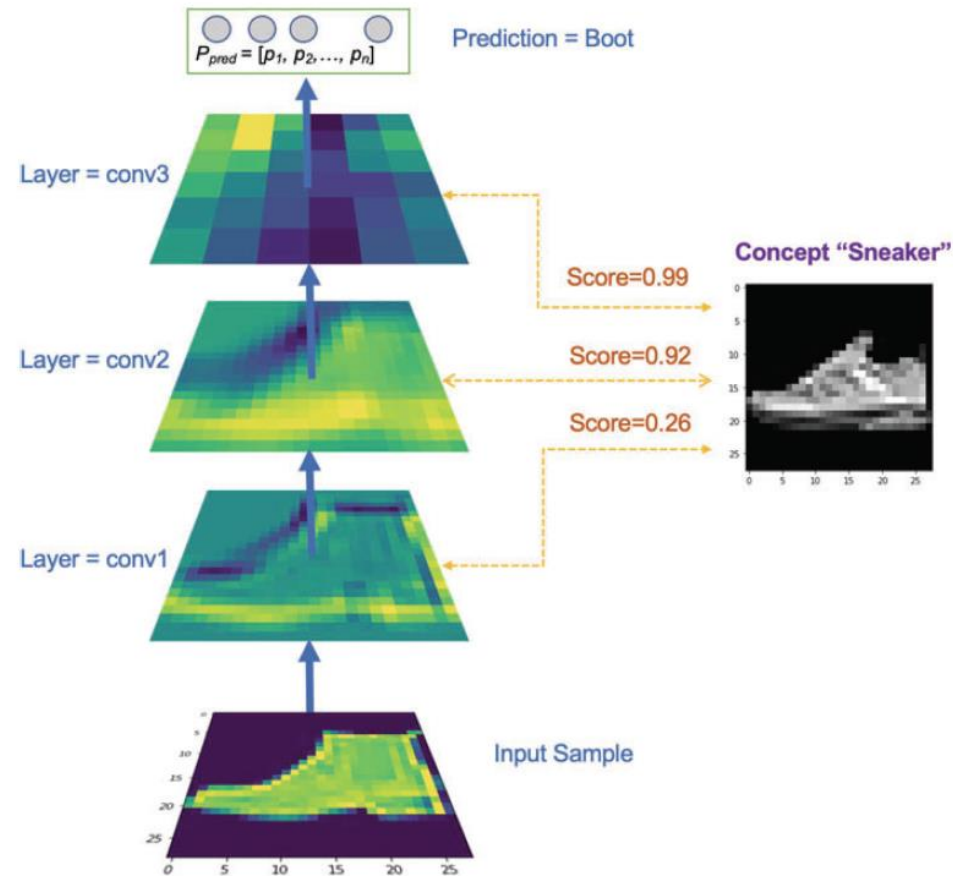
$$\frac{\partial h_k(x)}{\partial x_{a,b}}$$

- $v_C^l \in \mathbb{R}^m$ unit CAV for concept C in layer l , $f_l(x)$ the activation for input x at layer l

$$S_{C,k,l}(x) = \nabla h_{l,k}(f_l(x)) \cdot v_C^l$$



Concept Activation Vectors



Adversarial Examples



Safety-critical: stop-sign spoof → self-driving car misses the stop.



Security: spam mails engineered to dodge filters.



Physical screening: weapon disguised as umbrella in X-ray.

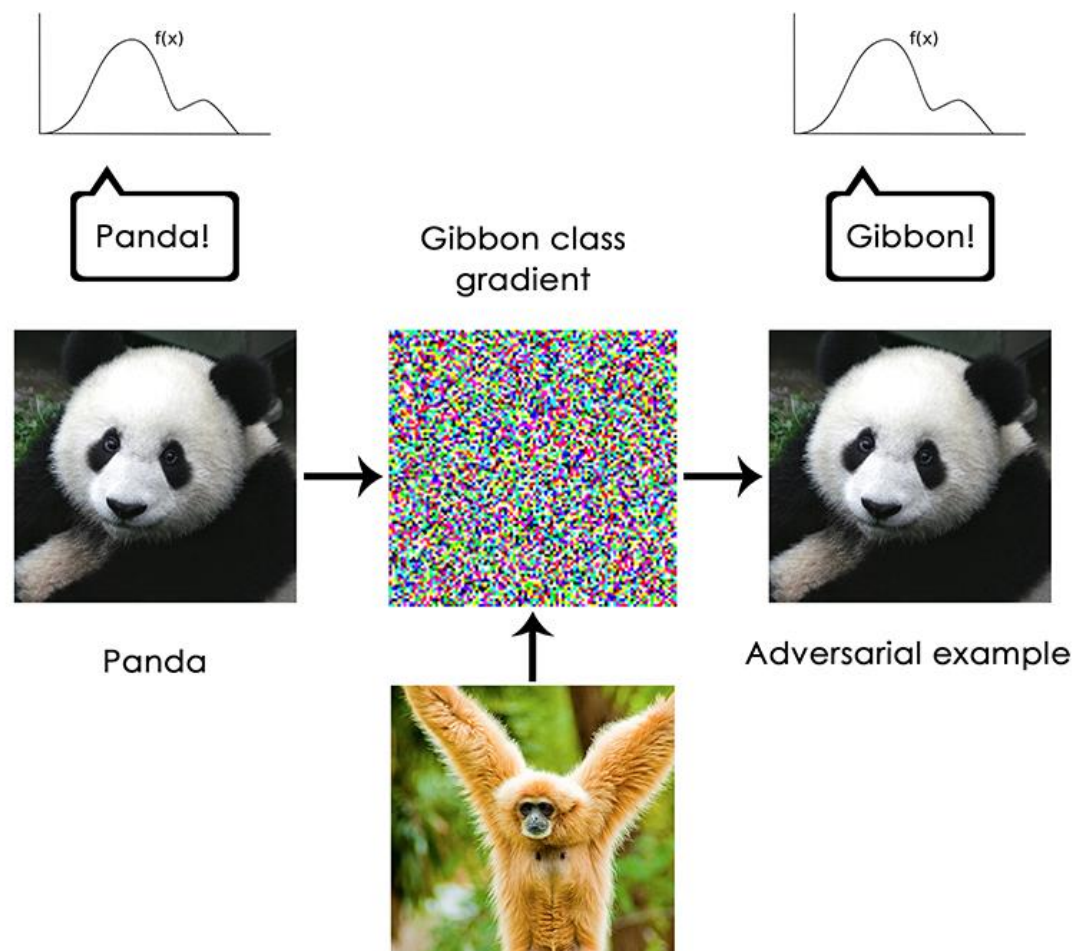


Adversarial inputs are a real **attack surface**, not a lab curiosity.



Gradient-based optimization attacks

- Use full gradients +L-BFGS to solve $\min_r \lambda \|r\|_2 + \text{loss}(f(x+r), y_t)$
 - Produces high-quality but slow adversarial images
- One-step attack (Goodfellow et al., 2015)
$$x^* = x + \epsilon \text{sign}(\nabla_x J(\theta, x, y_{\text{true}}))$$
 - Adds / subtracts ϵ per pixel \rightarrow instant adversary.
 - Fooling GoogLeNet at 99 % confidence.

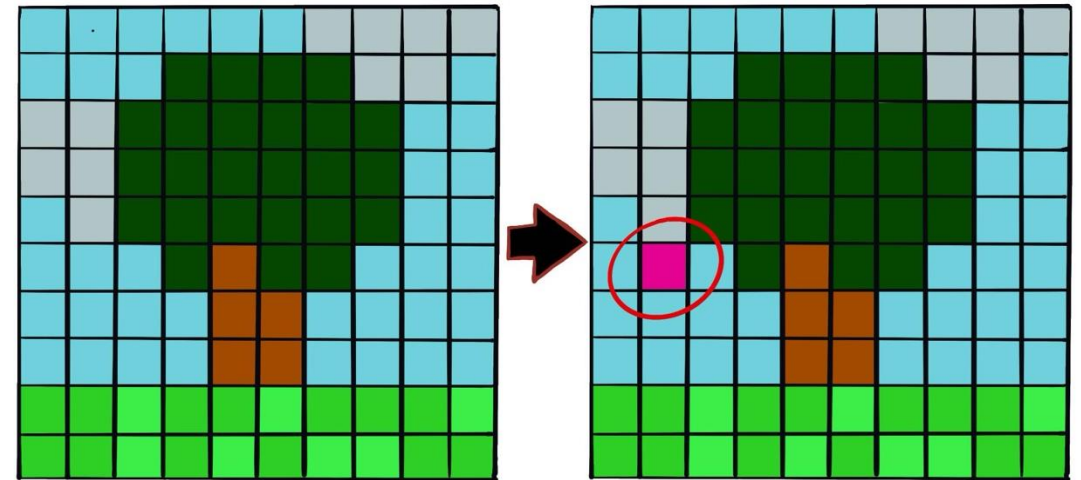


Adversarial Examples

One-pixel attack (Su et al., 2019):
differential evolution finds *one* RGB pixel
that flips the label.

Adversarial patch (Brown et al., 2018):
printable sticker forces any scene to be
predicted as a toaster – see banana
example (third image).

Removes the “imperceptible” constraint;
focuses on real-world deployability.



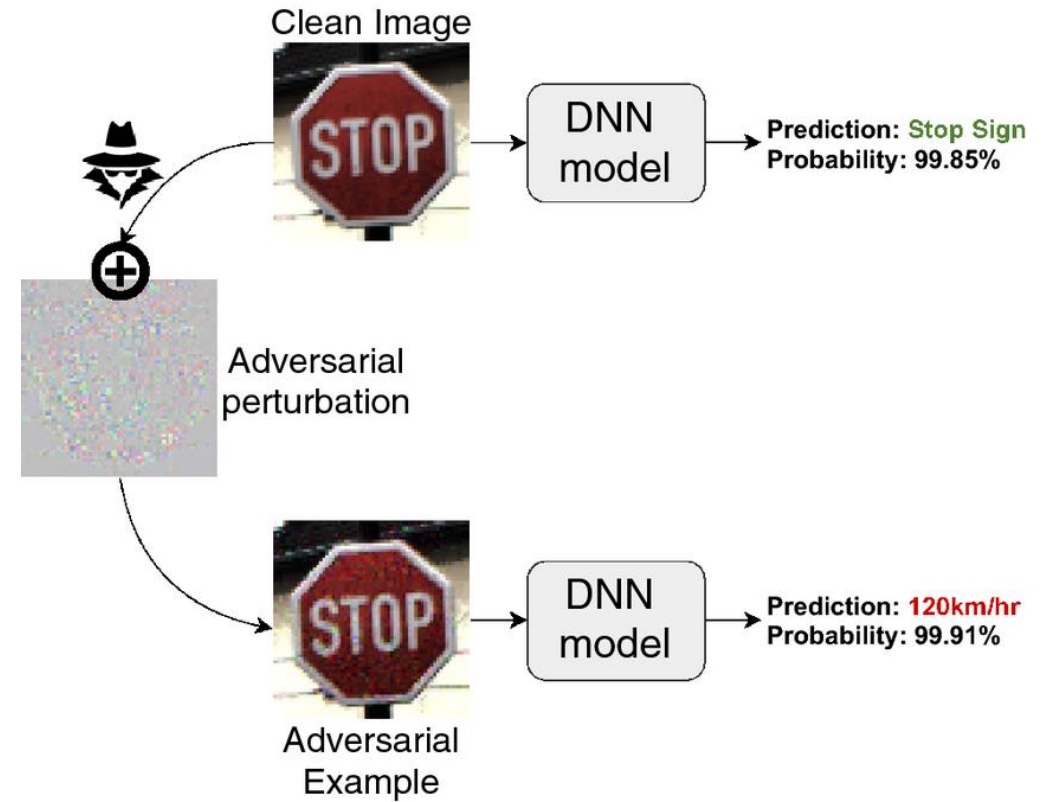
Why This matters?

Hardening tactics

- Adversarial training: iteratively retrain on crafted examples.
- Regularisation & robust optimisation (e.g., PGD adversarial training).
- Ensembles & randomisation (limited gains).
- Proactive testing: treat ML like cyber-security, seek unknown-unknowns.

Reality check

- No silver bullet; arms race between attackers & defenders.
- Interpretability tools help spot fragile features before adversaries do



Libraries

Source: Kamath, Uday, and John Liu. *Explainable artificial intelligence: an introduction to interpretable machine learning*. Vol. 2. Cham: Springer, 2021.

Model/Algorithm	Library
Attention (NMT)	TensorFlow
Attention (image captioning)	TensorFlow
LIME	Captum (PyTorch)
Occlusion	Captum (PyTorch)
RISE	Keras
Activation Maximization	tf-keras-vis
Saliency map	Captum (PyTorch)
DeepLIFT	Captum (PyTorch)
DeepSHAP	Captum (PyTorch)
Deconvolution	Captum (PyTorch)
Guided Backprop	Captum (PyTorch)
Integrated Gradients	Captum (PyTorch)
Layer-wise relevance propagation	Captum-0.4.0 (PyTorch)
Excitation backpropagation	excitationbp (PyTorch)
GradCAM	Captum (PyTorch)
TCAV	Captum (PyTorch)





DIGITAL



**Funded by
the European Union**

Funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or Horizon Europe: Marie Skłodowska-Curie Actions. Neither the European Union nor the granting authority can be held responsible for them.



DIGITAL

This project has received funding from the European Union's Horizon Europe research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 101119635