Explainable AI

Research Article

From Tables to Pixels: Bridging Explainability Across Tabular and Image-Based AI Models

M Murali Karthick
IIT Palakkad
Palakkad, India
murali47karthick@gmail.com

M Sreenath Karthick
IIT Palakkad
Palakkad, India
msreenathkarthick2005@gmail.com

I. Introduction

Explainable Artificial Intelligence (XAI) refers to techniques and methods that help in understanding and interpreting the decision making process of AI models (particularly a Black-Box Model like a Deep Neural Network) which led the models to arrive at a particular outcome.

The Black Box Problem : Many papers have stated the Explainability or Interpretability vs. Accuracy tradeoff in large complex deep learning models. Even though such models are extremely accurate , lack of understandability has led to lack of trust in the models. As AI is more prevalent than ever in all job domains, it is important to understand and explain or reason the decision made by the model , rather than blind trust.

Explainable AI helps in providing a transparent and more clarified view of the model, which indeed helps us in technical aspects such as debugging and clears confusion societal factors such as accountability, ethics and fairness, which indeed lend more trust in the working of the model.

II. Explainability vs Interpretability

There is a huge debate about what explainability and interpretability mean in the research field of XAI , for the sake of this report we will consider it.

Interpretability : How easily a human can understand the internal mechanics of a model

Explainability : How well can a human explain the prediction of a model after it happens?

Machine learning models are models whose features are human understandable , due to this alone many times the ML models become inherently explainable and interpretable since we train the model based on this feature . for example (Decision Tree, Linear Regression , KNN, SVM (in less feature dimensions) . But this is not the case of Deep Learning models , such as neural networks in which the features are not selected by humans. These are the models which form the primary black box models , which require a model-agnostic explanation to understand which parts of the input data decide the output.

III. Explainability for Tabular Data

For tabular data input, there are two different problems that we solve: Classification (predict a category or class label) and regression (predict a continuous numerical value).

A tabular data set can be considered to consist of two parts:

- **Input features:** $\mathbf{X} = [x_1, x_2, x_3, \ldots, x_n]$, where x_i denotes the value corresponding to the i^{th} feature.
- **Output label:** $\mathbf{Y} = [y]$

Here, both x_i and y are treated as column vectors.

Our goal as XAI ,in this type of dataset, is to identify which feature x_i is the most important feature and how it influences (positively/negatively) the output . And to understand how important a feature can be understood graphically to help us visually understand the relations.

Let's consider the following data set and explore various explanation approaches specifically designed for tabular data.

Our objective is to provide meaningful justifications for the model's predictions on whether an individual's income exceeds 50K. The dataset comprises 32,614 entries with features such as age, work class, education level (numerical), marital status, capital gain, and hours worked per week. A simple feedforward neural network was trained on this dataset and achieved an accuracy of 82.4%.

A. LIME

LIME (Local Interpretable Model Agnostic Explanation) is a popular model agnostic interpretability technique designed to explain the predictions of any black-box model. [5].

i) How does LIME work (Tabular Data)

- **Pick an instance** (that you want to explain).
- **Perturb the input locally** by creating variations around the instance, by slightly changing the feature values of the original instance.
- **Get predictions** from the black-box model for the perturbed instances.

- **Weight the perturbed samples** by their similarity to the original instance (usually using a kernel function).
- **Fit a simple interpretable model** (such as linear regression or lasso regression) to approximate the black box model prediction locally.
- **Interpret the prediction** using the weights (coefficients) of the simple model to determine which characteristics contributed the most to the original prediction.

ii) LIME Notebook (Inference)

To understand the inference capabilities of LIME, let us apply it to a trained model using a sample instance A from the dataset with the following features:

Numerical Features:
- Age = 40
- Capital Gain = 15024
- Capital Loss = 0
- Hours per Week = 55

Encoded Categorical Features:
- Work-Class = 4
- Education Num = 15
- Marital Status = 2
- Occupation = 10
- Relationship = 4
- Race = 4
- Sex = 1
- Country = 11

The model predicts that instance A has an income greater than \$50K.

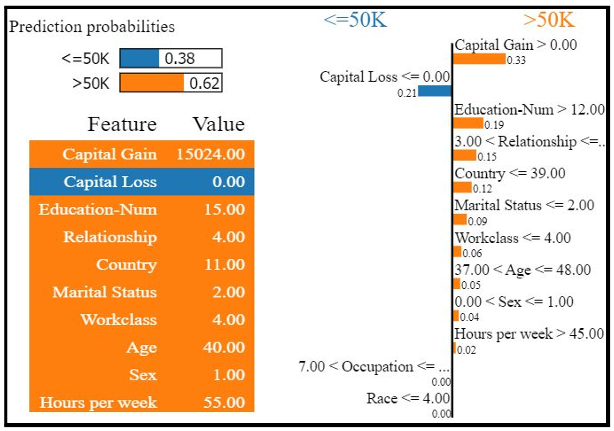

Figure I: LIME Explanation Visualization

From Figure I, we derive several key insights regarding the model's decision:

Prediction Percentage: This indicates the confidence level of the model to classify the instance A. It is derived by counting how many perturbed samples of A are classified as `<=50K` and `>50K`. For this instance:

- 38% of samples were classified as `<=50K`
- 62% of samples were classified as `>50K`

Feature Value Table: This table presents instance A's features in order of importance, colored to indicate which class the feature pushes the decision toward. From this:

- **Capital Loss** and **Occupation** support classification as `<=50K`
- All other features support classification as `>50K`

Feature Contribution Bar Plot: A bar chart representation of each feature's contribution to the final prediction:

- **Capital Gain** contributes 33% toward `>50K`
- **Capital Loss** contributes 21% toward `<=50K`
- Remaining features contribute variably, depending on direction and magnitude

These interpretations make the model's decision transparent and understandable for both developers and stakeholders.

iii) Strengths of LIME

- Intuitive and flexible; works for text, tabular, and image data.
- Provides an explanation for individual predictions.
- Model-agnostic — can be used with any black-box model.
- Helps uncover biases and errors in the model at a local level.

iv) Limitations of LIME

- Explanations are local and may not represent the global behavior of the model.
- The choice of perturbation method and similarity kernel can significantly affect the results.
- Computationally expensive when many perturbed samples are required.

B. SHAP

i) SHAPley Values

SHAP (SHapley Additive exPlanations) values explain how much each input feature contributes to a specific prediction made by a machine learning model. They are based on Shapley values from cooperative game theory and fairly distribute the difference between the actual prediction and the average prediction across all features. [4].

Positive SHAP values push the prediction higher, while negative values push it lower.

ii) SHAP Framework

SHAP is a unified framework for explaining the output of any machine learning model. It attributes the prediction of the model to its input features using Shapley values.

There are several model-agnostic and model-specific methods to compute SHAP values.

- **Kernel SHAP**: Suitable for any model.
- **Tree SHAP**: Preferred for tree-based models like XGBoost, Random Forest, etc.
- **Deep SHAP**: Designed for deep learning models.

iii) How SHAP Works

- For a given instance, generate many synthetic samples with various masked features.
- Evaluate the model on these samples.
- Use the model predictions and masking patterns to solve a weighted linear regression to estimate the contribution of each feature.

iv) Advantages of SHAP

- **Model-agnostic & model-specific**: SHAP works with any type of model, tree-based, linear, deep learning, etc.
- **Consistent & fair**: Ensures consistent feature attribution based on game theory principles.
- **Local & global explanations**: Offers both instance-level and dataset-level insights.
- **Visualization tools**: Provides intuitive visuals like force plots and summary plots to aid understanding.

v) Disadvantages of SHAP

- **Computationally expensive**: The exact computation of the SHAP value can be slow, especially for large datasets or complex models.
- **Interpretation complexity**: Difficult to interpret the results in high-dimensional settings.
- **Assumes feature independence**: May produce misleading explanations when features are highly correlated.

vi) SHAP Visualizations

a) Summary Plot (Global): A SHAP summary plot shows the impact of each feature on the model output in all samples. Each dot represents one instance, with the color indicating the feature value (red = high, blue = low) and its position showing the SHAP value (positive = increase prediction, negative = decrease prediction).

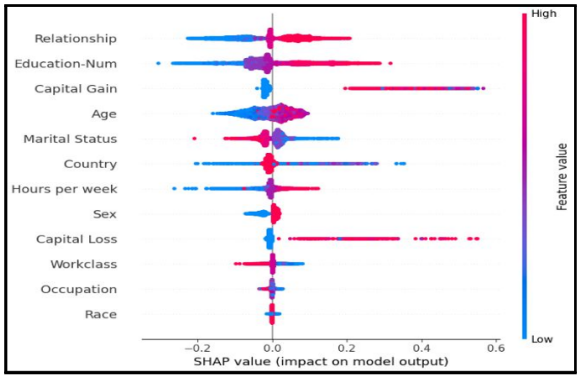

FIGURE II: SHAP SUMMARY PLOT

Inference:

- 'Capital Gain' and 'Relationship' have the strongest influence on the prediction of the model.
- High capital gain (red) pushes predictions positively (right), indicating higher predicted income.
- Low relationship values (blue) reduce the prediction (left).
- Features like "Race" and "Occupation" have minimal impact (clustered near zero).

b) Aggregation / Summary Bar Plot (Global): This bar plot shows the average absolute SHAP values for each feature, representing the overall importance in the model. Higher mean SHAP values indicate a greater contribution.

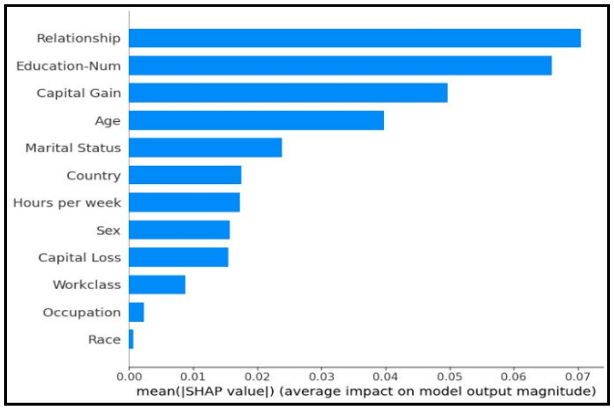

FIGURE III: SHAP SUMMARY BAR PLOT

Inference:

- "Relationship" is the most influential feature, followed by "Education-Num" and "Capital Gain".
- "Race" and "Occupation" have a low impact and rarely influence predictions significantly.

c) Waterfall Plot (Local): The SHAP waterfall plot shows how each feature contributes to shifting the model's prediction from the base value (expected prediction) to the final prediction for a specific instance.

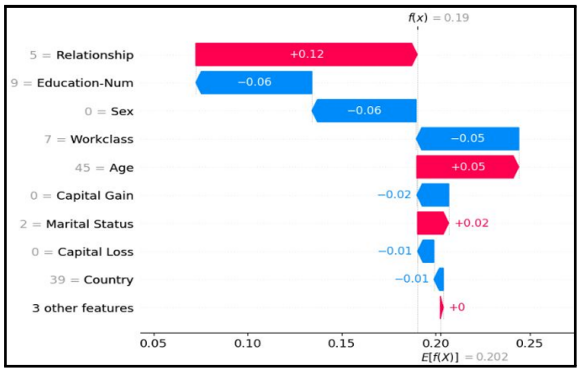

FIGURE IV: SHAP WATERFALL PLOT

Inference:

- Base value (E[f(X)]): 0.202
- Final prediction (f(X)): 0.19, classified as ≤50K
- More features reduce the prediction (blue), with fewer red ones trying to increase it.

d) Force Plot: This plot visualizes how individual features push the base prediction to the final result. It is a 1D plot unlike the waterfall and does not explicitly sort features by impact.

Figure V: SHAP Force Plot

Inference:

- Base value (E[f(X)]): 0.202
- Final prediction (f(X)): 0.62 ⇒ >50K classification
- Positive contributors: Sex = 1, Marital Status = 2, Relationship = 4, Age = 50, Capital Gain = 3103.0
- Negative contributors: Education-Num, Capital Loss = 0

e) Dependency Plot: A SHAP dependency plot shows how the value of a feature (X-axis) affects its SHAP value (Y-axis). Color represents a second feature to reveal interactions.

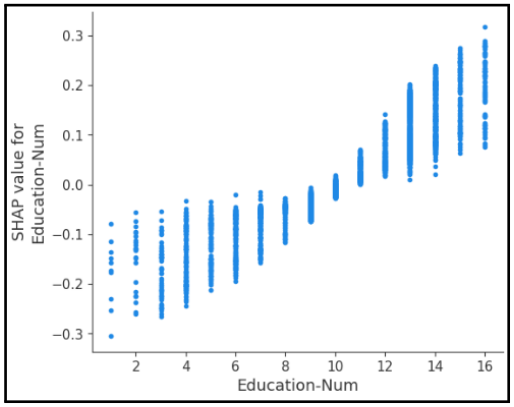

Figure VI: SHAP Dependency Plots: Education-Num

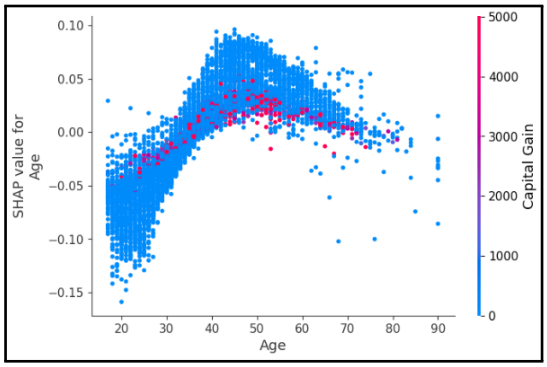

Figure VII: SHAP Dependency Plots: Age (colored by Capital Gain)

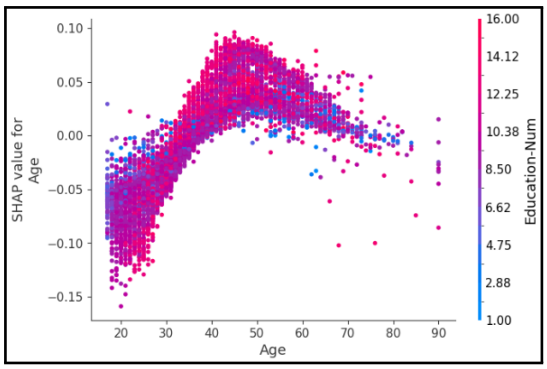

Figure VIII: SHAP Dependency Plots: Age (colored by Education-Num)

Inference:

- *(Education-Num)*: Higher education levels linearly increase the SHAP value. Low levels (1–6) have a negative impact.
- *(Age, colored by Capital Gain)*: SHAP value rises until 40 years, then declines. High capital gain (red) amplifies the effect.
- *(Age, colored by Education-Num)*: Higher education improves the positive effect of age, especially between ages 30–50.

IV. Explainability for Image Data

An **image dataset** typically consists of two components: the input data (a collection of images) and the output labels (representing the corresponding classes for those images). Most machine learning models trained on image datasets are used primarily for **classification purposes**.

Each image is treated as a **3D matrix** — defined by height, width, and color channels. Due to this structural complexity, **traditional machine learning models** often struggle to classify images accurately.

This is where **black-box models**, particularly deep learning models such as **Convolutional Neural Networks (CNNs)**, become essential. These models contain millions of parameters that allow them to automatically learn intricate patterns and features in the images - often features that are not defined by humans or interpretable directly.

However, despite their high **accuracy**, these models often **lack interpretability** due to their complexity. This creates a pressing need for visualization and interpretation techniques that can help us understand what the model has 'seen' or learned during classification.

To address this interpretability challenge, we apply **explainability techniques** such as:

1. **LIME** (Local Interpretable Model-agnostic Explanations) – adapted for image data
2. **CAM** (Class Activation Maps)

These techniques are applied in standard CNN archi-

tectures like **VGG16**, **VGG19**, and **ResNet50**, which have been **pre-trained on the ImageNet dataset**.

A) LIME for Image Data

LIME (Local Interpretable Model-agnostic Explanations) explains the prediction of any black-box model (like a CNN) by approximating it locally using a simple interpretable model (e.g., linear or lasso regression), focusing on the region around a specific prediction.

i) How LIME Works for Image Data

- **Input Image and Prediction:**
  - Provide an image to the pre-trained CNN.
  - Obtain the predicted class of the model.
- **Segment the Image into Superpixels:**
  - The image is broken into superpixels using a segmentation algorithm.
  - Each superpixel is a group of connected pixels with similar color/texture.
- **Create Perturbed Samples:**
  - Randomly turn ON/OFF superpixels (i.e., mask them with grey/black).
  - Generate multiple such perturbed images (e.g., 1000 samples).
  - Each perturbed image is fed into the black-box model, and the predicted probability for the original class is recorded.
- **Create a Binary Matrix:**
  - Each row corresponds to a perturbed image.
  - Each column indicates whether a particular superpixel was ON (1) or OFF (0).
  - The model output score forms the target variable for each row.
- **Fit a Local Interpretable Model:**
  - A simple interpretable model (e.g., linear regression) is trained on the binary matrix.
  - This model estimates how each superpixel contributes to the original prediction.
- **Rank Superpixels by Importance:**
  - The regression coefficients indicate which superpixels positively or negatively influence the prediction.
  - The most important superpixels are marked and visualized.

ii) Visualization Techniques

- **a) Image Mask Visualization:**
  - Highlights the top 5-10 superpixels that contributed the most to the prediction.
  - Positive contribution superpixels are often shown in green, and negative ones in red.
  - All other superpixels are shown with faded or intermediate coloring.
  - Helps identify the critical regions that influence the decision of the model.
- **b) Heatmap Visualization:**
  - Uses regression weights to generate a heatmap:
    - High positive weights: bright red to yellow.
    - High negative weights: dark blue to green.
    - Neutral/no impact: gray or faded out.
  - Helps distinguish relevant versus irrelevant regions of the image.

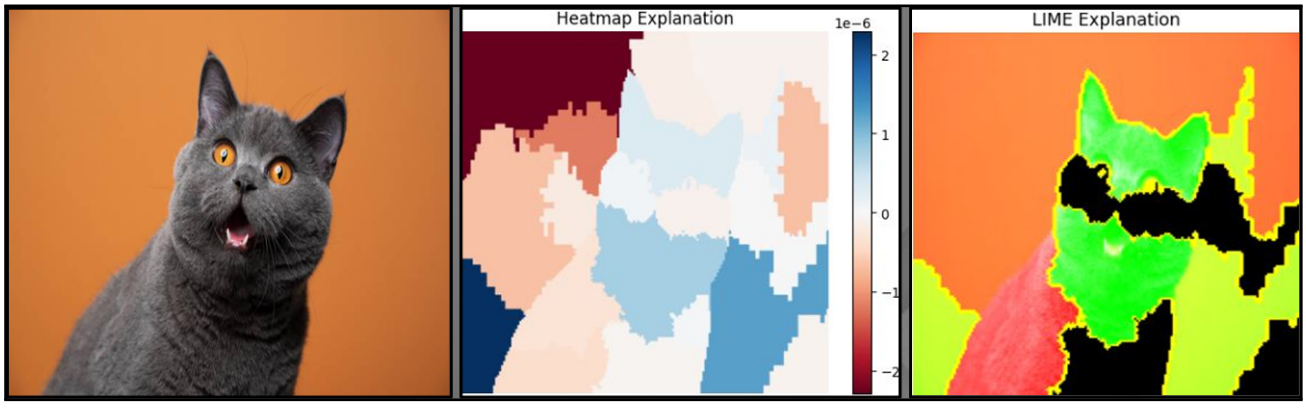

Figure IX: Heat Map and Image Mask Vizualisation

A) CAM (Class Activation Map)

CAM is a technique used to visualize the part of an input image that a CNN focuses on when making a particular classification decision. This is super useful for model interpretability,especially in computer vision tasks like image classification.

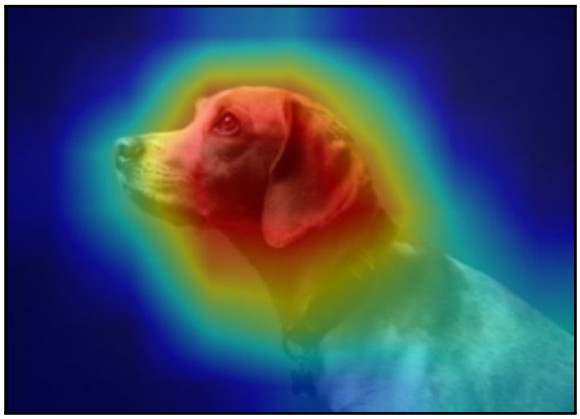

Figure X: Heat Map of CAM Vizualisation

i) Step-by-step Process of CAM

- **Input Image:**
  - A preprocessed image (resized, normalized) is fed into the CNN.
- **Feature Extraction:**
  - The image passes through convolutional layers, producing multiple feature maps from the last convolutional layer.
  - These feature maps capture spatial patterns (such as edges, textures, object parts, etc.), denoted as $f_k(x, y)$ for the k-th feature map.
- **Global Average Pooling (GAP):**
  - Instead of flattening, the architecture is modified to include a GAP layer.
  - This layer computes the average of each feature map across spatial dimensions:

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

  - F_k is a single value representing the overall strength of the k-th feature.
- **Output to Classifier:**
  - The pooled values F_k are passed to the final classifier layer.
- **Compute Class Score:**
  - Identify the target class c and retrieve the weights w_k^c connecting each feature map to class c.
  - Calculate the class score:

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

- **Generating the Class Activation Map (CAM):**
  - Multiply each feature map $f_k(x, y)$ by its corresponding class-specific weight w_k^c.
  - Compute the weighted sum to obtain CAM:

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

- **Upsample the CAM:**
  - Resize the CAM from its original size (e.g. 7×7 or 14×14) to the input image size (e.g. 224×224) using bilinear interpolation.
- **Overlay on Original Image:**
  - Normalize CAM values to the range $[0, 1]$.
  - CAM overlays are used as heatmaps (e.g. red = high importance, blue = low) on top of the original image.

ii) Types of CAM

- Grad-CAM
- Grad-CAM++
- Ablation CAM
- Layer CAM

*a) **Grad-CAM** (Gradient-weighted Class Activation Mapping)*

Grad-CAM is a powerful visualization technique that highlights regions in the input image that were important for a CNN prediction, using the gradients of the target class that flow into the final convolutional layer. [6].

Unlike standard CAM, which requires modifying the model (e.g., inserting GAP layers), Grad-CAM works without architectural changes and can be applied to a wide range of CNN models.

Step-by-step Process of Grad-CAM

- **Input Image:**
  - A preprocessed image (resized and normalized, for example, 224×224) is fed into a pre-trained CNN such as ResNet or VGG.
- **Forward Pass:**
  - The image is passed through the convolutional layers to compute the feature maps $A_k(x, y)$ for each channel k.
- **Identify Target Class:**

  - Perform a forward pass to compute the class scores.
  - Select the target class c (either the predicted class or a specific class of interest).
- **Backward Pass (Gradient Computation):**
  - Compute the gradient of the class score S_c with respect to each feature map:

$$\frac{\partial S_c}{\partial A_k(x,y)}$$

  - These gradients indicate the importance of each feature map to the target class prediction.
- **Compute Importance Weights:**
  - Average the gradients spatially to get the importance weight for each feature map:

$$\alpha_k = \frac{1}{Z} \sum_x \sum_y \frac{\partial S_c}{\partial A_k(x,y)}$$

  - where Z is the number of pixels in the feature map.
- **Generate Grad-CAM Heatmap:**
  - Compute a weighted combination of feature maps followed by a ReLU:

$$\text{GradCAM}_c(x,y) = \text{ReLU}\left(\sum_k \alpha_k A_k(x,y)\right)$$

  - ReLU is applied to focus only on features that positively influence the class score.
- **Up-sample (Resize the Grad-CAM):**
  - The heatmap (usually low-resolution as 7×7) is up-sampled to the input image size (e.g., 224×224) using bilinear interpolation.
- **Normalize and Overlay:**
  - Normalize Grad-CAM values to the range $[0, 1]$.
  - Overlay the heatmap on the original image.
    - Red regions indicate high importance.
    - Blue regions indicate low importance.

Why Grad-CAM is powerful:

- Works with any Convolutional Neural Network (CNN).
- Does not require modifying the architecture (e.g., adding GAP layers).
- Uses gradients only.
- Highlights local regions that provide evidence for a specific class prediction.

b) ***Grad-CAM++*** *(Gradient-weighted Class Activation Mapping Plus Plus)*

Grad-CAM++ is an enhanced version of Grad-CAM that provides better localization, especially when multiple instances of the same object class appear in an image. Refine the importance weight computation using second-order and third-order gradients, resulting in sharper, more class-discriminative heatmaps. Like Grad-CAM, it does not require changes to the network architecture. [1]

Step-by-step Process of Grad-CAM++

- **Input Image:**
  - A preprocessed image (resized and normalized, for example, 224×224) is fed into a pre-trained CNN (for example, ResNet, VGG).
- **Forward Pass:**
  - The image is passed through convolutional layers to compute feature maps $A_k(x,y)$.
- **Identify Target Class:**
  - Perform a forward pass to get the output class scores.
  - Select the target class c (usually the predicted class).
- **Backward Pass (Higher-Order Gradient Computation):**
  - Compute the following derivatives of class score S_c w.r.t. feature maps $A_k(x,y)$:

$$\frac{\partial S_c}{\partial A_k(x,y)}, \quad \frac{\partial^2 S_c}{\partial A_k(x,y)^2}, \quad \frac{\partial^3 S_c}{\partial A_k(x,y)^3}$$

  - These capture non-linear interactions, aiding in handling multiple object instances.
- **Compute Importance Weights $\alpha_{x,y}^k$ and w_k^c:**
  - For each pixel (x,y) on the feature map k, compute a pixel-wise weight $\alpha_{x,y}^k$ using the formula derived from the gradients.
  - Compute the final channel weight:

$$w_k^c = \sum_{x,y} \alpha_{x,y}^k \cdot \text{ReLU}\left(\frac{\partial S_c}{\partial A_k(x,y)}\right)$$

- **Generate Grad-CAM++ Heatmap:**
  - Use the weights to compute a weighted sum of feature maps:

$$\text{GradCAM++}_c(x,y) = \text{ReLU}\left(\sum_k w_k^c A_k(x,y)\right)$$

  - ReLU filters out negative influences.
- **Up-sample (Resize the Heatmap):**

  - The heatmap is up-sampled from its native size (e.g., 7×7 or 14×14) to the input size (e.g., 224×224) using bilinear interpolation.

- **Normalize and Overlay:**
  - Normalize the heatmap values to $[0, 1]$.
  - Overlay the heatmap on the original image.
  - Red = strong importance; Blue = weak importance.

Why Grad-CAM++ is Better

- Captures fine-grained spatial attention, especially for multiple instances of the same object.
- Uses second- and third-order gradients for better precision and pixel-wise importance.
- Works with existing CNN architectures without requiring changes (e.g., no GAP layer needed).
- Produces sharper and more class-specific heatmaps compared to Grad-CAM.
- Particularly effective when the same class appears in multiple locations in the image.

*c)**Ablation-CAM***

Ablation-CAM is a class activation mapping technique that enhances interpretability by directly measuring the importance of each channel (feature map) through ablation, that is, systematically removing or zeroing out channels and observing the effect on the model's output. Unlike Grad-CAM and Grad-CAM++, Ablation-CAM does not rely on gradients, making it more stable and architecture-agnostic. [2].

Step-by-step Process

- **Input Image:**
  - A preprocessed image (e.g., resized to 224×224 and normalized) is input to a CNN (e.g. VGG, ResNet).
- **Forward Pass:**
  - The image passes through the network to extract feature maps at a selected convolutional layer.
  - Denote the feature maps by $A_k(x, y)$ for each channel k.
- **Identify Target Class:**
  - Perform a forward pass to compute the class score S_c for the predicted or chosen target class c.
- **Ablation Analysis (No Gradients):**
  - For each feature map A_k, ablate (set to zero) that channel and recompute the class score $S_c^{(A_k=0)}$.
  - Calculate the change in class score:

$$\Delta S_k = S_c^{(\text{original})} - S_c^{(A_k=0)}$$

  - This reflects the contribution of feature map k to the prediction.
  - The importance weight for each feature map is then:

$$w_k = \Delta S_k$$

- **Generate Ablation-CAM Heatmap:**
  - Combine the feature maps linearly using the computed weights:

$$\text{Ablation-CAM}_c(x, y) = \text{ReLU}\left(\sum_k w_k A_k(x, y)\right)$$

  - The ReLU is applied to retain only positively influential regions.
- **Up-sample:**
  - Since feature maps are usually low-resolution (e.g., 14×14), the heatmap is up-sampled to the input size (e.g., 224×224) using bilinear interpolation.
- **Normalize and Overlay:**
  - Normalize the heatmap values to the range $[0, 1]$.
  - Overlay the heatmap on the original image.
  - Red = high importance; blue = low importance.

Why Ablation-CAM is Better

- Works without computing gradients, making it more numerically stable, especially for models with noisy or zero gradients (e.g., ReLU dead neurons).
- Directly measures importance through ablation, rather than relying on gradient-based approximations.
- Does not require backward hooks or gradient flow—suitable for models with custom or complex architectures.
- Provides more faithful and causal explanations of the model behavior.
- Performs well on images with multiple objects of the same class, offering localized attribution similar to Grad-CAM++.

*d) **Layer-CAM***

Layer-CAM is an advanced class activation mapping technique that generates fine-grained and class-discriminative visual explanations using positive gradients at the locations of individual pixels in the feature maps.

Unlike Grad-CAM, which averages gradients across spatial locations, Layer-CAM preserves spatial specificity, enabling better localization, especially at early or intermediate CNN layers. [3].

Step-by-step Process

- **Input Image:**
  - A preprocessed image (e.g., resized to 224×224, normalized) is input to a CNN (e.g. VGG, ResNet).
- **Forward Pass:**
  - The image is passed through the network to extract feature maps in a chosen convolutional layer.
  - Denote the feature maps at this target layer as $A_k(x, y)$ for each channel k.
- **Identify Target Class:**
  - Perform a forward pass to obtain the class scores.
  - Choose the target class c (usually the predicted class) and denote its score as S_c.
- **Backward Pass (Gradient Computation):**
  - Compute the gradient of the class score S_c with respect to each pixel of the feature maps:

$$\frac{\partial S_c}{\partial A_k(x, y)}$$

- **Compute Layer-CAM Activation:**
  - Multiply each feature map $A_k(x, y)$ with its corresponding positive gradient:

$$M_k(x, y) = \text{ReLU}\left(\frac{\partial S_c}{\partial A_k(x, y)} \cdot A_k(x, y)\right)$$

  - This captures pixel-wise importance through element-wise multiplication rather than global averaging.
- **Aggregate Over Channels:**
  - Sum across all channels to produce the final Layer-CAM heatmap:

$$\text{Layer-CAM}_c(x, y) = \sum_k M_k(x, y)$$

- **Up-sample:**
  - As the heatmap is low-resolution (e.g., 14×14), up-sample it to the input image size using bilinear interpolation.
- **Normalize and Overlay:**
  - Normalize the heatmap to the range $[0, 1]$.
  - Overlay it on the original image as a heatmap:
    - Red = High class importance
    - Blue = Low class importance

Why Layer-CAM is Better

- Retains pixel-level gradient information instead of averaging across the spatial map, leading to sharper and more localized saliency maps.
- Does not require global average pooling of gradients, allowing capture of fine-grained features even from shallow- or mid-level layers.
- Effectively distinguishes individual object parts, even when multiple instances of the same class are present.
- Can be applied to earlier layers which encode edge or texture features - useful for high-resolution tasks (e.g., segmentation).
- Improves interpretability by identifying the most influential pixels for each class decision.

V. Quantitative Evaluation of Class Activation Maps (CAMs

In quantitative evaluation of CAMs, the goal is to objectively measure how well a CAM localizes or highlights the actual object of interest in an image. This is particularly important when comparing different CAM techniques such as Grad-CAM, Grad-CAM++, Layer-CAM, Ablation-CAM, etc.

1. Intersection over Union (IoU)

Definition: IoU measures the overlap between the predicted region (obtained by thresholding the CAM heatmap) and the ground truth (GT) object region (bounding box or segmentation mask).

$$\text{IoU} = \frac{|\text{CAM} \cap \text{GT}|}{|\text{CAM} \cup \text{GT}|}$$

- Numerator: Number of pixels where both CAM and GT are active (overlap).
- Denominator: Total number of pixels active in either CAM or GT.

Interpretation: A higher IoU indicates better alignment with the true object. It is a direct measure of the quality of *localization.*

2. Recall

Definition: Recall tells us how much of the ground-truth object is covered by the CAM heatmap.

$$\text{Recall} = \frac{|\text{CAM} \cap \text{GT}|}{|\text{GT}|}$$

Interpretation: Recall captures *completeness* —

whether the CAM successfully includes the entire object. A high recall implies that most of the ground truth region is activated, even if it includes irrelevant parts.

3. Precision

Definition: Precision indicates how much of the CAM-activated region actually lies within the object.

$$\text{Precision} = \frac{|\text{CAM} \cap \text{GT}|}{|\text{CAM}|}$$

Interpretation: Precision captures *focus* — how accurately CAM highlights the actual object without bleeding into the background. High precision means that the activation is mostly within the object region.

4. F1-Score

Definition: F1-score is the harmonic mean of Precision and Recall.

$$\text{F1} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Interpretation: The F1-score balances both completeness and accuracy. It is especially useful when one metric is high and the other is low.

5. Saliency Score (Optional but Important)

Definition: The saliency score measures how much of the CAM's energy (heatmap intensity) lies inside the ground truth object.

$$\text{Saliency Score} = \frac{\sum_{(x,y)\in \text{GT}} \text{CAM}(x,y)}{\sum_{\text{all pixels}} \text{CAM}(x,y)}$$

Interpretation: This metric is similar to precision, but considers the heatmap intensity rather than the binary mask. It is useful for soft CAMs and avoids arbitrary thresholding. A high saliency score indicates that CAM puts more energy where it matters, inside the true object region.

VI. CAM Results Analysis

In this section, we analyze various Class Activation Mapping (CAM) techniques—namely Grad-CAM, Grad-CAM++, Ablation-CAM, and Layer-CAM—to evaluate their ability to localize objects of interest in images. We quantitatively assess their performance using metrics such as Intersection over Union (IoU), Recall, Precision, F1-Score, and Saliency Score to ensure objective comparison of localization quality and focus.

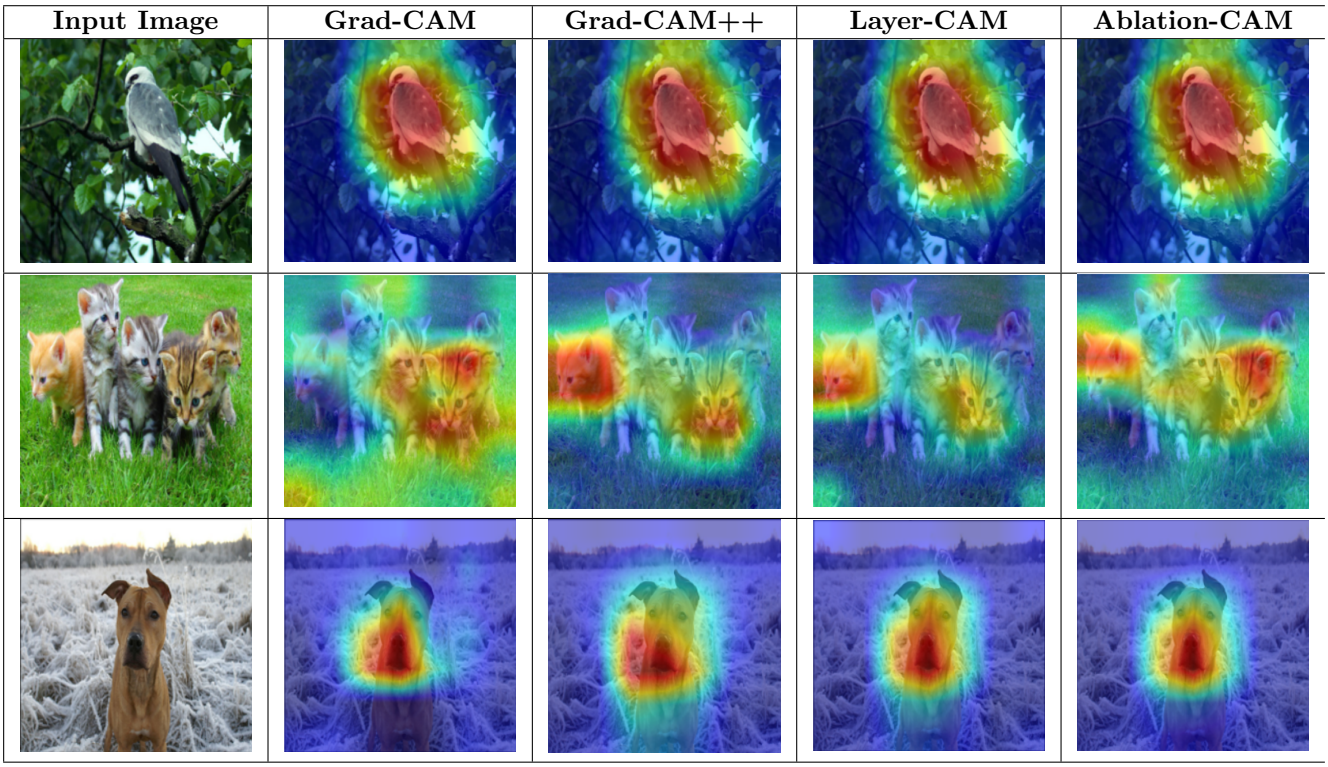

Table I: Comparison of CAM methods across 3 examples

We calculate the performance of each CAM technique on a sample image for which the ground truth bounding box is available. This allows us to quantitatively compare how well each method highlights the true region of interest. By thresholding CAM outputs and comparing them with the ground truth mask, we compute localization metrics such as IoU, Recall, Precision, F1-Score, and Saliency Score.

FIGURE XI: ORIGINAL IMAGE

FIGURE XII: GROUND TRUTH MASK

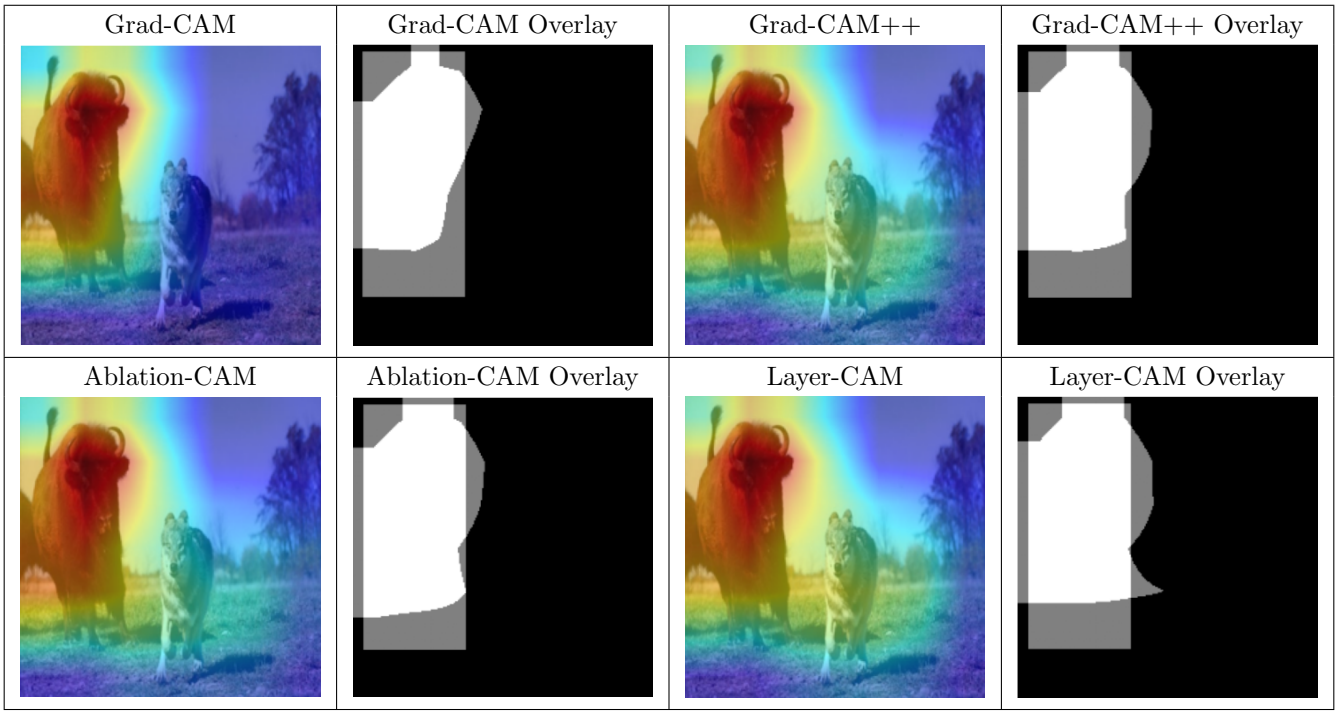

TABLE II: VISUALIZATION OF VARIOUS CAM METHODS AND THEIR OVERLAY WITH GROUND TRUTH BOUNDING BOXES.

CAM Method	Saliency	IoU	Recall	Precision	F1-Score
Grad-CAM	0.6959	0.5971	0.6566	0.8683	0.7478
Grad-CAM++	0.6767	0.6410	0.7326	0.8369	0.7813
Ablation-CAM	0.6904	0.6777	0.7723	0.8469	0.8079
Layer-CAM	0.6671	0.6457	0.7644	0.8061	0.7847

TABLE III: QUANTITATIVE EVALUATION OF DIFFERENT CAM METHODS

Result Analysis

Table III presents the quantitative evaluation of four Class Activation Mapping (CAM) techniques: Grad-CAM, Grad-CAM++, Ablation-CAM, and Layer-CAM. The assessment is based on key metrics such as Saliency score, Intersection over Union (IoU), Recall, Precision, and F1-score, providing insight into the localization accuracy and discriminative power of each method.

Grad-CAM, while being the foundational and most general method, shows the lowest IoU (0.5971) and Recall (0.6566) among the evaluated techniques. However, it maintains a high Precision (0.8683), indicating that the

regions it does highlight are often relevant, albeit less comprehensive. Grad-CAM requires no architectural changes and relies solely on first-order gradients, making it easy to implement across a wide variety of CNN models. It is particularly effective in identifying key discriminative regions that influence class predictions, though it may produce coarser and less spatially precise saliency maps due to the averaging of gradients across spatial locations.

Grad-CAM++ improves upon Grad-CAM by incorporating higher-order gradient information (second- and third-order derivatives), allowing finer spatial discrimination in class activation maps. This results in a better IoU (0.6410) and F1-score (0.7813) than Grad-CAM, particularly for cases involving multiple occurrences of the same object class. Its sharper and more focused heatmaps make it suitable for applications demanding precise localization, all while remaining compatible with existing CNN architectures without modifications such as Global Average Pooling (GAP).

Ablation-CAM exhibits the strongest overall performance, achieving the highest IoU (0.6777), Recall (0.7723), Precision (0.8469), and F1-score (0.8079). This superior performance stems from its gradient-free design, where neuron importance is assessed through systematic ablation. This approach yields more faithful and causal explanations, especially in scenarios where gradient information may be unreliable or noisy (e.g., due to ReLU saturation or vanishing gradients). Furthermore, Ablation-CAM does not require backward hooks or gradient flow, making it suitable for models with custom or non-standard architectures. It also performs robustly on images with multiple object instances of the same class, offering class-specific and localized attention similar to Grad-CAM++.

Layer-CAM performs competitively with a Recall of 0.7644 and an F1-score of 0.7847. Its key advantage lies in retaining pixel-level gradient information rather than averaging over spatial locations, which results in sharper and more localized saliency maps. By avoiding global average pooling, Layer-CAM is able to highlight fine-grained features, particularly from shallower or intermediate convolutional layers. This enables the method to capture edges and textures more effectively—especially beneficial for tasks such as semantic segmentation. Moreover, Layer-CAM is capable of distinguishing between multiple object instances, thereby enhancing both interpretability and class discrimination.

Summary:

- **Ablation-CAM** is the most robust and accurate, offering gradient-free, faithful explanations with strong localization.
- **Layer-CAM** provides high-resolution, spatially precise maps by preserving pixel-wise gradients and utilizing intermediate layers.
- **Grad-CAM++** balances complexity and precision using higher-order gradients for sharper, more class-specific saliency.
- **Grad-CAM** remains a practical and general-purpose method, ideal for rapid deployment and baseline comparisons.

These distinctions highlight that the choice of CAM method should align with the specific task requirements—e.g., high interpretability and resolution (Layer-CAM), gradient instability robustness (Ablation-CAM), or implementation simplicity (Grad-CAM).

References

[1] Aditya Chattopadhay, Anirban Sarkar, Prantik Howlader, and Vineeth N Balasubramanian. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *2018 IEEE winter conference on applications of computer vision (WACV)*, pages 839–847. IEEE, 2018.

[2] saurabh desai and Harish Guruprasad Ramaswamy. Ablation-cam: Visual explanations for deep convolutional network via gradient-free localization. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, March 2020.

[3] Peng-Tao Jiang, Chang-Bin Zhang, Qibin Hou, Ming-Ming Cheng, and Yunchao Wei. Layercam: Exploring hierarchical class activation maps for localization. volume 30, pages 5875–5888, 2021.

[4] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[5] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[6] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.