DIGITAL FINANCE

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

Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

State Secretariat for Education, Research and Innovation SERI

Funded by the European Union

MARIE CURIE ACTIONS

DIGITAL

Deep Learning xAI

Faizan Ahmed

DIGITAL

Funded by
the European Union

Deep Learning

- Learns features directly from raw data (no manual feature design).
- Works well with images, audio, text, and high-dimensional data.
- Why it's powerful:
  - Learns filters/features automatically.
  - Decides which parts of the input are important.
- Examples:
  - Dense Neural Networks (DNNs)
  - Convolutional Neural Networks (CNNs)
  - Autoencoders (AE, VAE)
  - Generative Adversarial Networks (GANs)
  - Graph Neural Networks (GNNs)

DIGITAL

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"

DIGITAL

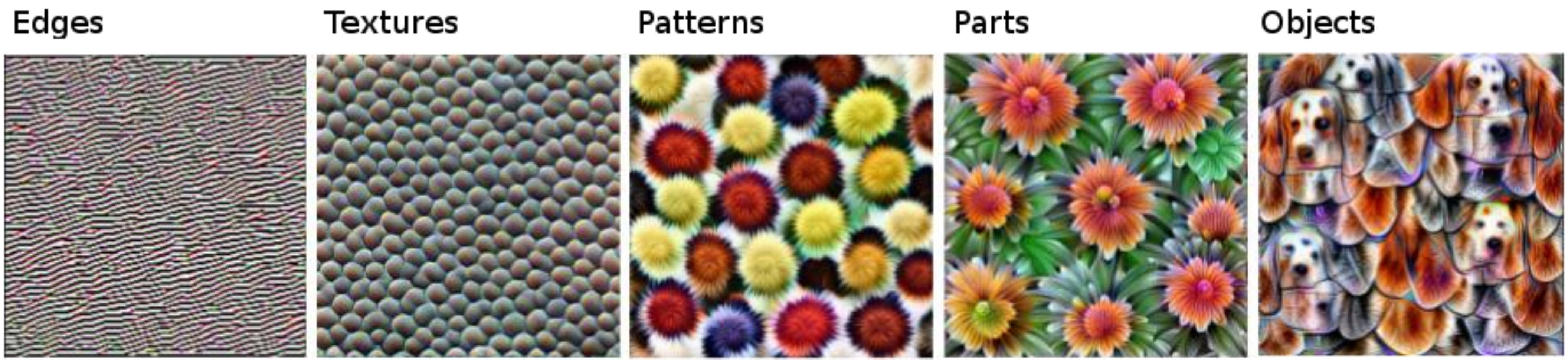
XAI for Deep Learning

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.
Two main questions	• Which features are learned and where? • How to visualize them?

DIGITAL

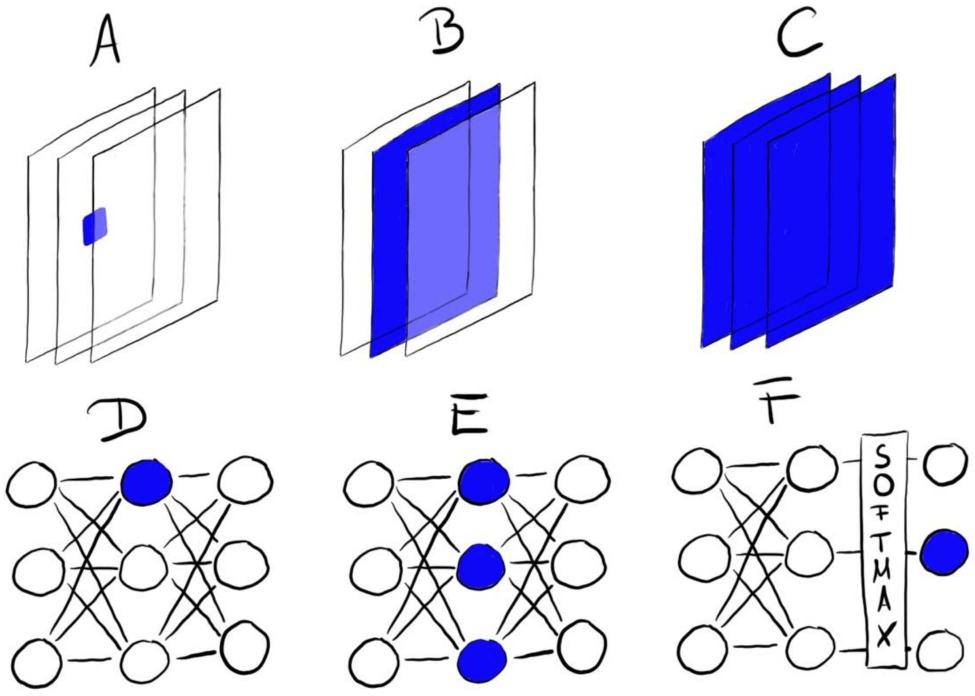
Learned Features

- First layers: edges & colors
- Middle layers: textures & shapes
- Deep layers: object parts & concepts



DIGITAL

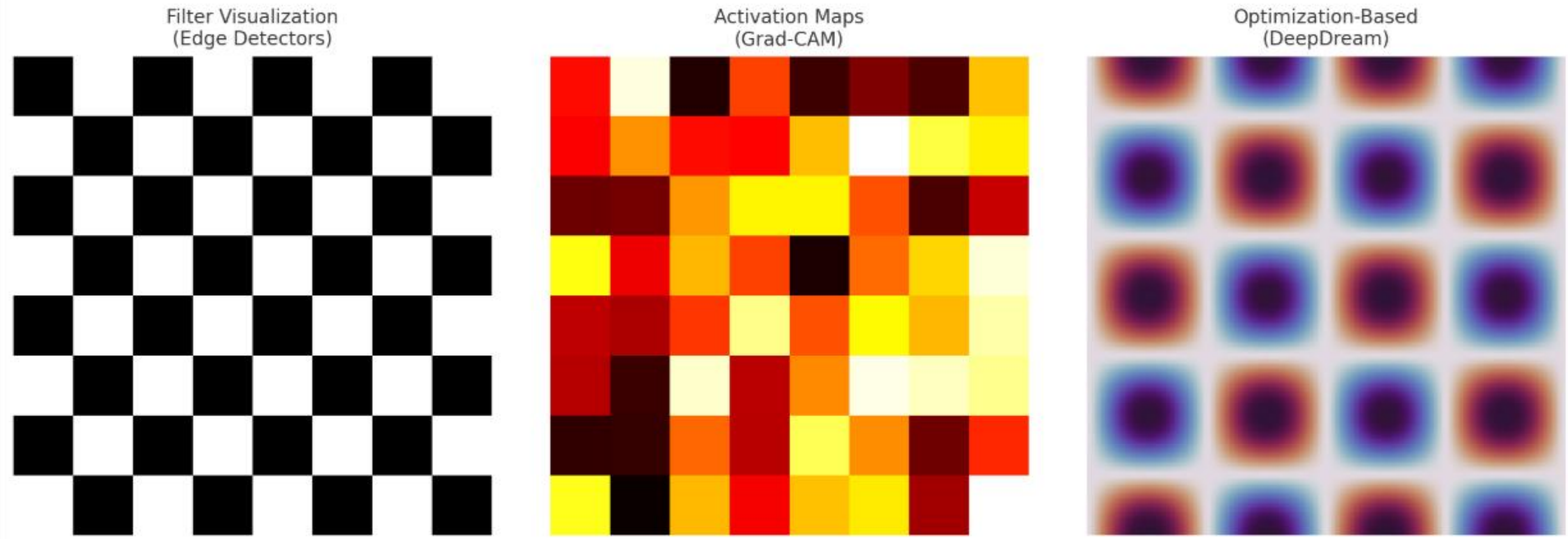
Feature Visualization



- Feature visualization reveals learned features by finding the input that maximally activates a specific neural network "unit".
- "Unit"
  - Convolution neuron,
  - Convolution channel, Convolution layer, Neuron,
  - Hidden layer,
  - Class probability neuron (or corresponding pre-softmax neuron)

DIGITAL

Feature Visualization Methods



DIGITAL

Optimization-based

- 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.

DIGITAL

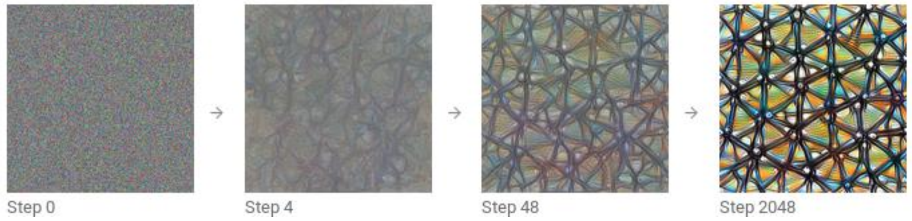
Optimization based

Neuron	Channel	Layer/DeepDream	Class Logits	Class Probability
$\texttt{layer}_n\texttt{[x,y,z]}$	$\texttt{layer}_n\texttt{[:,:,z]}$	$\texttt{layer}_n\texttt{[:,:,:]}^2$	pre_softmax[k]	softmax[k]

Source: https://distill.pub/2017/feature-visualization/

DIGITAL

Optimization based



- Maximizing vs. minimizing activations shows what excites or suppresses a neuron/unit.
- Ways to find images:
  - Existing images: find samples that strongly activate a u
  - Limitation: may reflect dataset correlations, not true f
  - Generated images: optimize from noise with constrai
  - Techniques: jitter, rotation, scaling, regularization.

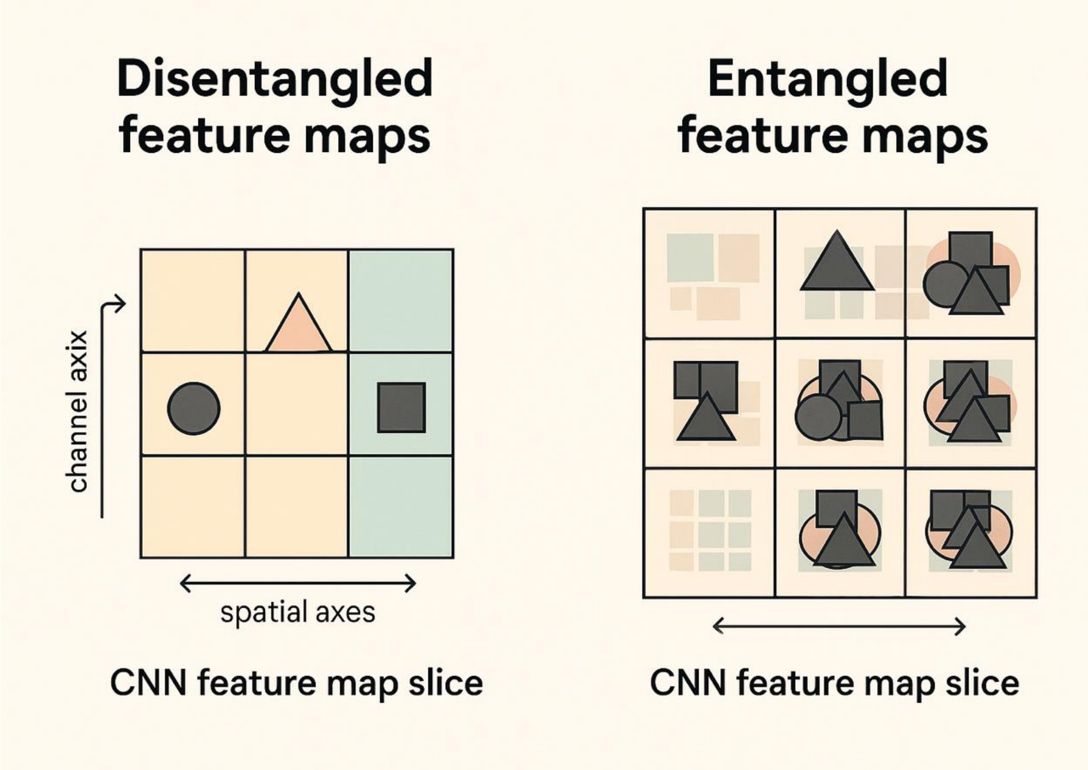
DIGITAL

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

DIGITAL

Network Dissection



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

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

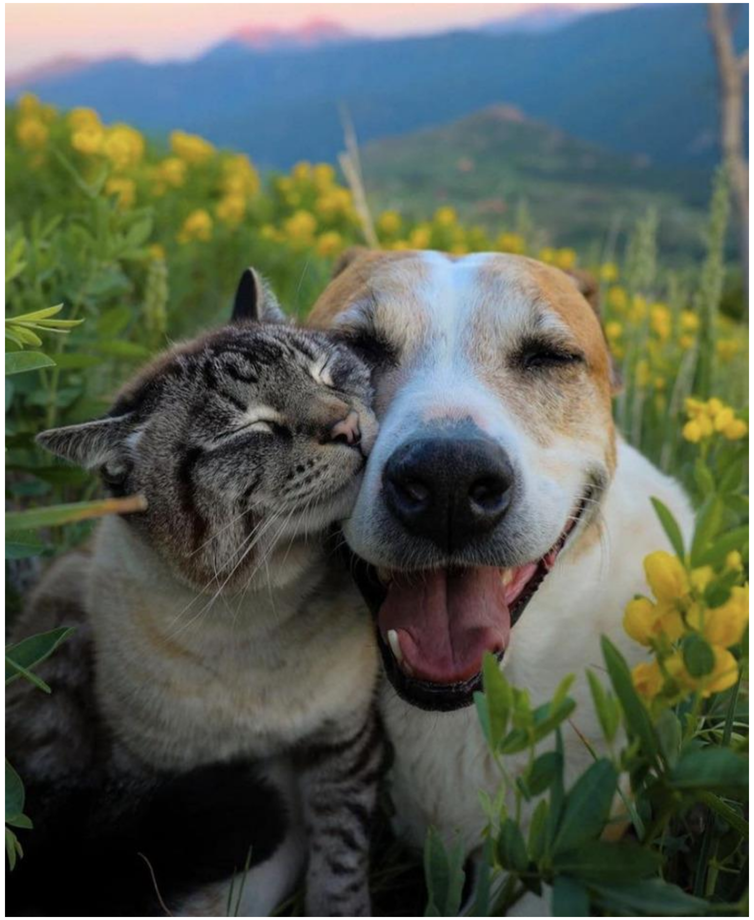
DIGITAL

Network Dissection

Convolutional neural networks are not perfectly disentangled.

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

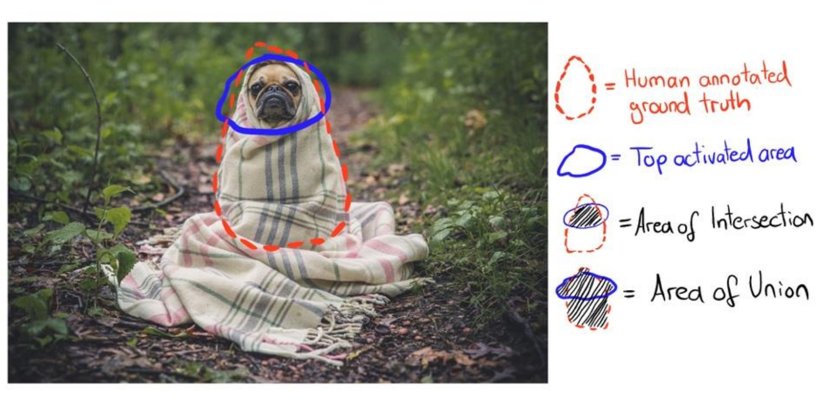
- links highly activated areas of CNN channels to human-understandable concepts like objects, colors, and textures.



DIGITAL

Network Dissection

Get	Get images with human-labeled visu skyscrapers
Measure	Measure the CNN channel activations for these images.
Quantify	Quantify the alignment of activations and labeled concepts.



DIGITAL

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 */`

Collect images with human-labelled visual concepts.
Use the *Broden* dataset for maximum concept diversity.

`/* Measure - Retrieve network activations */`

foreach *convolutional channel* k **do**
    foreach *image* x *in Broden* **do**
        Forward-propagate x to the target layer containing k;
        Record pixel activations $A_k(x)$.
    Compute the global activation distribution α_k over all images;
    Let T_k be its 99.5-percentile threshold.
    foreach *image* x *in Broden* **do**
        Upsample $A_k(x)$ to the resolution of x;
        Binarise: $M_k(x) \leftarrow 1\big(A_k(x) \geq T_k\big)$.

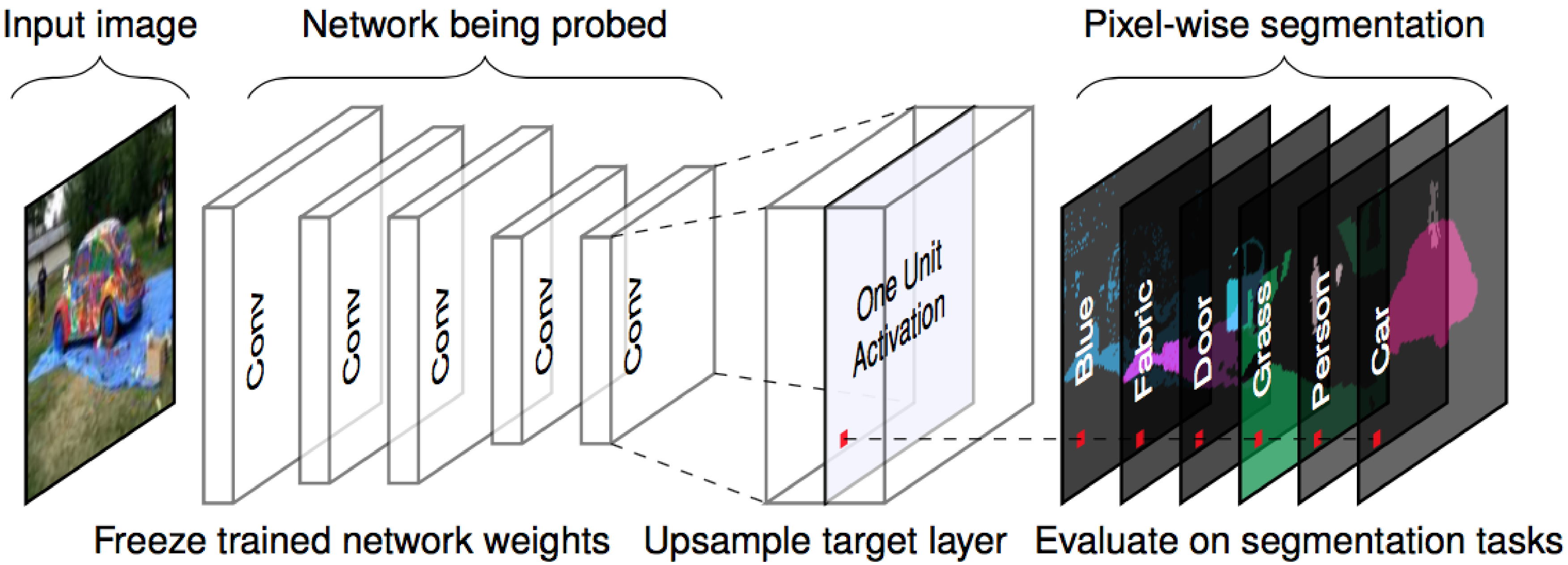
`/* Quantify - Activation-concept alignment */`

foreach *channel* k **do**
    foreach *concept mask* c **do**
        Compute $IoU_{k,c} = \frac{|M_k(x) \cap L_c(x)|}{|M_k(x) \cup L_c(x)|}$
        if $IoU_{k,c} > 0.04$ **then**
            Mark unit k as a detector for concept c.

DIGITAL

Network Dissection

quantifies the interpretability of a unit of a convolutional neural network.



DIGITAL

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.

DIGITAL

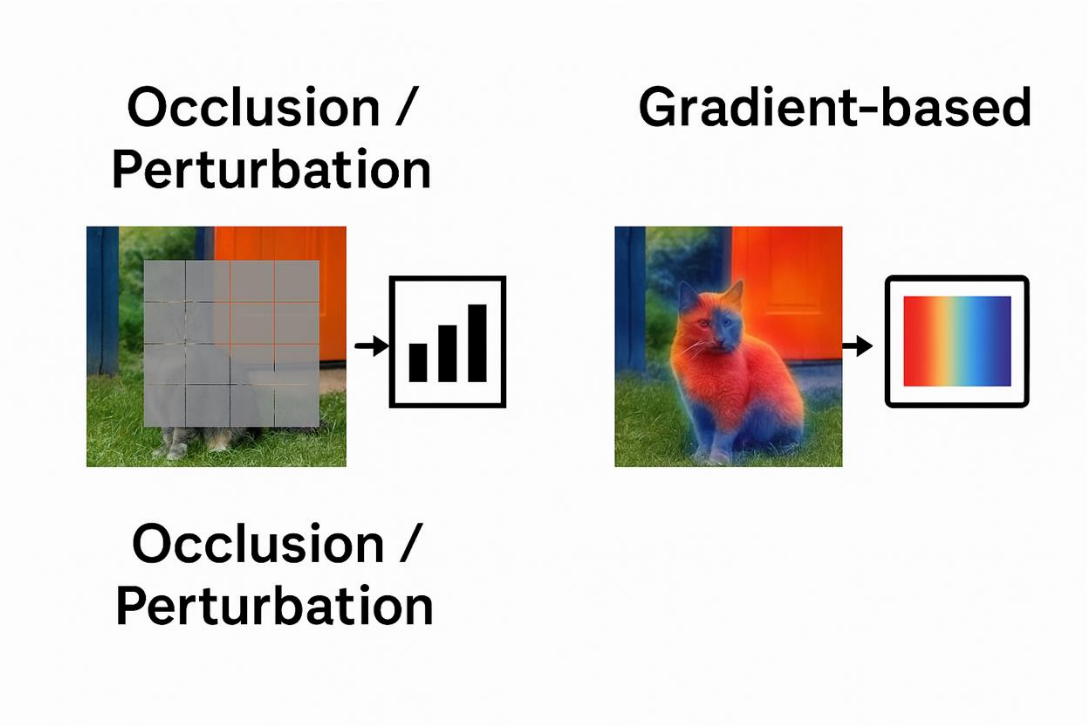
Feature Attribution

- 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 $S \in \Re^{C}$ [for regression $C = 1$]
    - For image I we have $S(I) = [S_1(I), \cdots, S_C(I)]$
  - Input to feature attribution $x \in \Re^{p}$ (pixel, tabular data, words, etc) with p features
  - Output: Relevance score for each feature $R^c = [R_1^c, \cdots, R_p^c]$, where c indicates the relevance for the c^{th} output $S_C(I)$

DIGITAL

Pixel Attribution

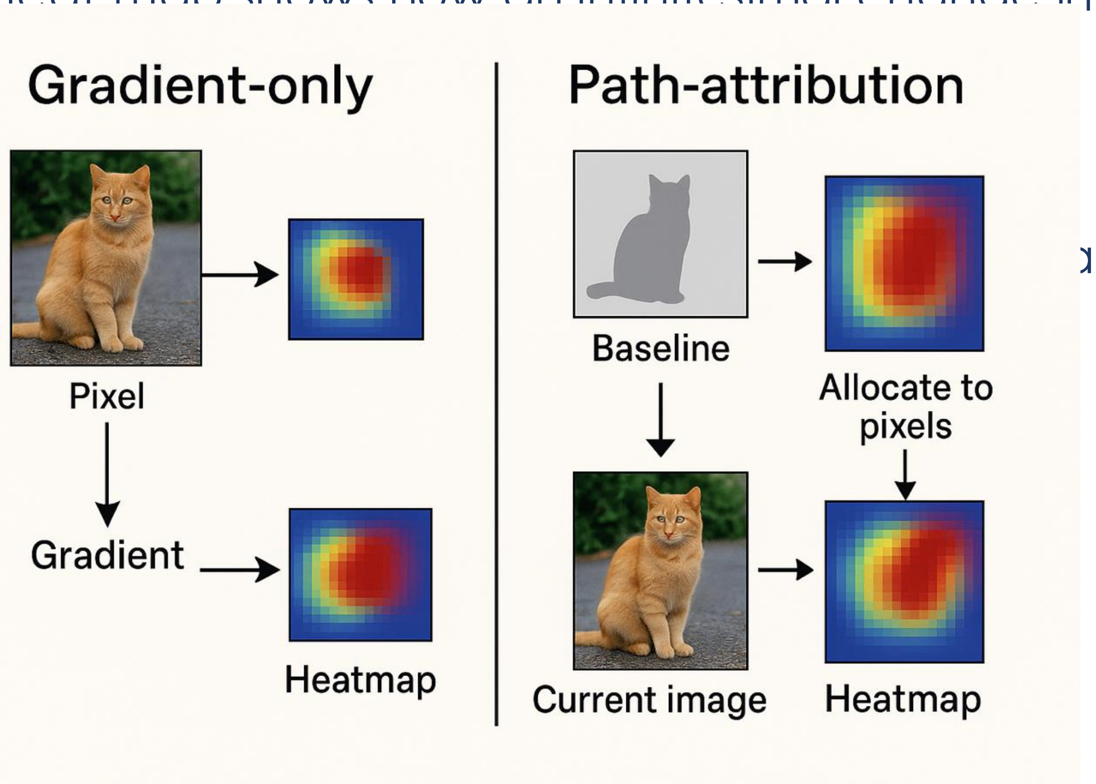
- Feature (pixel) attribution: Explains predictions by attributing relevance to each input feature (pixels, tabular data, words).
- Two major classes(pixel attribution):
  - **Occlusion / perturbation methods** (e.g., SHAP, LIME) explain a model by hiding or altering small regions of the image and observing 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.



DIGITAL

Pixel Attribution

- Gradient-only : Backpropogate once. The heat-map shows how an infinitesimal change in each pixel would raise (warm colours) or lo
  - Examples – Vanilla Gradients, Grad-CA
  - local sensitivity
- Path-attribution: Compare to a baseline. Su a grey "zero" image) from the current score c
  - Examples – Integrated Gradients, Deep If the method is complete, the pixel sco
  - contribution relative to a reference.
- Both produce a same-size heat-map you c



DIGITAL

Vanilla Gradient (Saliency Maps)

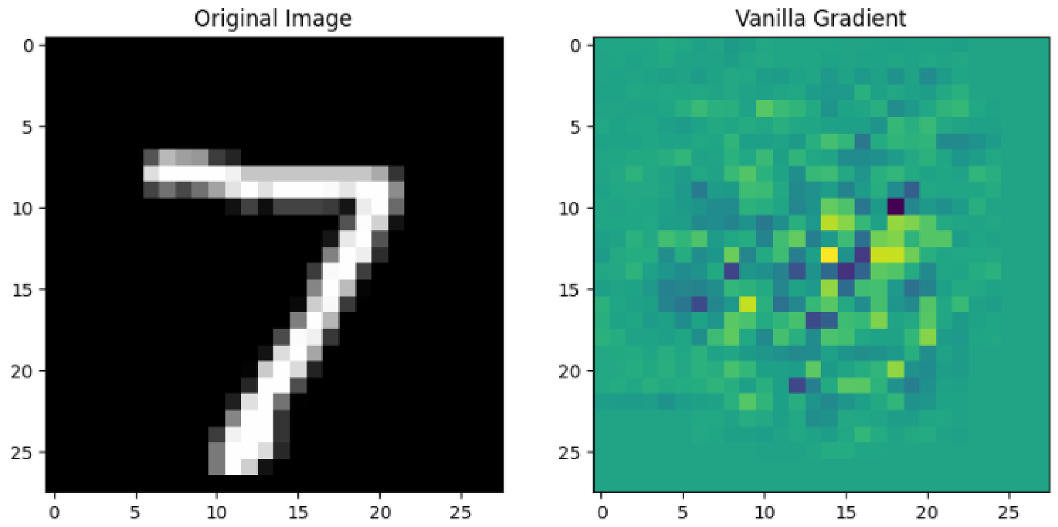
- Calculates the gradient of the loss function for the desired class relative to input pixels.
  - Produces a map showing pixel influence on class prediction, with values ranging from negative to positive.
- Three steps:
  - Forward Pass: Process the image through the neural network to activate outputs.
  - Gradient Computation: Calculate the gradient
  - $E_{grad}(I_0) = \frac{\delta S_c}{\delta I}\Big|_{I=I_0}$
  - 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.

DIGITAL

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

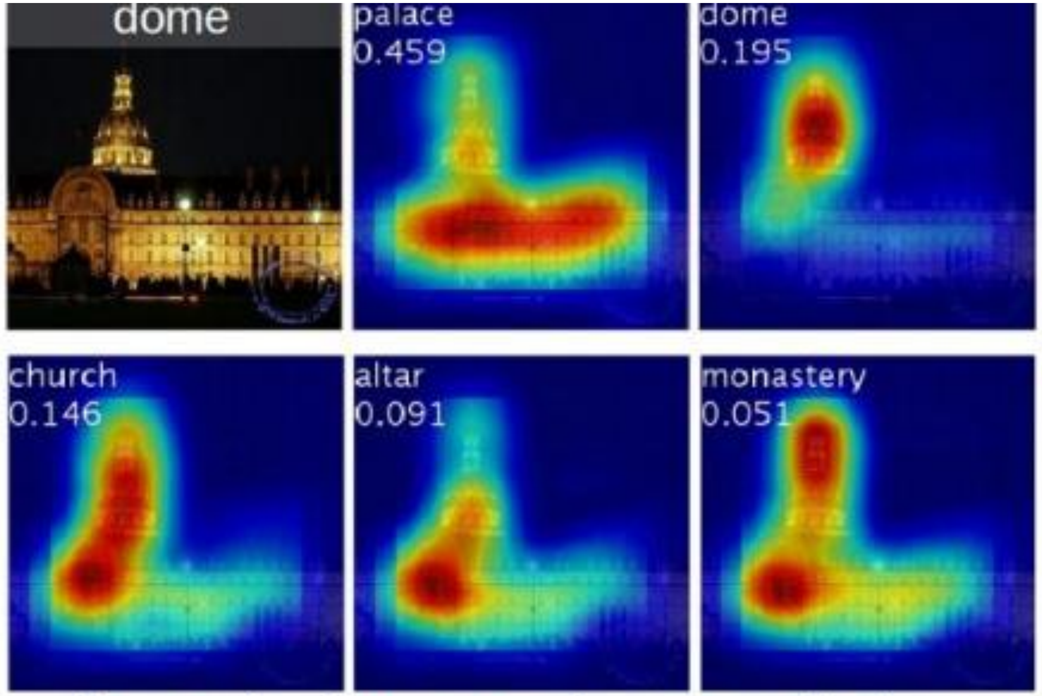
$$S_c(I) \approx \left.\frac{\delta S_c}{\delta I}\right|_{I_0} + b = w^T I + b$$



DIGITAL

Class Activation Map (CAM)

CAM shows which parts of an image the CNN focuses on to recognize a specific category.



Class activation maps of top 5 predictions

Class activation maps for one object class

[

CAM: How it Works?

- Feature maps from CNN
  - The last convolutional layer produces multiple feature maps ($f_k(x, y)$).
- Global Average Pooling (GAP)
  - Each feature map is averaged resulting in one value per feature map F_k
  - This reduces dimensionality while keeping key information.
- Class-specific weights
  - Each class ccc has weights (w_k^c) showing how important each feature map is.
  - Weighted sum gives the class score (S_c).
- Final prediction
  - Apply softmax to convert scores into probabilities (P_c).

DIGITAL

CAM: How it Works?

- Class-specific weights
  - Each class c has weights ($(w_1^c, w_2^c, \dots, w_n^c)$
  - These indicate how important each feature map (f_k) is for predicting that class.
- Compute CAM
  - Multiply each feature map by its weight and sum them:
  - $M_c(x, y) = \sum_k w_c^k f_k(x, y)$
  - This gives a heatmap showing which regions of the image contribute most to class c.
- Upsampling step
  - Since the CAM has smaller dimensions than the original image, it is resized/upsampled to align with the input image for visualization.

DIGITAL

CAM: Simple Example

Suppose the last convolutional layer produces two feature maps:

$$f_1 = \begin{bmatrix} 2 & 1 \\ 0 & 3 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \end{bmatrix}$$

- **Step 1: Global Average Pooling (GAP)** for feature map f_k:

$$F_k = \frac{1}{Z} \sum_{x,y} f_k(x, y)$$

  where Z is the number of spatial positions. For out example:

$$F_1 = \frac{2+1+0+3}{4} = 1.5, \quad F_2 = \frac{1+0+2+1}{4} = 1$$

- **Step 2: Class-specific weights (for "Dog")** Suppose the values of weights (after the model is retrained):

$$w_1^{dog} = 0.8, \quad w_2^{dog} = 0.2$$

- **Step 3: Class score for class c:**

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

  where w_k^c is the learned weight for feature map k and class c. For our example,

$$S_{dog} = w_1^{dog} \cdot F_1 + w_2^{dog} \cdot F_2 = 0.8 \cdot 1.5 + 0.2 \cdot 1 = 1.4$$

- **Step 4: Class Activation Map (CAM):**

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

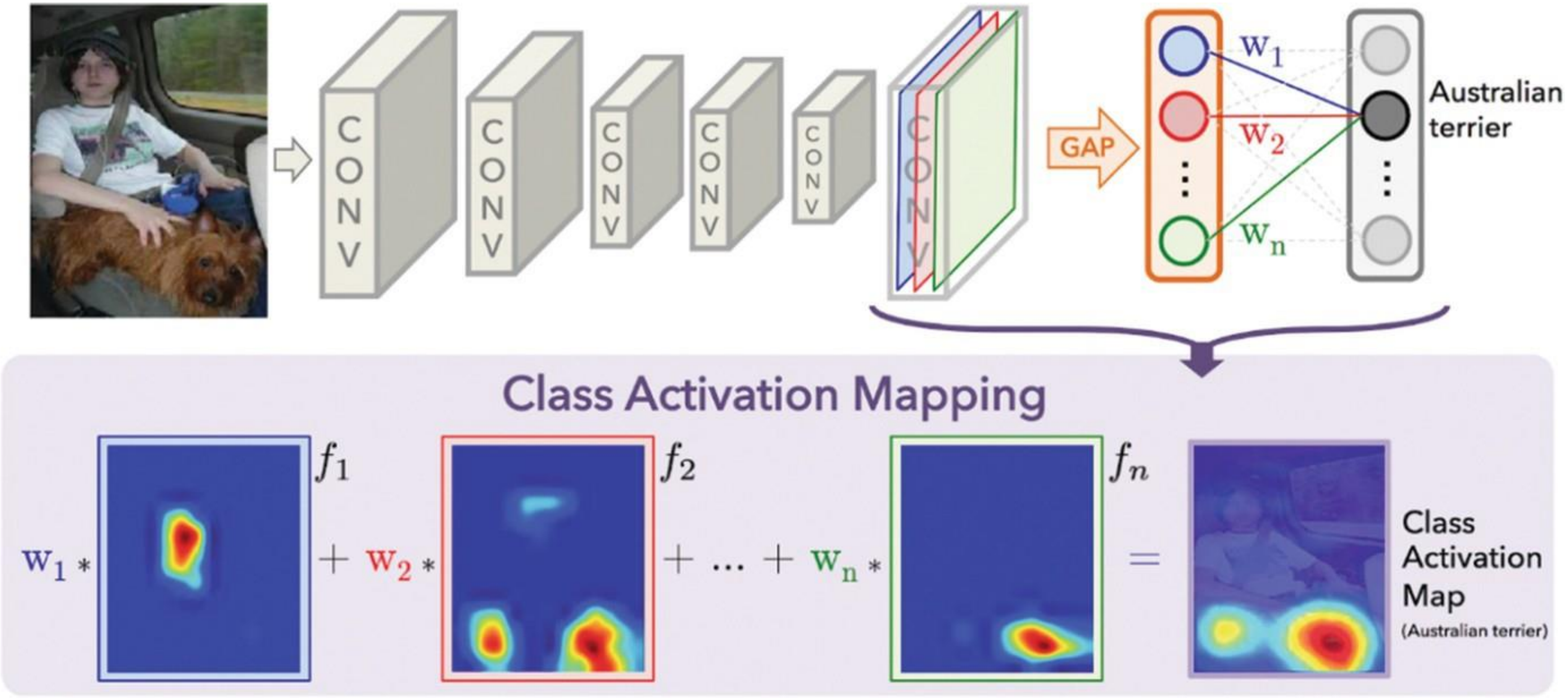
  For our exmple.

$$M_{dog}(x, y) = 0.8 \cdot f_1(x, y) + 0.2 \cdot f_2(x, y)$$

$$M_{dog} = \begin{bmatrix} 1.8 & 0.8 \\ 0.4 & 2.6 \end{bmatrix}$$

  The CAM can then be upsampled to the input image size to form a heatmap showing which pixels contribute most to the "Dog" prediction.

Class Activation Map (CAM)



DIGITAL

Gradient Class Activation Map (Grad-CAM)

- Uses gradients (not weights) from the last convolutional layer.
    - No retraining or architectural changes needed.
- Use gradients of the class score with respect to feature maps.
- Gradients tell us which feature maps are important for the class.

Grad-CAM: Working

- Compute gradients for the top predicted class w.r.t. feature maps i.e. $\frac{\partial S_c}{\partial f_k(x,y)}$
- Apply global average pooling to get weights.

- $\alpha_k^c = \frac{1}{Z}\sum_{x,y}\frac{\partial S_c}{\partial f_k(x,y)}$
- Take dot product of weights and feature maps.
- $\sum_k \alpha_k^c f_k(x,y)$
- Apply ReLU → keep only positive contributions.
- $M_c(x,y) = ReLU(\sum_k \alpha_k^c f_k(x,y) \;\;)$

DIGITAL

Grad-CAM

Algorithm 2: Gradient-weighted Class Activation Mapping

Input: Image I, trained CNN, target class c

Output: Heat-map L_{GradCAM}

; /* Forward pass */

$A^k \leftarrow$ feature-maps of the *last conv* layer;

$S_c \leftarrow$ predicted score for class c;

; /* Backward pass */

$\frac{\partial S_c}{\partial A^k} \leftarrow$ gradients w.r.t. each map;

; /* Channel importance */

$\alpha_k = \frac{1}{Z} \sum_{i,j} \frac{\partial S_c}{\partial A^k_{ij}}$ *[r]global average pooling

; /* Linear combination & ReLU */

$L_{\text{GradCAM}} = \text{ReLU}\left(\sum_k \alpha_k A^k\right)$;

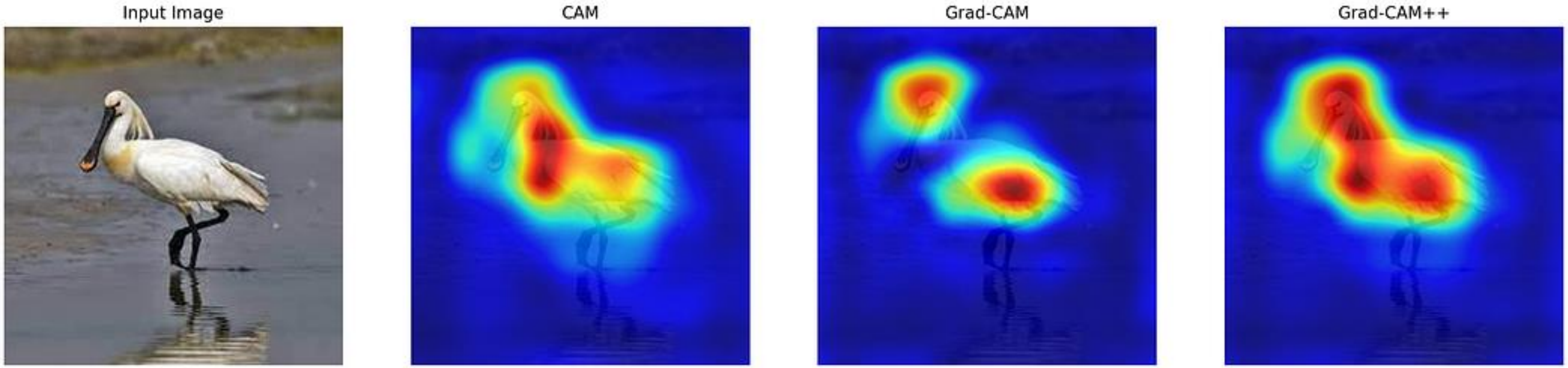
; /* Upsample */

Resize L_{GradCAM} to the resolution of I and overlay;

DIGITAL

CAM vs GRAD-CAM

https://medium.com/@tanishqsardana/visualizing-how-cnn-thinks-cam-grad-cam-grad-cam-1dabcf886cc5

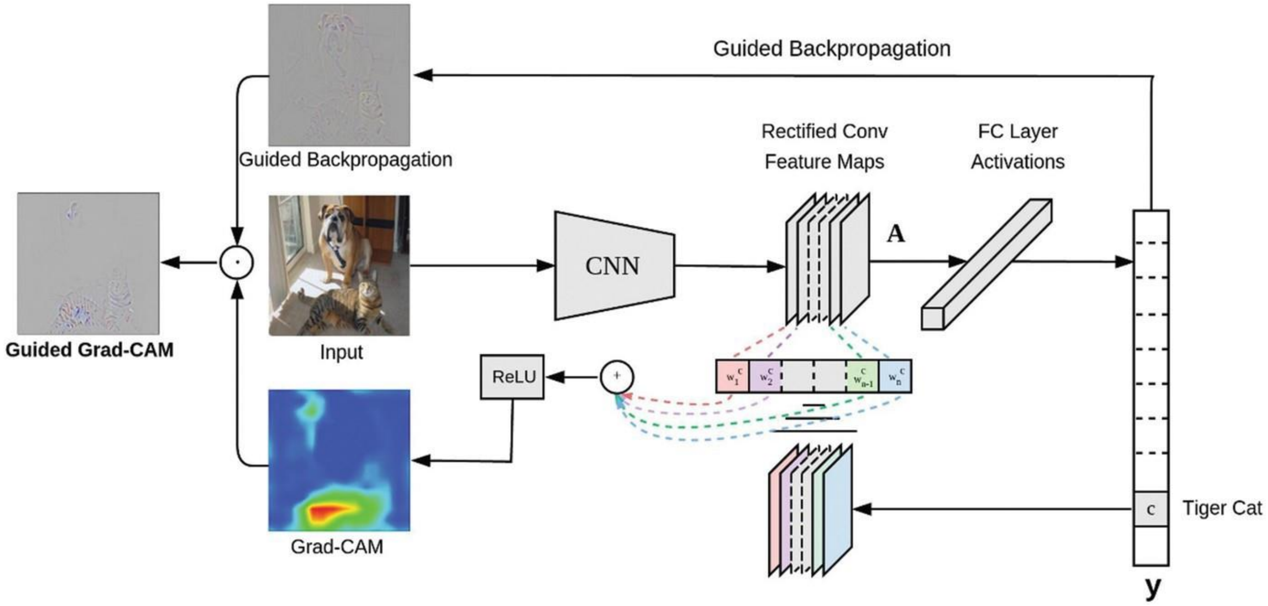


DIGITAL

Gradient Class Activation Map (Grad-CAM)

- An evolution of the CAM is the Grad-CAM.
- Grad-CAM does not retrain the network.

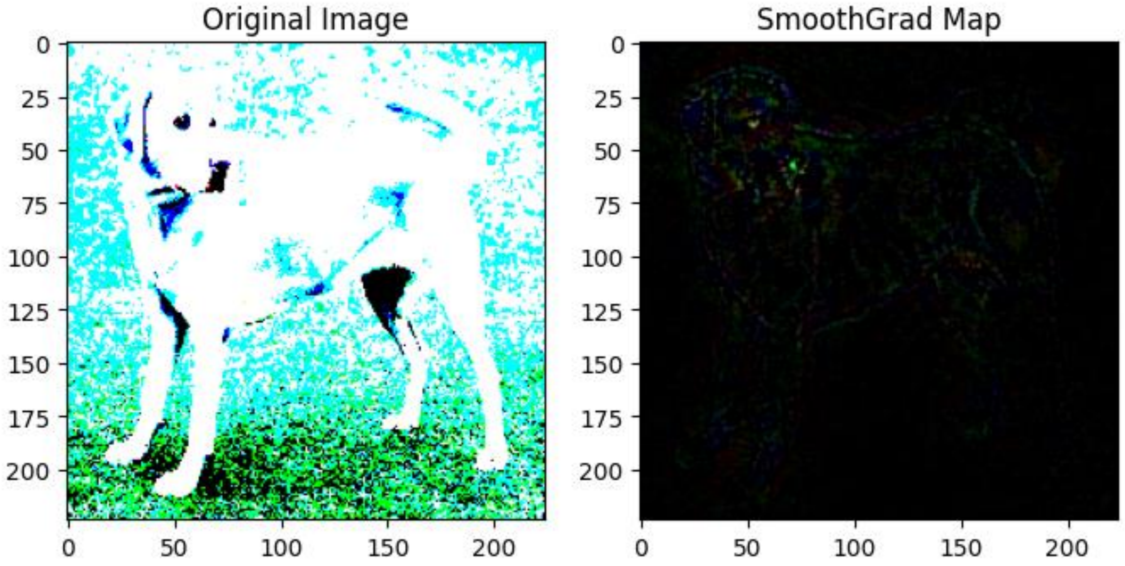
- Grad-CAM's explanations can suffer from gradient problems.



DIGITAL

Smooth Gradient

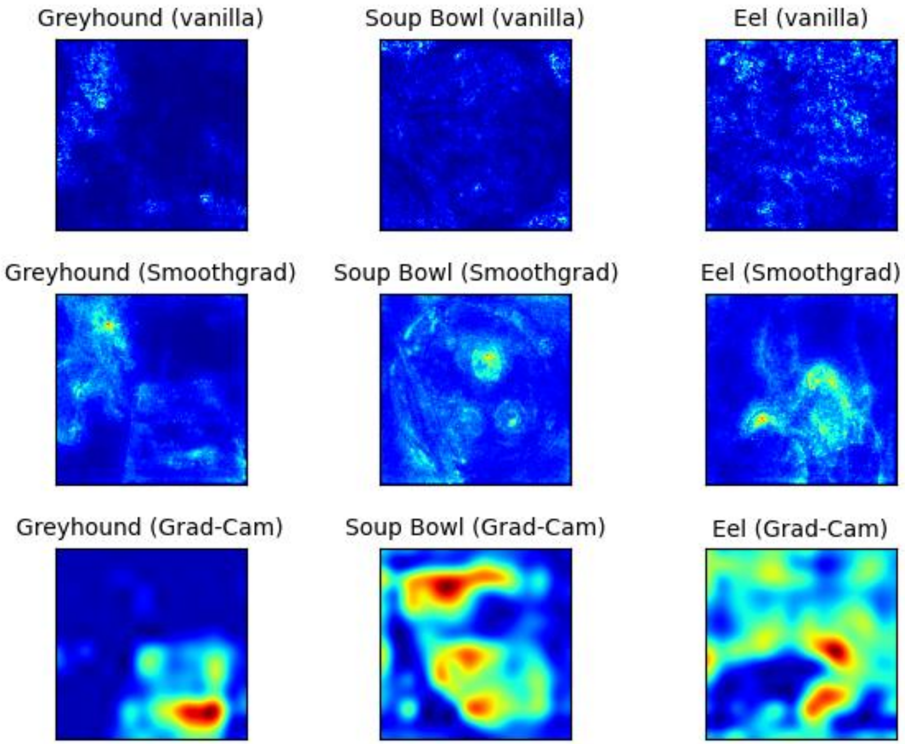
- A generic method
  - Noise Addition: Generate multiple versio
    noise.
  - Pixel Attribution: Create pixel attribution r
  - Averaging: Average these maps to prod
    visualization.
- Derivatives in neural networks can be noisy d
- Averaging over several noisy samples reduce
  insights.



DIGITAL

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%.



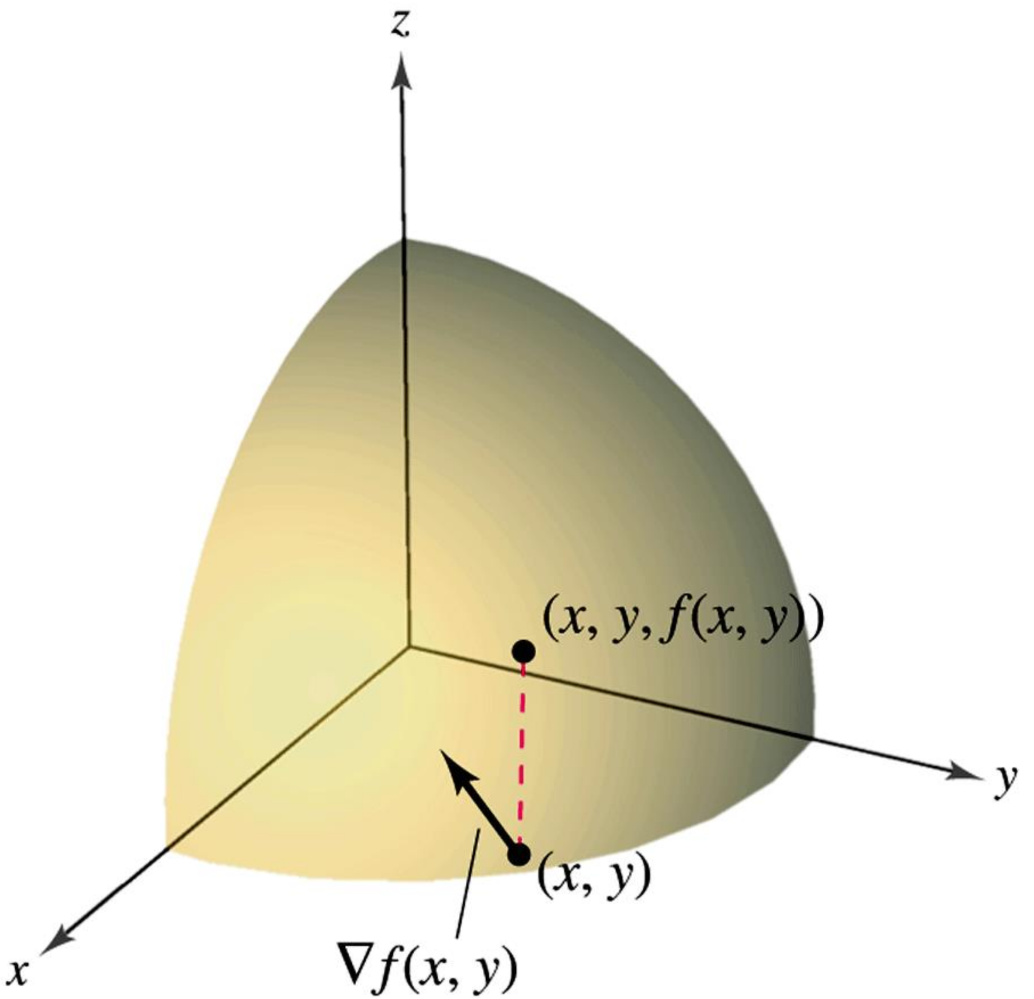
DIGITAL

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

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

DIGITAL

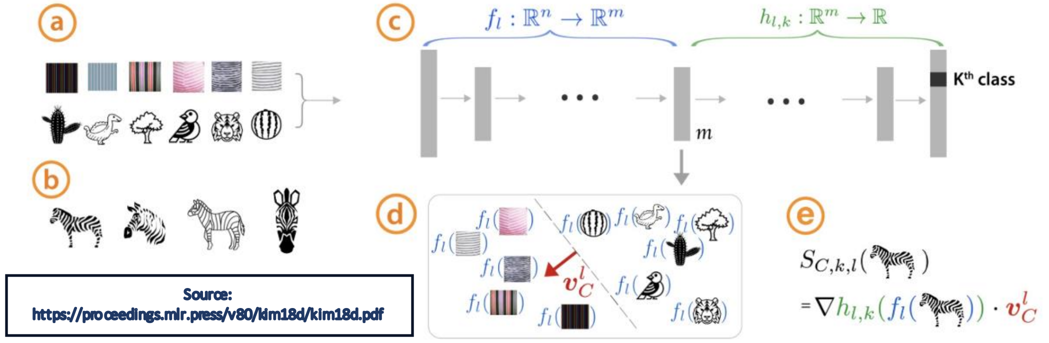
Concept Activation Vectors

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

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

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

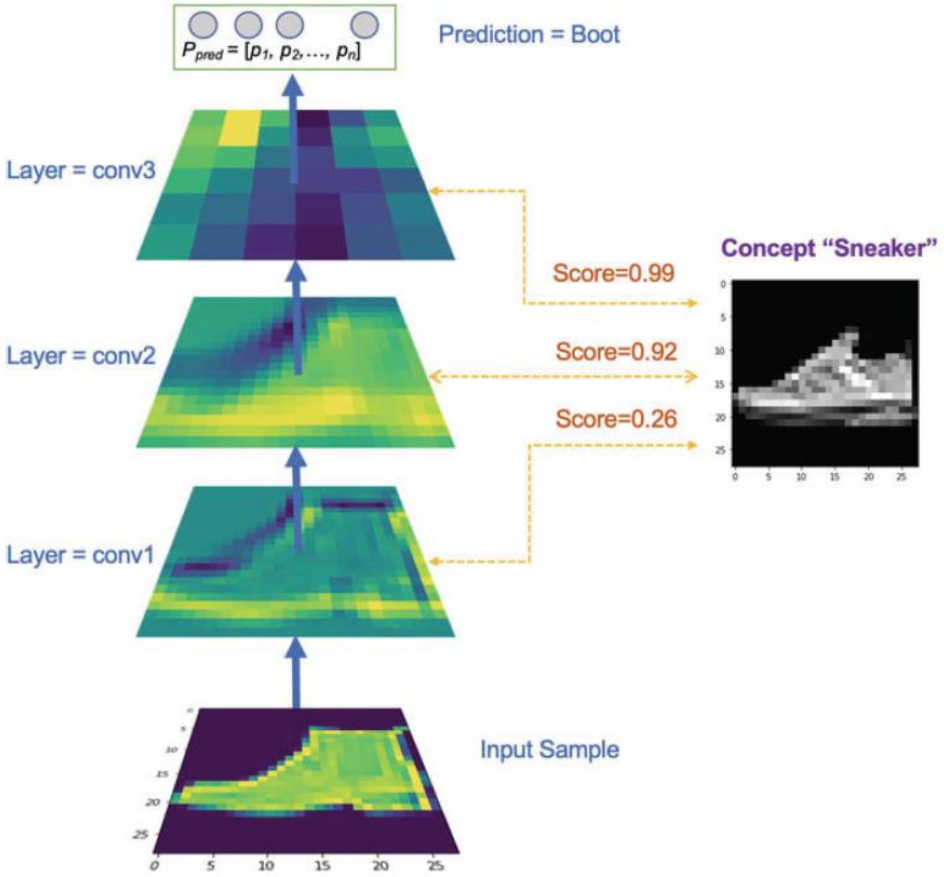
$$S_{C,k,l}(x) = \nabla h_{l,k}\big(f_l(x)\big) \,.\, v_C^l$$



Source: https://proceedings.mlr.press/v80/kim18d/kim18d.pdf

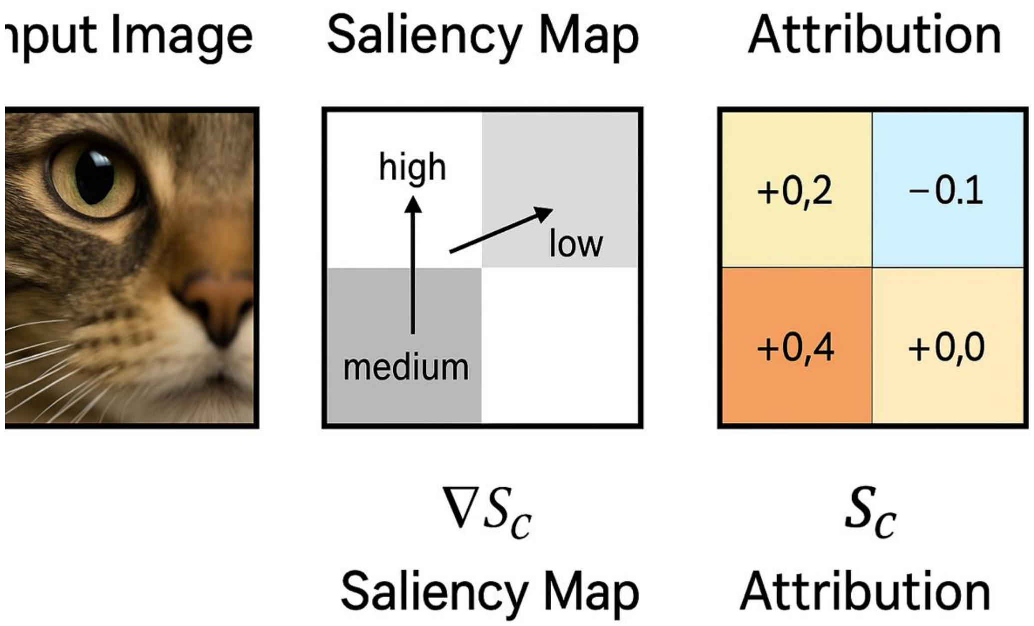
DIGITAL

Concept Activation Vectors



DIGITAL

From Saliency Maps to Formal Decomposition



- Recall: the saliency map computes the **gradient of the class score** with respect to each pixel:

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

This shows **how sensitive** the class score S_c is to small input changes.

- However, sensitivity ≠ contribution.

→ It tells *how changing a pixel would affect* S_c, *not how much that pixel contributed to* S_c itself.

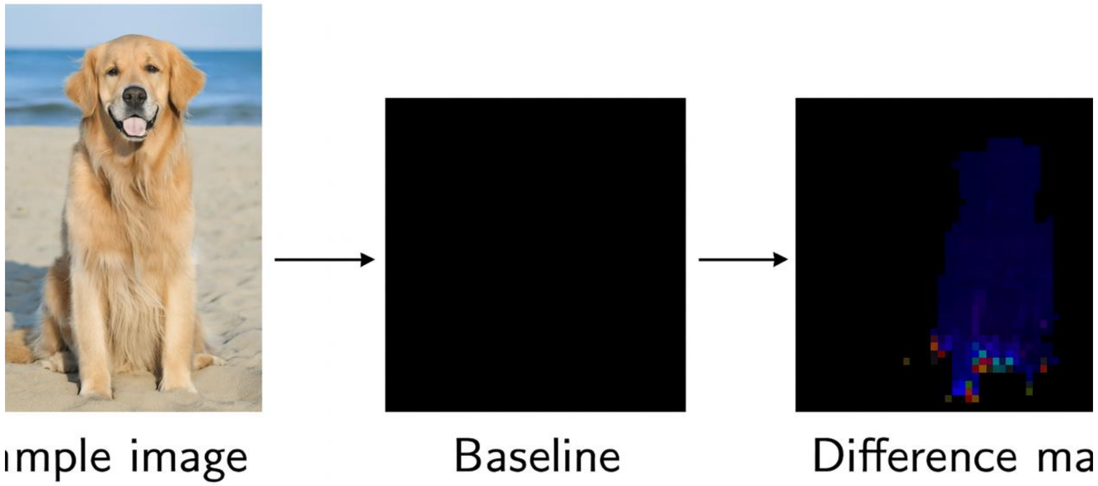
- To understand **actual contribution**, we approximate how S_c can be reconstructed from its inputs using **Taylor Decomposition**.

The Taylor Decomposition of the Class Score

- Expand $S_c(I)$ around a **reference image** I_0:
$$S_c(I) \approx S_c(I_0) + \sum_i (I_i - I_0)\left(\frac{\partial S_c(I_0)}{\partial I_i}\right)$$
- Each term
$$R_i = (I_i - I_0)\left(\frac{\partial S_c(I_0)}{\partial I_i}\right)$$

gives the **relevance (contribution)** of pixel i.

- I_0 represents an input with *no class-specific information* (e.g., black image, blurred image, dataset mean).
- The difference $I - I_0$ defines *how much signal* each pixel adds.
  - Good baseline → meaningful attributions;
  - Bad baseline → distorted relevance.



“The baseline defines what ‘zero relevance’ means.

DIGITAL

The Taylor Decomposition of the Class Score

Works only **locally** around I_0.

Deep networks are **highly nonlinear** — single expansion can't follow their internal structure.

Doesn't account for how activations interact **across layers**.

Explanations can become **unstable or noisy**.

DIGITAL

Deep Taylor Decomposition (DTD)

- **Deep Taylor Decomposition (DTD)** applies Taylor expansion *locally at each layer* of the network.
- Each neuron's relevance R_j^{l+1} is redistributed to its inputs R_i^l using local linear rules derived from Taylor analysis.
- Ensures **relevance conservation**:

$$\sum_i R_i^l = \sum_j R_j^{l+1} = S_c(I)$$

Layer-wise Relevance Propagation (LRP) implements these rules efficiently, producing stable and interpretable heatmaps.

DIGITAL

From Deep Taylor to LRP

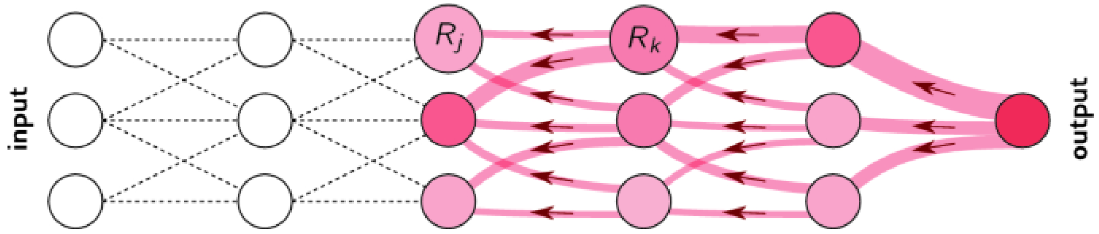
- LRP operationalizes Deep Taylor Decomposition.
- Instead of explicitly computing local Taylor expansions, it **defines redistribution rules** that approximate them efficiently.
- Each layer applies a rule ensuring **relevance conservation**:

$$\sum_i R_i^l = \sum_j R_j^{l+1} = S_c(I)$$

- These rules vary depending on layer type (dense, convolutional, etc.).

🧠 *"LRP replaces symbolic Taylor expansion with practical propagation rules."*

Layer wise relevance propagation



- Uses weights and neural activations
- Created by forward-pass (i.e. prediction, not training)
- Go back from prediction to input
- Visualize image pixels which caused high activation

- Drawback
  - only one sample explained

LRP Formulas

Rule	Formula
LRP-0	$R_i = \sum_j \frac{a_i w_{ij}}{\sum_{i'} a_{i'} w_{i'j}} R_j$
LRP-ε	$R_i = \sum_j \frac{a_i w_{ij}}{\sum_{i'} a_{i'} w_{i'j} + \varepsilon \cdot \mathrm{sign}(\sum_{i'} a_{i'} w_{i'j})} R_j$
LRP-γ	$R_i = \sum_j \frac{a_i (w_{ij} + \gamma w_{ij}^{+})}{\sum_{i'} a_{i'} (w_{i'j} + \gamma w_{i'j}^{+})} R_j$
LRP-$\alpha\beta$	$R_i = \sum_j \left[\alpha \frac{a_i w_{ij}^{+}}{\sum_{i'} a_{i'} w_{i'j}^{+}} - \beta \frac{a_i w_{ij}^{-}}{\sum_{i'} a_{i'} w_{i'j}^{-}} \right] R_j, \quad \alpha - \beta = 1$

DIGITAL

Why so many LRP formulas?

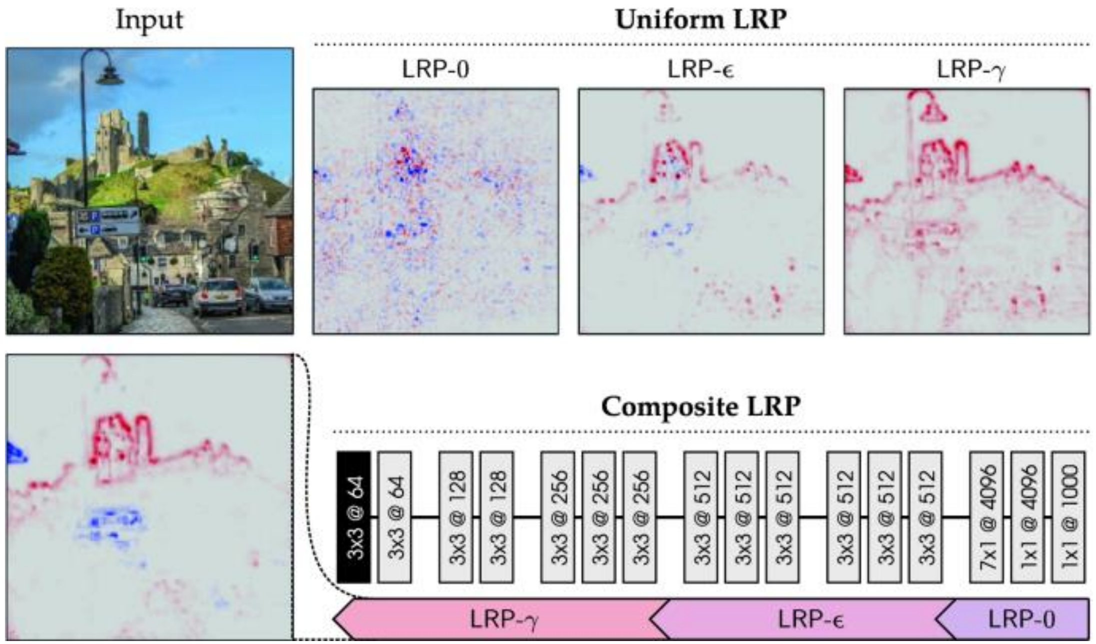
Rule	When to Use	Advantage	Limitation / Source
LRP-0	For simple linear or fully connected layers	Conceptually clear and easy to interpret	Can become unstable when denominators approach zero
LRP-ε	Most commonly used in deep networks	Stabilizes relevance propagation and prevents division by small values	Small ε may blur fine details in heatmaps — *Bach et al. (2015, PLOS ONE)*
LRP-γ	Convolutional layers (especially with ReLU)	Emphasizes positive (supporting) contributions, sharper explanations	Ignores inhibitory (negative) evidence — *Bach et al. (2015)*
LRP-αβ	General-purpose; useful when both positive and negative evidence matter	Flexible; allows balancing support and suppression by tuning α, β	Requires parameter tuning (commonly α=2, β=1) — *Montavon et al. (2017, Pattern Recognition)*

DIGITAL

Do we need to apply uniformly?

- Instead of using one uniform rule, one can apply different rule to different layers.

https://link.springer.com/chapter/10.1007/978-3-030-28954-6_10



DIGITAL

Comparing Explanation Methods

Category	**Method**	**Key Idea**	**Output Type**	**Limitation**
Feature Visualization	Activation Maximization	Generate input that maximizes neuron/class	Synthetic feature pattern	Not input-specific
Attribution	SHAP / Integrated Gradients	Distribute output over input features	Scalar attribution scores	May violate conservation
Saliency Maps	Vanilla Gradient	Sensitivity of score to input	Gradient map	Local, noisy
Taylor Decomposition	Linear expansion around baseline	Additive local contributions	Per-feature relevance	Not layer-aware
LRP	Layer-wise relevance redistribution	Exact decomposition of class score	Stable, additive heatmap	Depends on rule choice

DIGITAL

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.

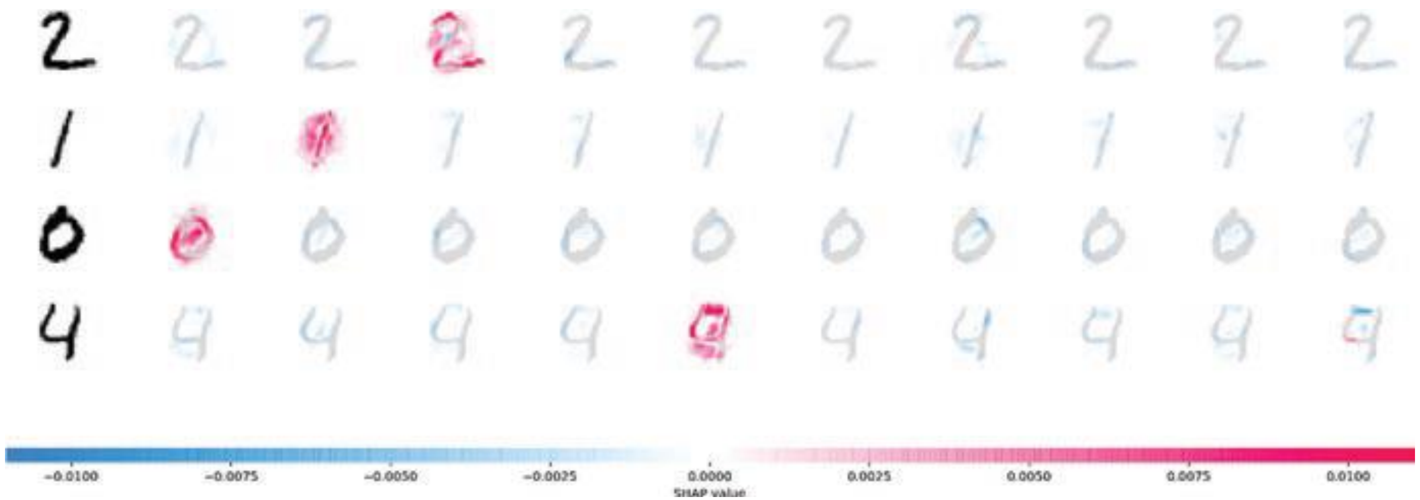
DIGITAL

DeepShap / DeepLift

- decomposes the prediction of a single-pixel neural network.
- done by carrying out the backpropagation of the contribution of all neurons in the network for each input feature.
- DeepLift compares the activation of each neuron with its reference activation and evaluates the importance of each contribution, starting from this difference
- This is similar to Shapley Values where the input features are the players in the coalitions, the activation of each neuron is their payment and we are trying find how to split the contribution to the output of the model between the input features

DIGITAL

DeepShap / DeepLift



DIGITAL

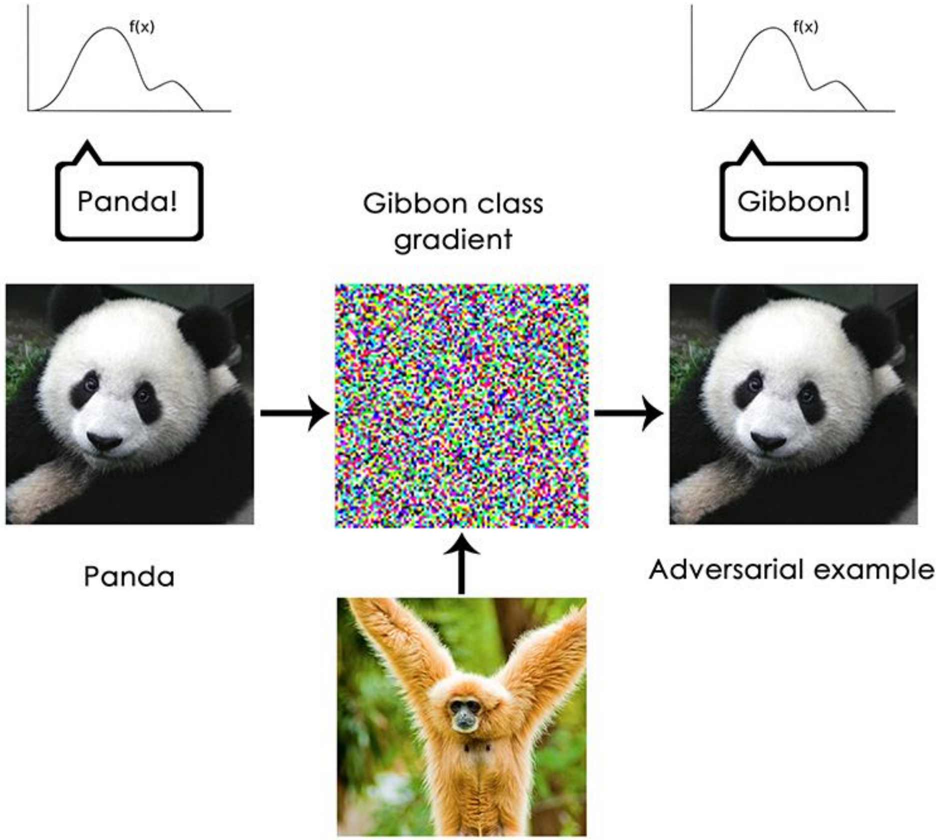
DeepShap / DeepLift

- Pros:
  - Does not rely on gradient, so does not suffer from discontinuous gradients nor zero gradient nodes, nor numerical instability.
  - Can reveal dependencies hidden in other methods.
- Cons:
  - DeepLift has it’s own activation function, hence requires wither retraining or a surrogate model.
  - Has to be done for each input feature, for bigger images, this is an issue.

DIGITAL

Gradient-based optimization attacks

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



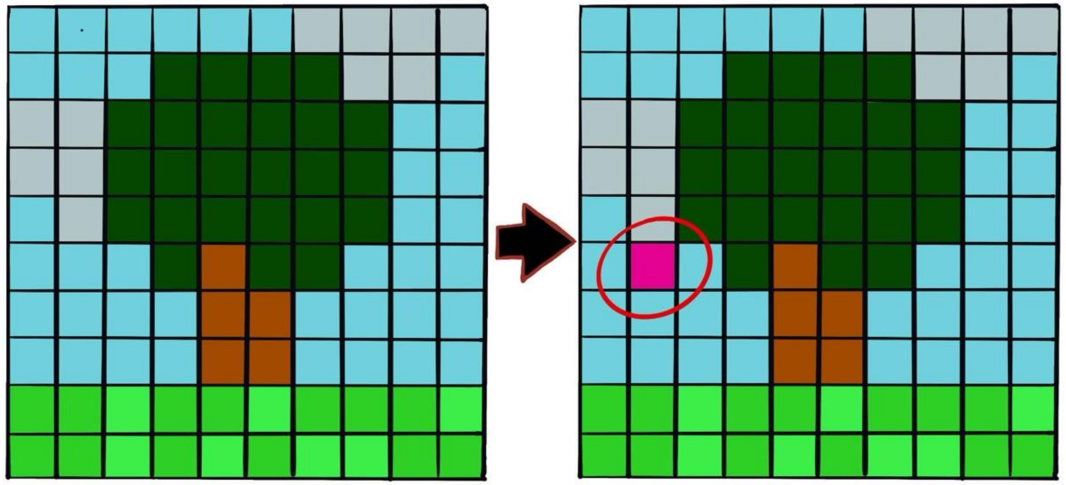
DIGITAL

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.



DIGITAL

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

DIGITAL

The present/future…

Focus Area / Method	Key Idea / Improvement	Representative Paper
Transformer-native LRP	Extends LRP and Deep Taylor decomposition to attention layers and transformer blocks	Chefer et al., *CVPR 2021*, "Transformer Interpretability Beyond Attention Visualization"
Generic Attention Explainability	Unified propagation of relevance in multi-modal and encoder–decoder transformers	Chefer et al., *ICCV 2021*, "Generic Attention-Model Explainability"
Concept-based Explanations	Learns and attributes human-interpretable concepts (ConceptSHAP, TCAV+)	Ghorbani et al., *NeurIPS 2023*, "ConceptSHAP: Concept-Based Explanations with Shapley Values"
Diffusion-based Counterfactuals	Uses diffusion models to generate realistic, semantically coherent counterfactuals	Anonymous, *CVIU 2024*, "Diffusion Models for Counterfactual Explanations (DiME)"
Causally Guided Counterfactuals	Generates counterfactuals consistent with causal structure and avoids spurious correlations	Qiao et al., *arXiv 2025*, "Causally-Guided Adversarial Steering"
Foundation Model Explainability	Explains large multi-modal and vision–language models (CLIP, ViT) using scalable relevance propagation	Zhang et al., *IEEE T-AI 2024*, "Explainability for Foundation Models: A Survey"
Refined LRP for Transformers	Adapts LRP rules (ε, γ, αβ) for attention, residual, and normalization layers	Voita et al., *TMLR 2023*, "LRP in Transformers: Accounting for Residuals and Attention"

DIGITAL

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