Unit-3: Explainability for Image Data

Why is explainability needed for CNNs?

CNNs work on the concept of feature extractions -> **What is the connectivity to explanation?**

Problem with learning spurious correlations

Regulatory approvals

Ensuring fairness and bias while learning

Human-AI Collaborations opportunities are plenty

Unit-3: Integrated Gradients (IG)

IG is a local attribution method, meaning it explains a model’s prediction for a single example image.

It generates a clear saliency map that highlights the image pixels or regions that most strongly influence the model’s prediction.

It is a Post-hoc explainability technique

IG evaluates the model's output for an actual input against a baseline input (usually a black or zero image) to measure how each pixel's value affects the change in output from the baseline to the actual prediction.

Decisions are based on features

The gradient of a function tells us how the function values change when the inputs are changed slightly

The gradient is a vector of derivatives; the gradient tells us for each input feature if the model function's prediction will increase or decrease when you take a tiny step in some direction of the feature space.

Unit-3: Integrated Gradients (IG)

Relying on gradient information alone can be problematic ????

The gradient tells you how the function changes at one specific point — it gives local information about the slope.

It does not tell you about the overall structure of the function – there may be many minima points

if the function is non-convex, following the gradient may lead to a local minimum rather than the global minimum.

Gradient Saturation??

Gradient Saturation?? Gradient saturation occurs when a model's prediction no longer responds to changes in a pixel's value, leading to a near-zero gradient, even though that pixel remains crucial to the prediction.

Unit-3: Integrated Gradients (IG)

The Integrated Gradients technique examines the model gradients along a path in feature space, summing up the gradient contributions along the path

Baseline is the starting point of the integration path — it represents an input with no information relevant to the model’s prediction.

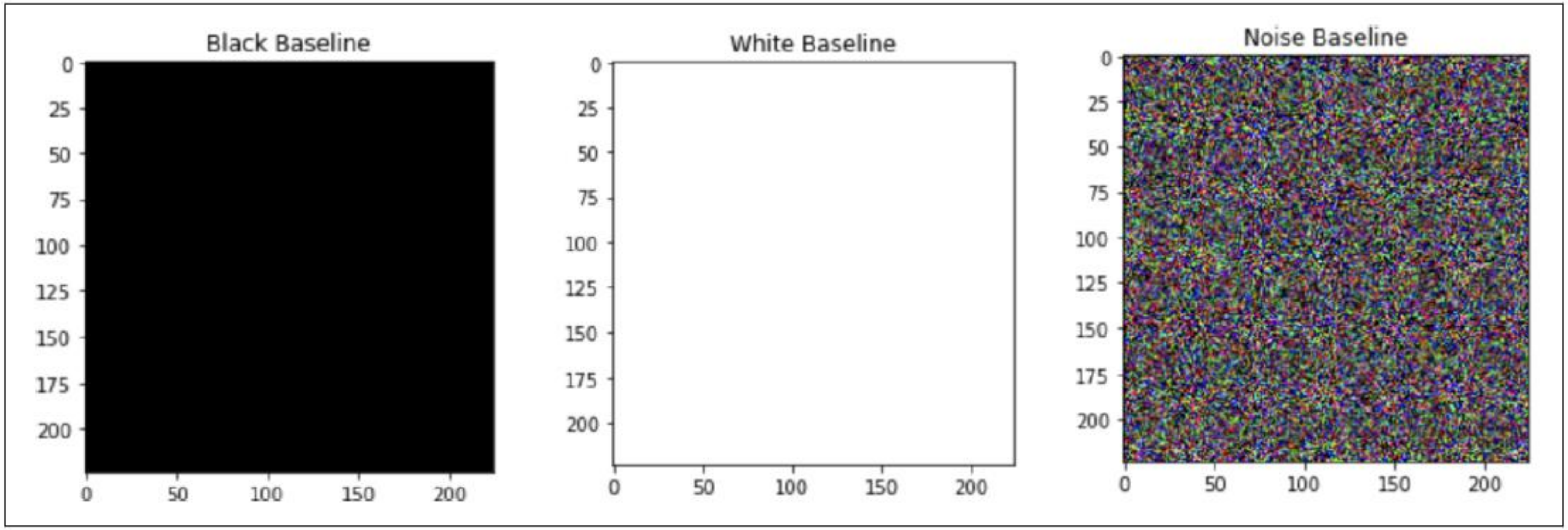
The ideal baseline should lack relevant information for the model’s prediction, allowing us to introduce features as we transition from the baseline to the image.

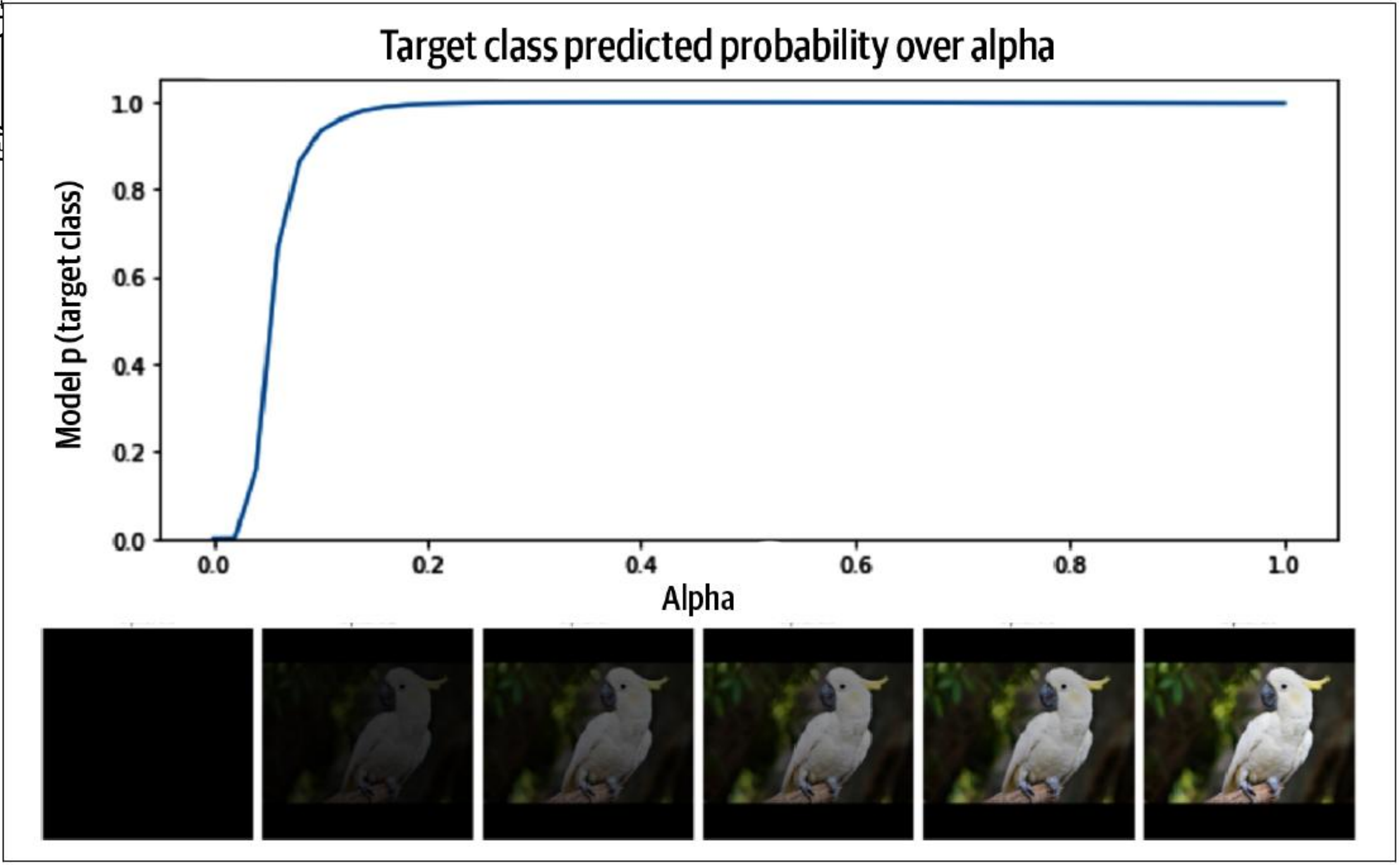
Choosing a Baseline

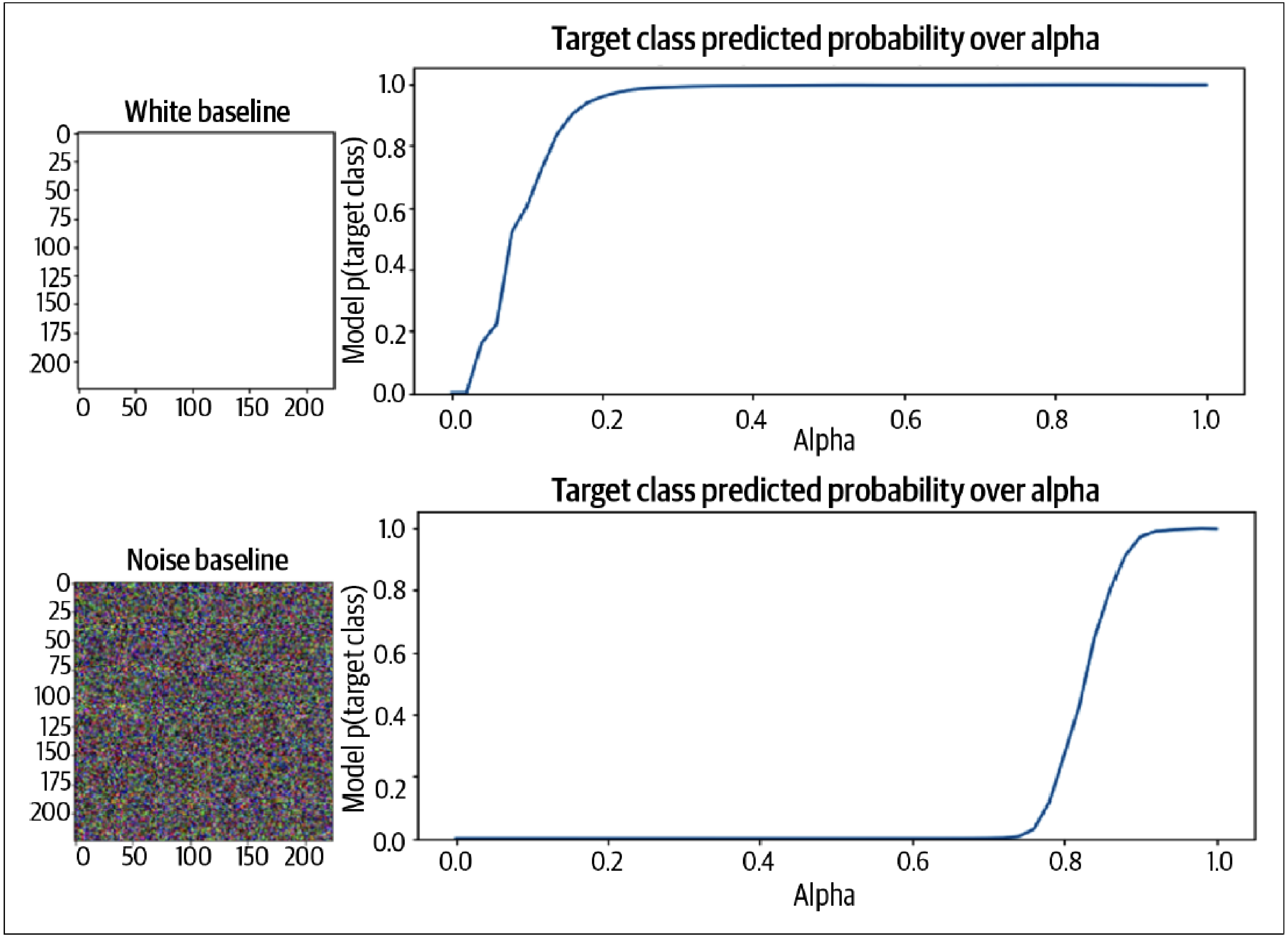
A good baseline contains neutral or uninformative pixel features.

Working with image models, the most commonly used baseline images are black image, white image, or noise

Unit-3: Integrated Gradients (IG)




Target class predicted probability over alpha
Model p (target class)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Alpha


White baseline
Target class predicted probability over alpha
Model p(target class)
Alpha
Noise baseline
Target class predicted probability over alpha
Model p(target class)
Alpha

Unit-3: Blurred Integrated Gradients (IG)

Blur-IG eliminates the need for a baseline parameter by using the blurred input image as the baseline for Integrated Gradients implementation.

- The most blurred image acts like the baseline (it has almost no details).
- The sharp, original image is the actual input.
- Along the path, we gradually reduce the blur — effectively revealing features step by step.

We can identify which image regions are most critical for its decision.

Intermediate images are always realistic-what about IG?

Unit-3: Guided Integrated Gradients (IG)

Guided IG uses an adapted path instead of a straight line from the baseline to the input image, resulting in saliency maps that better align with the model's prediction.

Guided IG moves in the direction where features (i.e., pixels) have the smallest absolute value of partial derivatives.

Guided IG adjusts pixel values non-uniformly, taking smaller steps in directions where the gradient ($\partial F/\partial x_i$) is low, meaning the model's output remains relatively stable.

Guided IG advantage over Blurred IG

Blur IG requires a specific blur kernel (e.g., Gaussian) and a set of blur levels.

Guided IG removes this manual choice — it is parameter-free and purely data-driven.

XRAI (eXplanation via Regional Attributes Integration)

Proposed by Google Research -> It is a region-based attribution built on IG

One common approach of determining salient inputs is to rely on the changes in the model output, such as gradients of the output with respect to the input features.

For example, Integrated Gradients (IG) determines the salient inputs by gradually varying the network input from a baseline to the original input and aggregating the gradients.

Sanity checks have been developed to determine whether a saliency method’s results meaningfully correspond to a model’s learned parameters.

Sensitivity-n empirically measures the quality of a saliency method’s output by comparing the change in the output prediction to the sum of attributions.

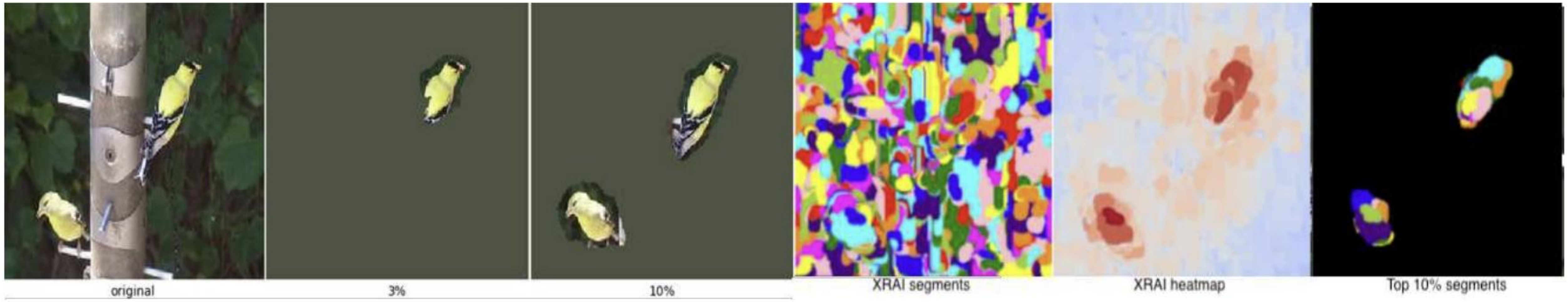


Figure 2. XRAI is a saliency method that incrementally grows attribution *regions*. In this figure, the most important regions for classifying the image as "goldfinch" are revealed for different area thresholds: the technique first identifies one bird (3%), then two birds (10%). The pool of possible segments to choose from are represented by the colored regions (XRAI segments). A saliency heatmap indicates which of these regions provide the most predictive power (XRAI heatmap). Finally, the most salient segments at area threshold of 10% (according to XRAI's ranking) are shown. Notice that XRAI constructs the bird from numerous segments and does not solely rely on segment quality.

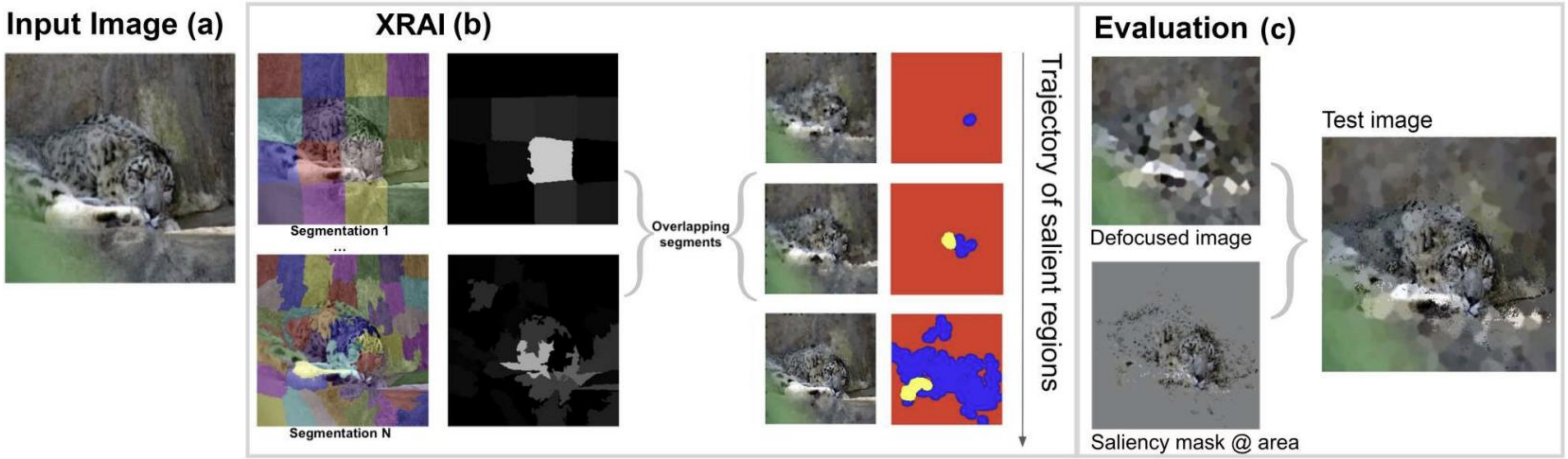


Figure 3. **(a)** Input leopard image. **(b)** XRAI's segmentation process. First, the image is over-segmented to many overlapping regions of various shapes, then the segments are gradually added with respect to their integrated gradients density. The ranking of region importance can be recovered from the trajectory. In this case, XRAI reconstructed the leopard's face, yielding a correct classification, then added the body and rest of the image. **(c)** Diagram of the evaluation method for a single image and a given area threshold. The unfocused image and salient region mask get combined to produce the saliency-focused image. This image is fed back to the classifier to measure performance.

XRAI (eXplanation via Regional Attributes Integration)

Proposed by Google Research -> It is a region-based attribution built on IG

One common approach of determining salient inputs is to rely on the changes in the model output, such as gradients of the output with respect to the input features.

For example, Integrated Gradients (IG) determines the salient inputs by gradually varying the network input from a baseline to the original input and aggregating the gradients.

Sanity checks have been developed to determine whether a saliency method's results meaningfully correspond to a model's learned parameters.

Sensitivity-n empirically measures the quality of a saliency method's output by comparing the change in the output prediction to the sum of attributions.

XRAI (eXplanation via Regional Attributes Integration)

Proposed by Google Research

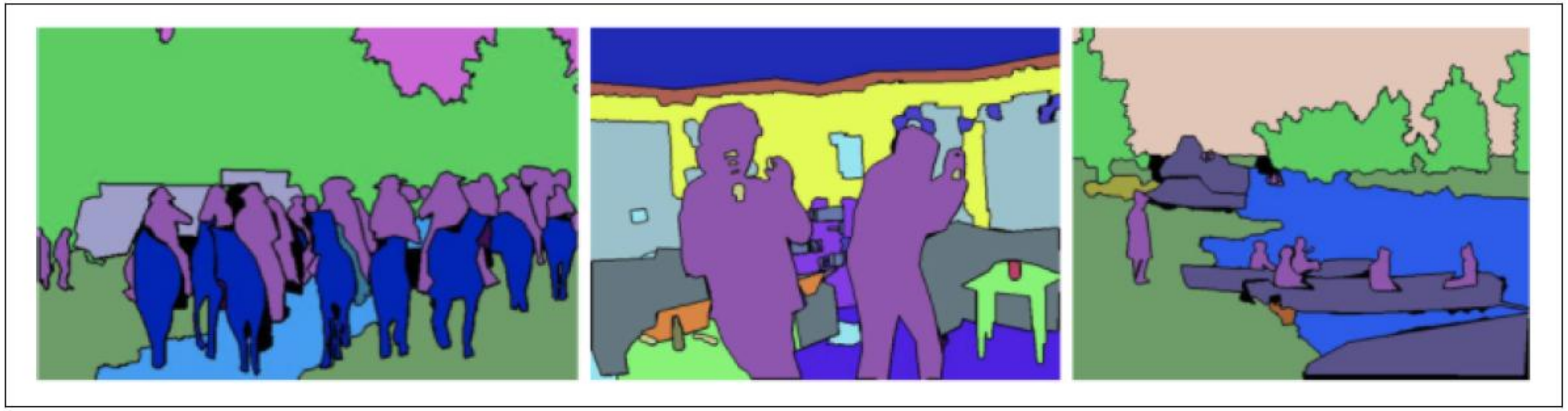
It is a region-based attribution built on IG

Find the RoI using segmentation, based on a similarity metric, and find the attribution scores for regions

Which regions of an input image are most responsible for prediction

Super Pixels

The saliency maps obtained by XRAI highlight pixel regions of interest in the image.



XRAI (eXplanation via Regional Attributes Integration)

Felzenszwalb’s algorithm

It segments an image into meaningful regions (superpixels) by grouping together pixels that are:

- similar in color or intensity
- spatially connected
- having smooth boundaries

The idea proposed by Felzenszwalb and Huttenlocher is to select edges from a graph, where each pixel corresponds to a graph node, and neighboring pixels are connected by undirected edges, with edge weights measuring the dissimilarity between pixels.

1. Represent the image as a graph
2. Sort the edges by weight
3. Merge regions when similarity is high
4. Use a threshold function to prevent over-merging

XRAI (eXplanation via Regional Attributes Integration)

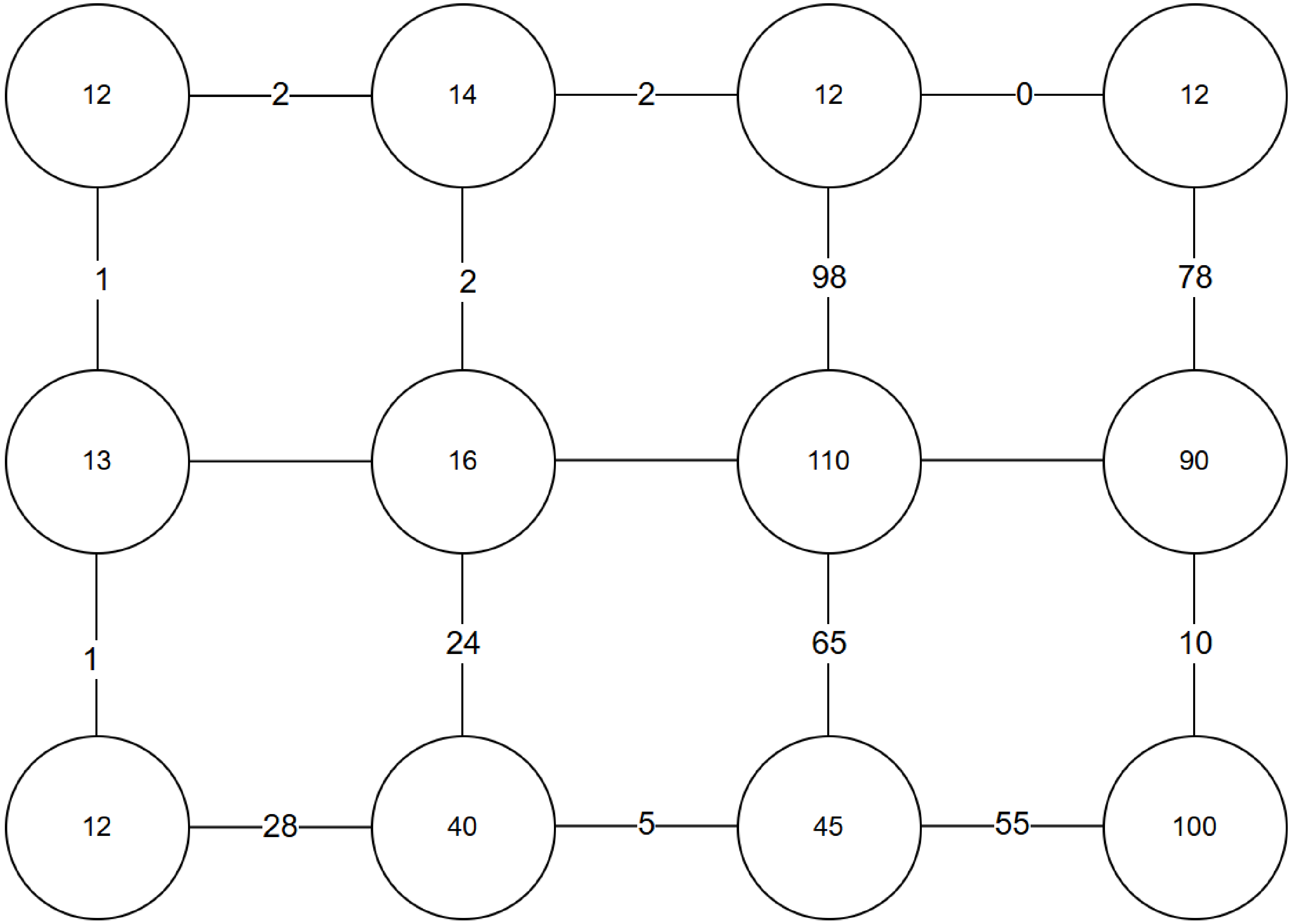
Problem Formulation

Let $G = (V, E)$ be an undirected graph such that,

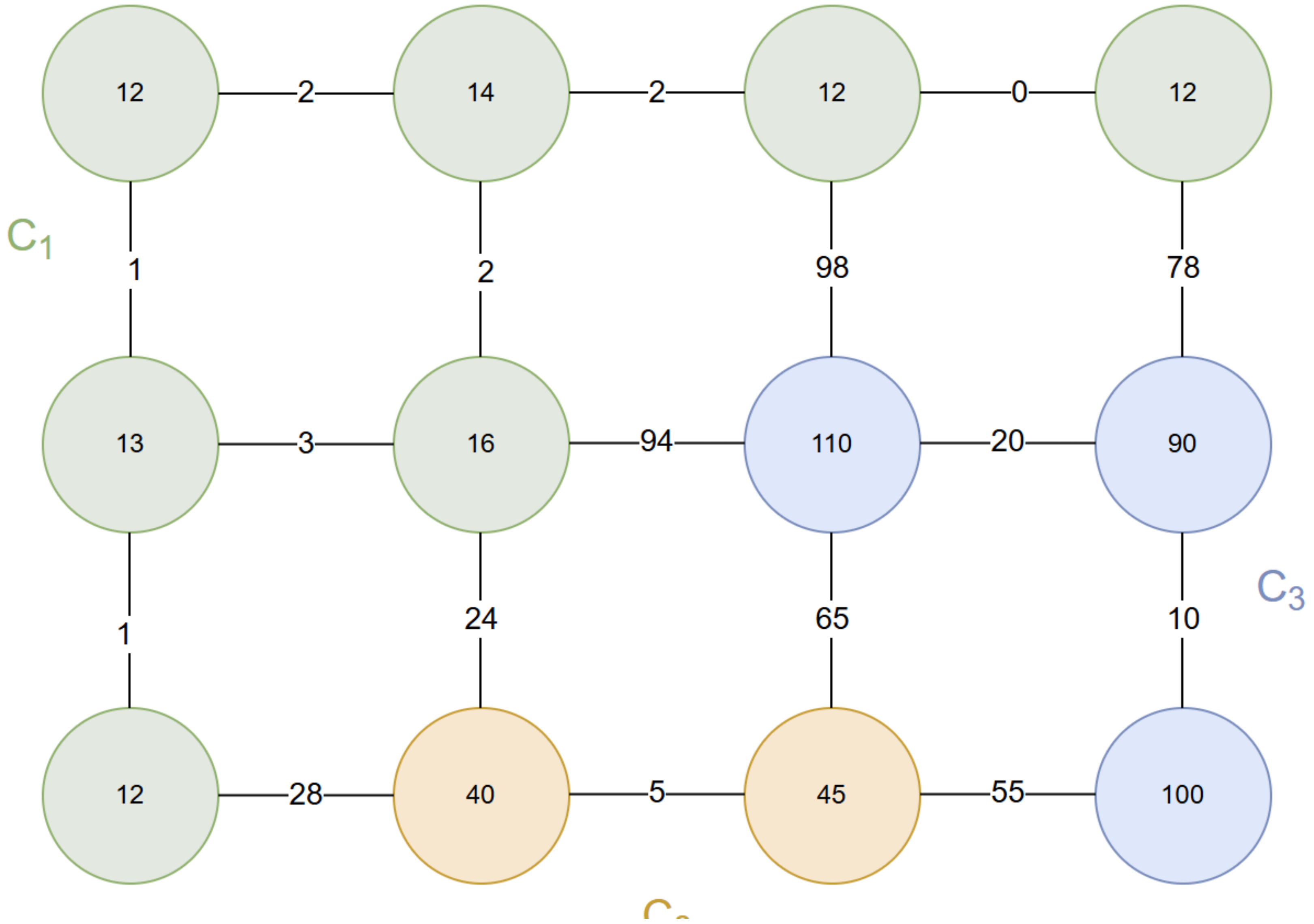
- $v_i \epsilon V$: set of vertices or pixels in the image to be segmented.
- $e = (v_i, v_j) \epsilon E$: set of edges corresponding to pairs of neighbouring vertices or pixels.
- Each edge $e = (v_i, v_j) \epsilon E$ has a weight $w(v_i, v_j)$ denoting the dissimilarity between v_i and v_j.

S is a segmentation of a graph G such that $G' = (V, E')$ where $E' \subset E$.
S divides G into G' such that it contains distinct components (or regions) C

XRAI (eXplanation via Regional Attributes Integration)



XRAI (eXplanation via Regional Attributes Integration)



Grad-CAM (Gradient-weighted Class Activation Mapping)

Generalizes the CAM

CAM can highlight the regions in an image that are important for a CNN's prediction for a specific class -> CAM tells **where the model is looking** in the image to make its prediction, by combining feature maps with their importance weights.

CAM works only for CNN architectures that have a global average pooling (GAP) layer before the final fully connected layer.

Grad-CAM relaxes the architectural restriction: it can work with any CNN-based model, even without global average pooling.

In the context of Grad-CAM, the "class score" refers to the raw output of the neural network for a specific class before applying softmax.

It produces a localization heatmap highlighting the regions in an image most influential for predicting a given class label.

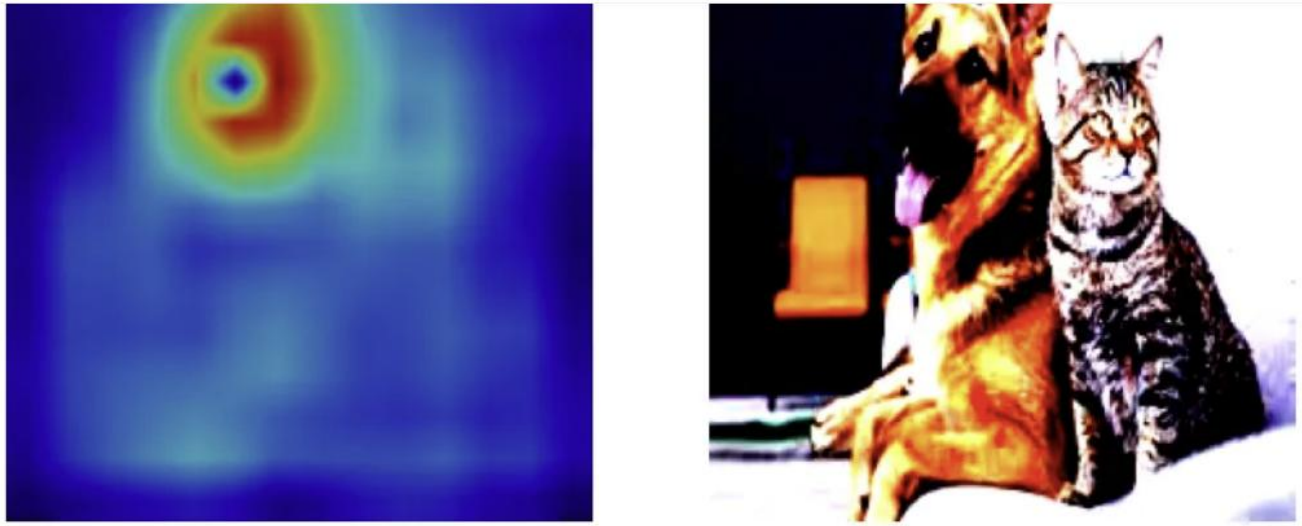


fig 1: activation map for dog

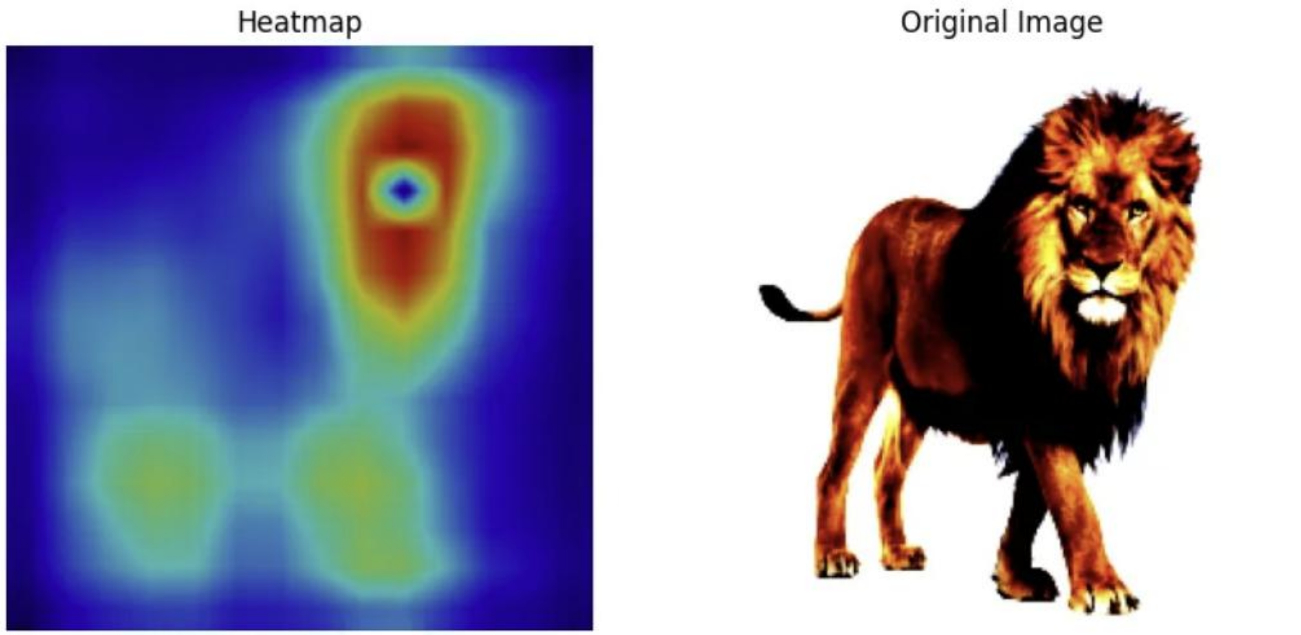


fig 2: activation map for lion

How Grad-CAM Works?

CAM is a localization map of the image that is computed as a weighted activation map

Localization map: A heatmap that shows **which regions of the image** contributed most to the decision of the model (e.g., a dog's face for "dog" class).

Activation map: In a CNN, each convolutional filter produces a **feature map** (activation map) that detects certain patterns like edges, textures, shapes.

Weighted**:** Not all features are equally important for predicting a particular class. CAM assigns **weights** to each feature map based on how strongly it influences the class prediction.

CAM combines multiple feature maps into **one heatmap**, where each feature map is multiplied by a class-specific weight:

$$\text{CAM}(x, y) = \sum_k w_k^c \cdot f_k(x, y)$$

Where:

•$f_k(x, y)$ =activation value at position (x, y)in feature map k

•w_k^c =how important feature map $k$is for class c

LIME (Local Interpretable Model-agnostic Explanations)

It is a post hoc, perturbation-based explainability technique.

Can be used for regression or classification models

Works by changing regions of an image, turning them on or off and rerunning inferences to see which regions are most influential to the model’s prediction.

Uses an inherently interpretable model to measure the influence or importance of input features.

The LIME algorithm works by passing lots and lots of slightly perturbed examples of the original input to the model and then measures how the model predictions change with these small input changes

Local explanations mean that LIME gives explanations that are locally faithful within the surroundings or vicinity of the observation/sample being explained.

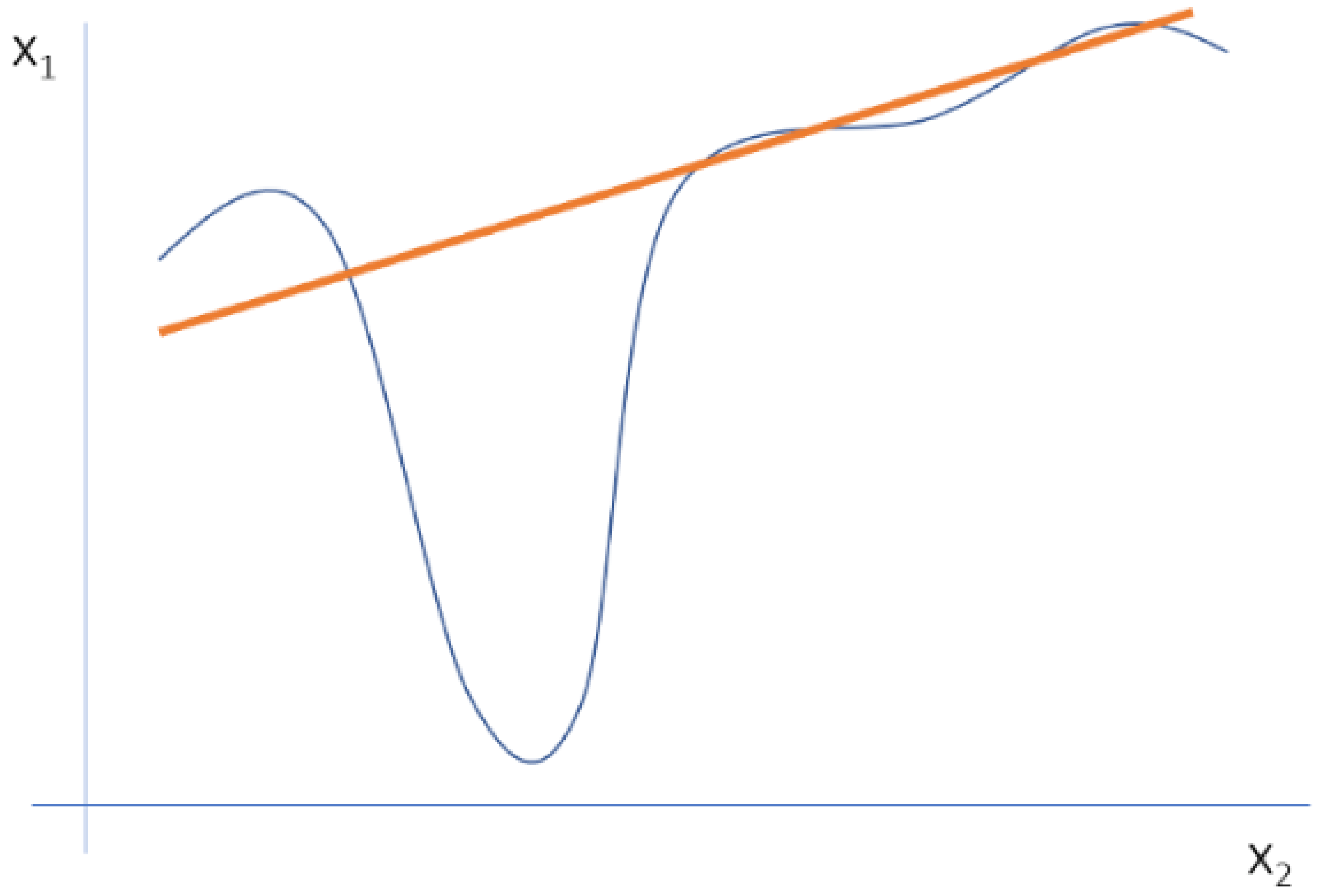
How to generate a perturbation of an image and second, what it means to measure how the model prediction changes on these perturbations.

LIME (Local Interpretable Model-agnostic Explanations)

- LIME segments the image into regions called superpixels.
- LIME then perturbs these interpretable components by setting the pixel values in each superpixel region to a gray value.
- Each of these perturbed instances is then given to the model to generate new prediction values for the class predicted initially.
- LIME quantitatively measures the contributions of the superpixel feature inputs, by training a smaller, interpretable model to provide explanations for the original model.
- Linear models are inherently interpretable because the weights of each feature directly indicate the importance of that feature for the model’s prediction.
- LIME’s elegance in generating weightings by turning regions of the image on and off is also its downfall.
- Since the perturbed images remove entire areas from the prediction, it effectively shifts the model’s attention to what remains visible in the image.

In the current implementation, only linear models are used to approximate local behavior.

A linear approximation of the local behavior for two features is not a good representation and won't capture the highly non-linear behavior of the model.



Guided Backpropagation

It modifies the **backpropagation** process to allow only **positive** contributions to pass backward.

This **filters out irrelevant or suppressing signals**, leaving high-resolution pixel-level details.

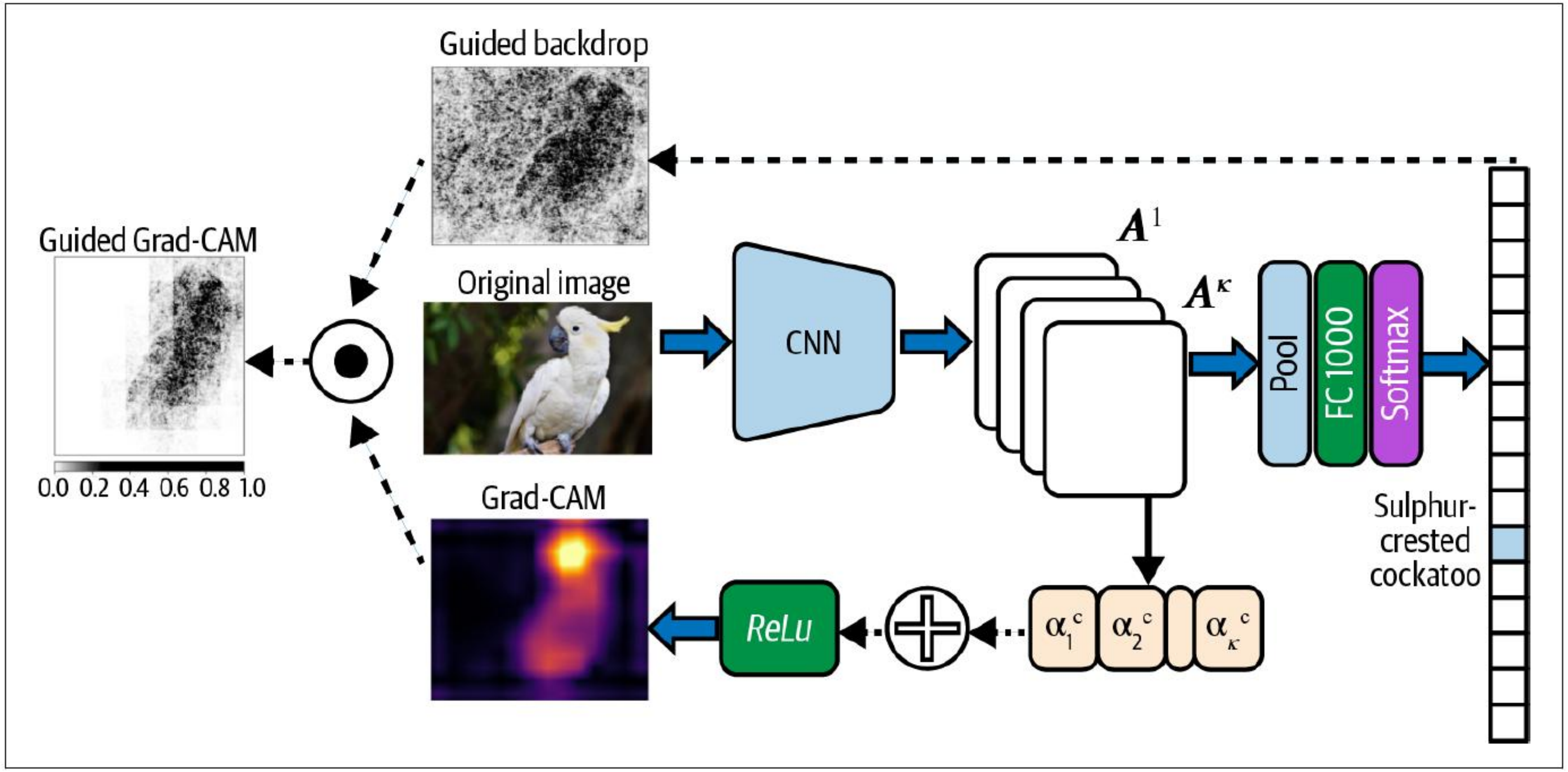
DeConvNets are built upon deconvolution layers, also known as transposed convolutions.

DeConvNets are similar to convolutional networks but work in reverse (reversing filters, reversing pooling, etc.) so that they reconstruct the spatial resolution of the original input tensor.

The idea of using DeConvNets as an explainability method is to visualize the internal activation layers of a trained CNN by “undoing” the convolution blocks of a CNN using deconvolution layers block by block.

Guided Grad-COM

Combines the best of both Grad-CAM and Guided Backprop.



Guided backdrop
Guided Grad-CAM
Original image
CNN
A^1
A^K
Pool
FC 1000
Softmax
0.0 0.2 0.4 0.6 0.8 1.0
Grad-CAM
ReLu
α_1^c
α_2^c
α_K^c
Sulphur-crested cockatoo