Explainability AI (XAI) - Unveiling the Black Box

Sailesh Babu Ram Mannuru Manoj Padala

May 03, 2024

1. Abstract:

Deep learning models are integral to critical applications across healthcare, finance, and autonomous systems, necessitating high levels of trust, transparency, and interpretability. This project addresses these needs by developing and assessing state-of-the-art explainability techniques. Our objectives include investigating and implementing feature attribution methods, saliency maps, and model-agnostic approaches to demystify model decisions. We aim to create user-friendly tools and visualizations that enhance the accessibility of model interpretations. The methodology encompasses a thorough literature review, and implementation of various explainability techniques like Integrated Gradients, **SHAP**, **LIME**, **Grad-CAM**, **Occlusion Sensitivity**, **Class Activation Maps(CAM)** and **Deep Feature Factorisation(DFF)**followed by the development of an interactive platform compatible with major deep learning frameworks. We will evaluate these methods across various architectures and domains to determine their effectiveness in improving model transparency and user trust. The expected outcomes include a suite of validated explainability techniques and a comprehensive tool for interpreting deep learning predictions, ultimately contributing to informed decision-making and model refinement in practice. This project proposes a robust framework for enhancing the explainability of deep learning models, promoting broader adoption and trust in automated systems.

2. Description of the Dataset:

As this project focuses on explainability techniques rather than training models, we do not utilize a specific dataset. Instead, we leverage pre-trained models and sample images to demonstrate the effectiveness of explainability techniques.

3. Description of the Deep Learning Network and Training Algorithm:

Since we do not train a model from scratch, we utilize pre-trained convolutional neural networks (CNNs) such as ResNet-50. These models have been trained on large-scale datasets like ImageNet and demonstrate strong performance in various visual recognition tasks. We focus on the architecture and internal mechanisms of these models to apply explainability techniques.

4. Explainability Techniques:

In this section, we will discuss the following explainability techniques:

- Class Activation Maps
- Deep Feature Factorization
- LIME (Local Interpretable Model-agnostic Explanations)
- SHAP (SHapley Additive exPlanations)
- Grad-CAM (Gradient-weighted Class Activation Mapping)
- Occlusion

4.1 Class Activation Maps (CAMs):

Description: Class Activation Maps (CAMs) are a type of visualization technique used to understand the decision-making process of convolutional neural networks (CNNs). They provide insights into which regions of an input image contribute most to the model's prediction. CAMs highlight the discriminative regions of an image by generating heatmap overlays, indicating where the model focuses its attention during classification.

This technique is particularly useful for understanding which parts of an image are most relevant for a particular class prediction. Workflow:

- **Feature Extraction**: CAMs utilize the activations of the last convolutional layer in the CNN, which capture high-level features of the input image.
- **Global Average Pooling (GAP)**: The feature maps are subjected to global average pooling to obtain a summary of each feature map.
- **Weighted Combination**: The class score is computed by taking a weighted combination of the feature map activations, where the weights are derived from the gradients of the predicted class score with respect to the feature maps.
- **Heatmap Generation**: The weighted combination results in a class activation map, which is then upsampled to the original image size to produce a heatmap overlay.

Mathematical Explanation:

- **Feature Extraction**: Let A^l denote the activation map of the last convolutional layer.
- **Global Average Pooling (GAP)**: The GAP operation computes the class activation map M by taking the weighted average of the activation map A^l for each feature map k, where the weights w_k are derived from the gradients of the predicted class score y_c with respect to the feature maps:

$$M = \sum_k w_k A_k^l$$

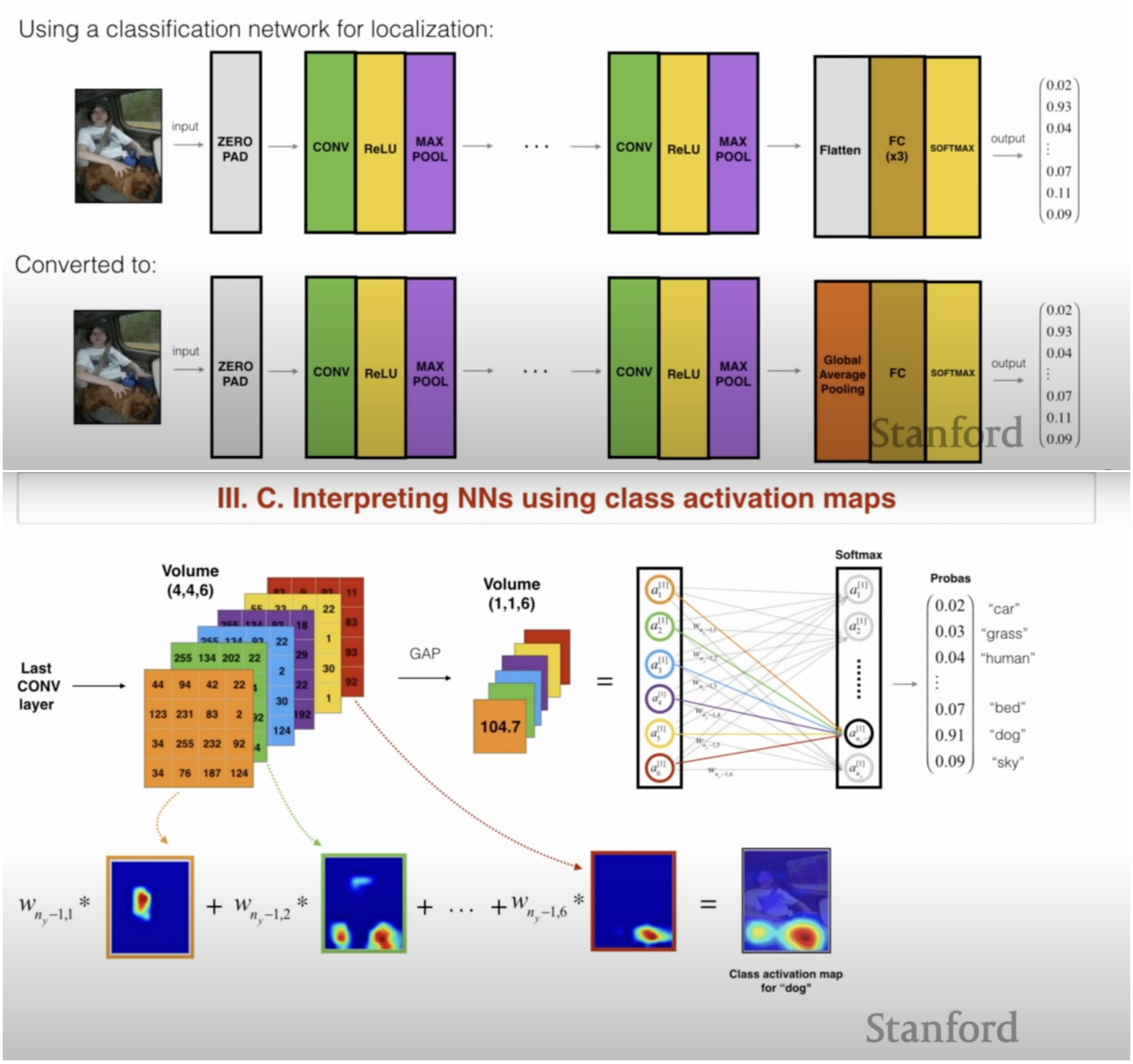
- **Heatmap Generation**: The class activation map M highlights the regions of the input image that contribute the most to the prediction of a specific class c.

Workflow:

1. Forward pass through the pre-trained CNN to obtain the feature maps.
2. Compute the gradients of the predicted class score with respect to the feature maps.
3. Apply GAP to obtain the class activation map.
4. Generate a heatmap overlaying the activation map on the input image.

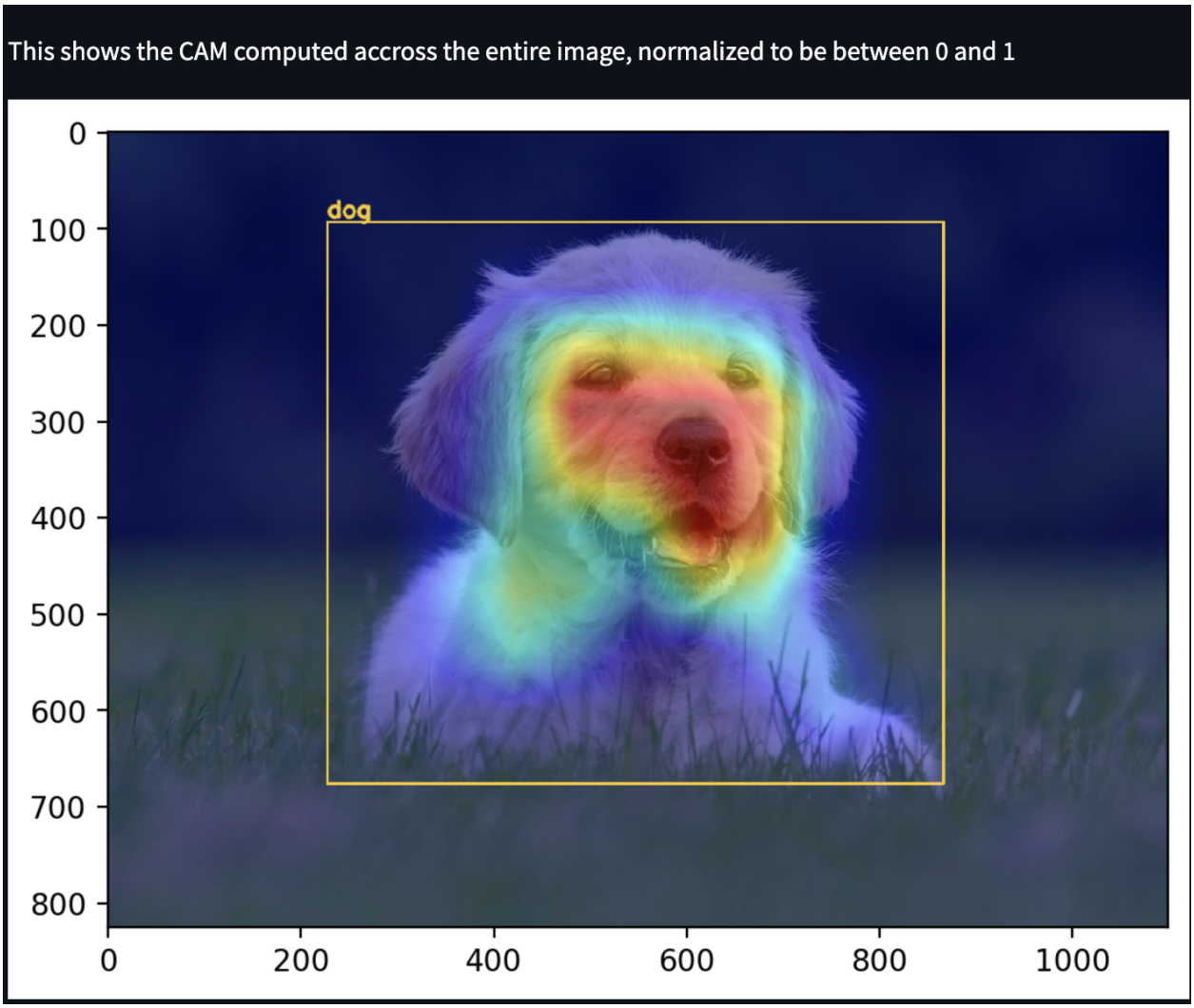
CAMS in a view: These two images are taken from CS203, Neural Networks course.

1. The first image represents how the last layer of CNNs are converted to a different network using Global Average Pooling followd by fully connected and a softmax layer.
2. The second image represents how feature maps are averaged to generate a final heatmap that is responsible for predicting the final target class.



Example: Consider an image of a dog classified as a "Labrador Retriever" by the model. CAMs highlight the regions in the image, such as the dog's face and body, that influenced the classification decision.

Image from dashboard: The image showcases a dog with its face prominently highlighted using class activation maps (CAMs). This visualization provides insights into the regions of the image that contribute most to the model's prediction, enhancing our understanding of the model's decision-making process.



4.2 Deep Feature Factorization (DFF):

Description: Deep Feature Factorization (DFF) is an innovative method for creating insightful visualizations about the internal representations of deep neural networks (DNNs). Unlike traditional explainability methods such as Grad-CAM, which focus on highlighting regions of an image that contribute to a specific category prediction, DFF delves deeper into the underlying concepts discovered by the model and how they are classified.

Motivations for Deep Feature Factorization: Traditional explainability methods like Grad-CAM provide valuable insights into where the model looks in an image to make predictions. However, they have limitations:

- Lack of granularity: They often fail to reveal the internal concepts the model identifies, such as individual object parts or features.
- Limited to target categories: They generate heatmaps specific to a single target category, making it difficult to understand the contributions of multiple objects in the image.
- Complexity in visualization: The visualization of multiple heatmaps for different categories can be overwhelming and inefficient for interpretation.

Deep Feature Factorization Workflow:

1. Reshaping Activations: The activations from the last convolutional layer of the pre-trained CNN are reshaped into 1D vectors to prepare them for factorization.
2. Non-negative Matrix Factorization (NMF): NMF is applied to these vectors to decompose them into distinct concepts. This process results in two matrices: W containing the feature representations of the detected concepts, and H containing how the pixels correspond with these concepts.
3. Concept Classification: If using the activations from the last layer, the remaining part of the network (e.g., fully connected layers) can be run on the concepts to classify each concept. This step provides insight into how the model interprets each concept.
4. Concept Visualization: To create a single visualization summarizing all concepts, each concept is assigned a unique color, and the intensity is modulated based on the heatmap. This approach ensures that the most important concept for each pixel is retained in the final visualization.

Mathematical Explanation:

- **Concept Activation Computation**: DFF computes the concept activations C by analyzing the neuron activations A^l in the network:
$$C = A^l \cdot W$$
where W is a weight matrix.
- **Factorization**: The concept activations C are factorized into interpretable concepts X using techniques such as Singular Value Decomposition (SVD) or Non-negative Matrix Factorization (NMF):
$$C = X \cdot Y$$
- **Visualization**: The extracted concepts X are visualized to provide insights into the learned representations of the model.

Connecting Concepts with Model Output: Using activations from the last convolutional layer simplifies connecting concepts with the model's output. For example, in ResNet50, running the fully connected layer on the concepts facilitates classification. For activations from earlier layers, alternative approaches like unpacking concepts to 2D tensors or analyzing concept heatmaps can be explored.

Example of Deep Feature Factorization: Consider an image classification task where DFF is applied to visualize concepts within an image of a dog. DFF may reveal concepts such as "dog face," "dog body," "dog fur," and "paws," providing granular insights into the model's interpretation of the image. Each concept is assigned a distinct color, and the intensity reflects its importance in the final classification.

Image from dashboard: The image displays a dog identified as a Golden Retriever by the ResNet model, with deep feature factorization (DFF) revealing multiple classifications. The model identifies the face region as a Golden Retriever while categorizing other parts as distinct entities such as hare and wood rabbit, showcasing the intricate interpretation of features learned by the neural network.

DFF image from the paper [Deep Feature Factorization]:(https://arxiv.org/pdf/1806.10206) The provided image showcases the effectiveness of Deep Feature Factorization (DFF) in identifying common elements across multiple images. Through DFF, features such as pyramids, animals, people, and monument parts are accurately matched and corresponded across different images, highlighting the robustness and interpretability of the method in understanding similarities within diverse visual datasets.

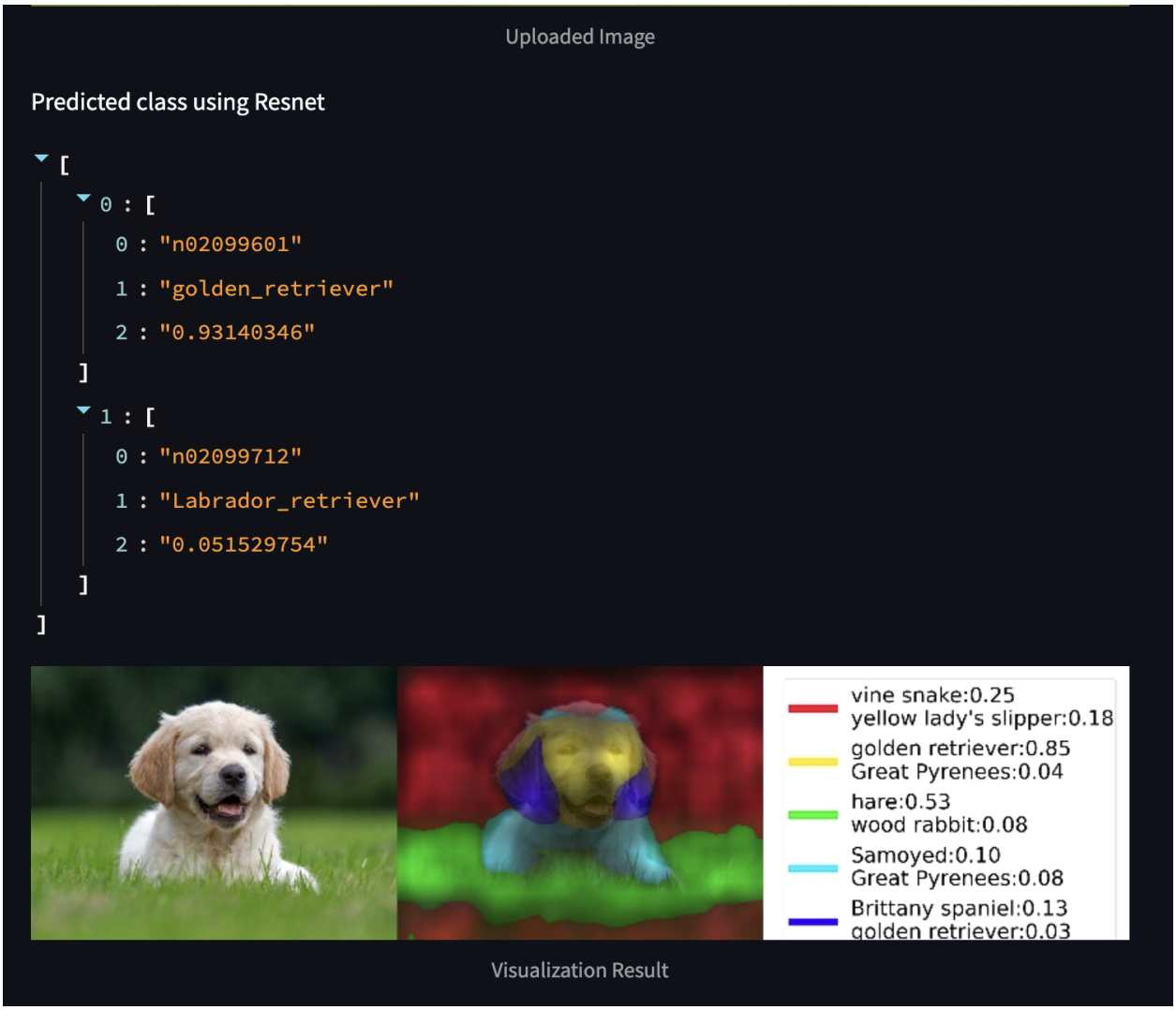


Figure 1: DFF Example1

Fig. 1: *What in this picture is the same as in the other pictures?* Our method, Deep Feature Factorization (DFF), allows us to see how a deep CNN trained for image classification would answer this question. (a) Pyramids, animals and people correspond across images. (b) Monument parts match with each other.

Figure 2: DFF illustration

Conclusion from DFF: Deep Feature Factorization offers a more detailed and nuanced understanding of DNNs compared to traditional explainability methods. By uncovering internal concepts and their classifications, DFF enhances transparency and interpretability, leading to more trust in deep learning models. The ability to visualize multiple concepts in a single image streamlines interpretation and facilitates deeper insights into model behavior.

4.3 LIME :

Description: LIME (Local Interpretable Model-agnostic Explanations) is an XAI technique that explains the predictions of any classifier in an interpretable and faithful manner, by approximating it locally with an interpretable model. LIME works by perturbing the input data and observing the corresponding changes in the output, allowing users to understand how individual features affect predictions.

How Lime Works: Here's an example using a cat to explain how Lime works:

1. **Select the Instance for Explanation**: Suppose we have an image that our model has classified as a cat. We want to understand why the model made this decision.
2. **Perturb the Data**: To explore how the model behaves near our specific instance (the original cat image), we create new images by making slight modifications. This could involve altering the brightness, adding noise, or slightly shifting parts of the image. Each of these new images is a perturbed version of the original.
3. **Predict Outcomes**: We then input these perturbed images into the original model to see what it predicts for each. Maybe some altered images are still classified as a cat, while others are not.
4. **Weight the New Samples**: Each perturbed image is given a weight based on its similarity to the original image. Images that are more similar to the original (less perturbed) have higher weights because the explanation focuses on the local environment around the original image.
5. **Fit an Interpretable Model**: Using the perturbed images and their predictions, we fit a simple, interpretable model, such as a linear model. This model uses the perturbed data (features from images

and their weights) to approximate how the original complex model behaves around the original instance. The goal is to predict the outcome (cat or not) based on these simplified inputs.

6. **Extract Explanation**: From the simple model, we can now extract an explanation. For instance, the model might indicate that the presence of pointy ears and whiskers heavily influences the prediction of a "cat." If the model assigns high positive coefficients to these features, it suggests they are crucial for the model's decision to classify the image as a cat.

Outcome:

We have chosen Tensorflow as the framework and mobilenetv2 as the pre-trained model. Keeping the image size at 224, and added a few normalization to the image. Then I used a shark as the image.

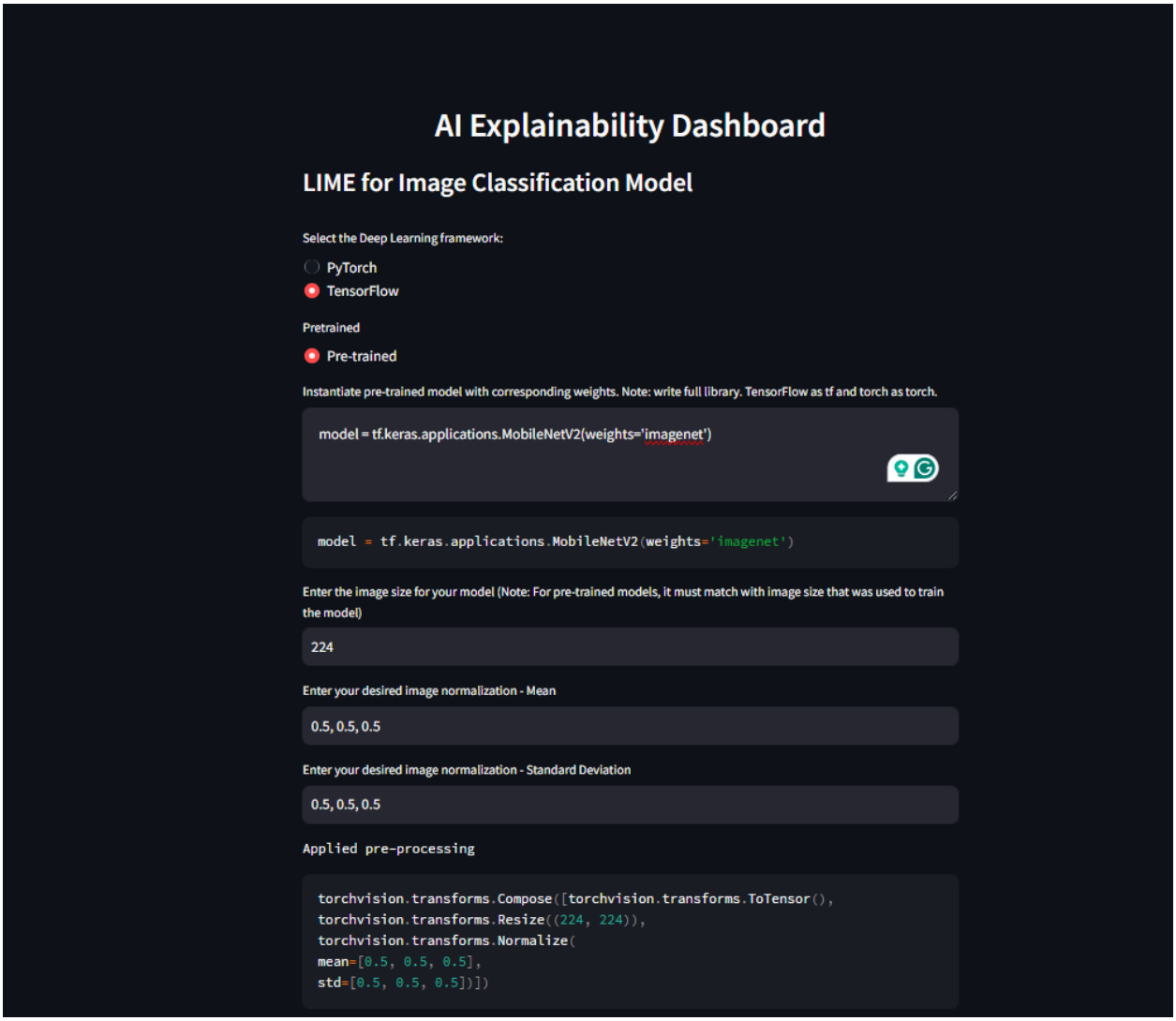


Figure 3: LIME DASHBOARD

Image from dashboard: The image above has formed the super-pixels or regions of interest based on the analysis.

The image above imposes this region of interest on the original image which is giving us more understanding. The final output can be seen on above image with a bar value varying from red to blue.

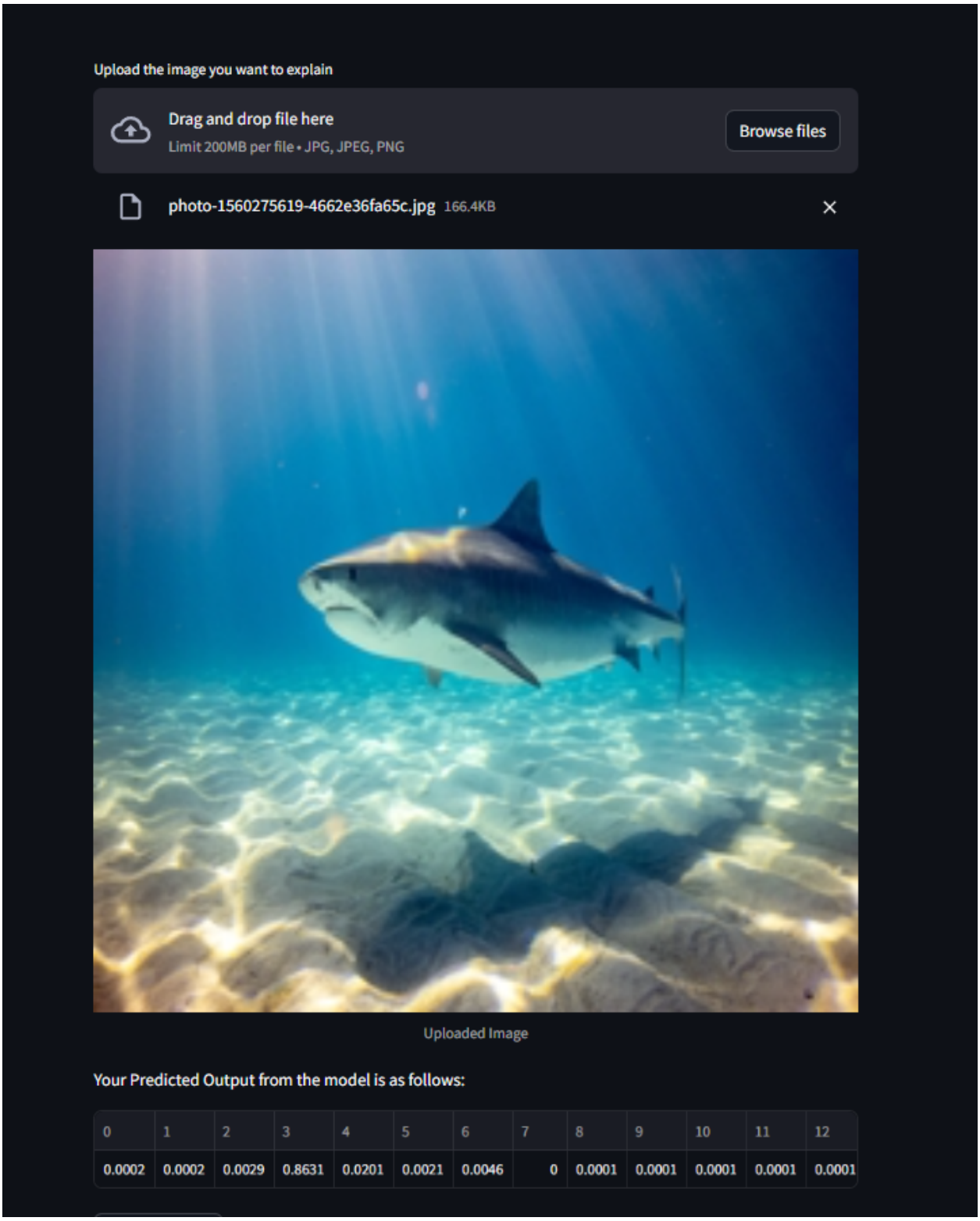


Figure 4: LIME DASHBOARD

Figure 5: LIME DASHBOARD

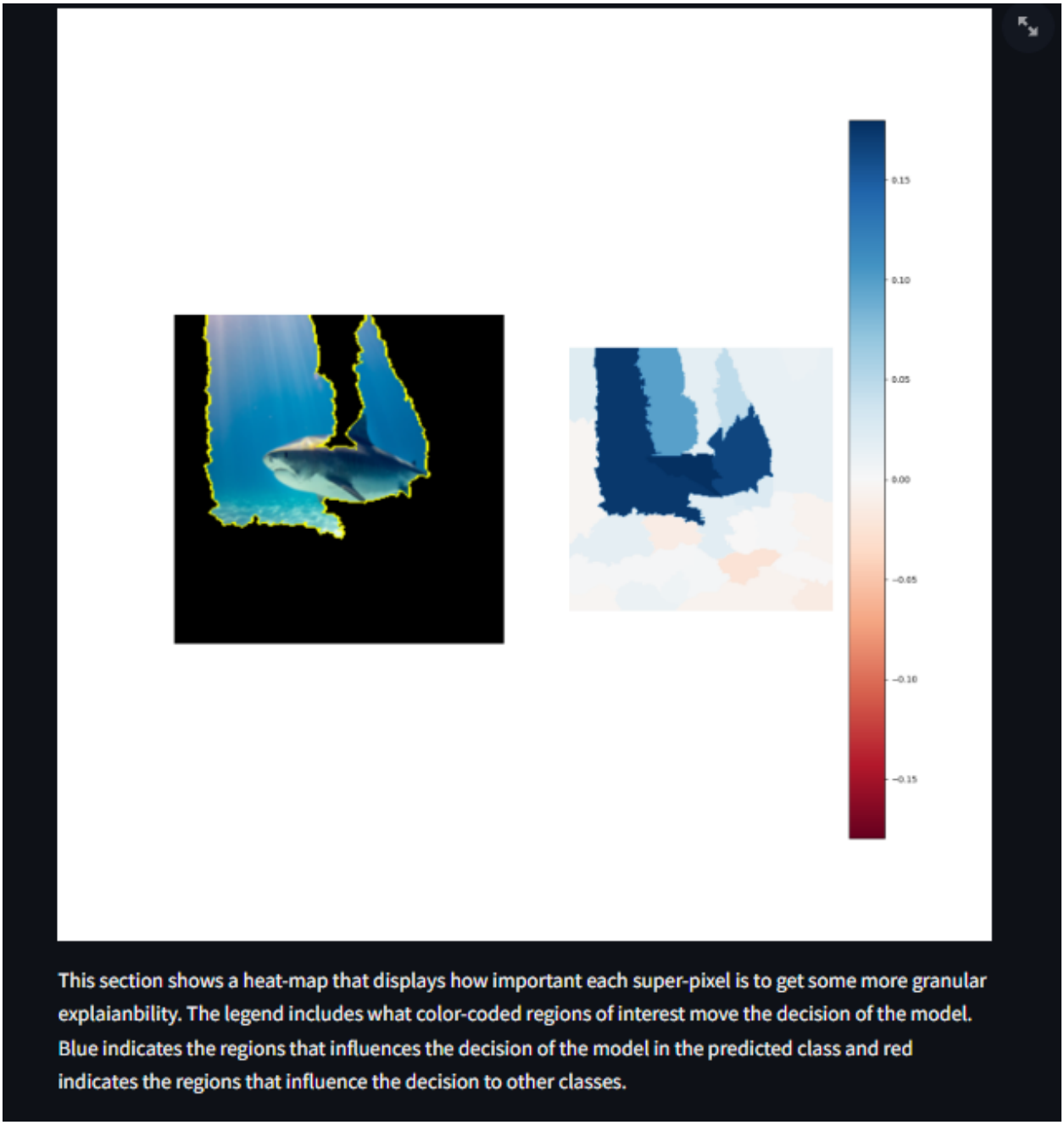


Figure 6: LIME DASHBOARD

4.4 SHAP:

Description: SHAP (SHapley Additive exPlanations) is an XAI technique derived from cooperative game theory, used to explain the output of any machine learning model. SHAP values provide a measure of the importance of each feature in a prediction, indicating how each contributes to pushing the model output from the baseline prediction to the current prediction. This method ensures fairness by distributing the "payout" (the prediction) fairly among the features based on their contribution.

- **Framework**: For SHAP implementation, I have used Pytorch and Tensorflow.
- **Explanation for Individual Predictions**: SHAP provides explanations for individual predictions, allowing users to understand how the model behaves at a specific instance.
- **Pixel-level Analysis**: Unlike LIME, SHAP considers every pixel as a feature and provides the shapely value for each.
- **Compatibility**: SHAP is compatible with any model, offering flexibility to work with different machine learning frameworks and types of models, whether they are custom-built, pre-trained, or a combination of both.

SHAP values explain the influence of each feature on the specific prediction by showing the contribution of each feature to the prediction.

How SHAP Works:

Using a similar example with an image classified as a cat:

1. **Select the Instance for Explanation** We start with the same image of a cat that our model has classified as containing a cat. We aim to uncover the reasoning behind this classification.
2. **Compute SHAP Values** SHAP values are computed for each feature of the input image. This involves using permutations of the input features to evaluate the change in the prediction output when a feature is present versus when it is absent.
3. **Aggregate Contributions** For image data, this might involve considering how specific regions of the image (like pixels or segments) contribute to the model's prediction. This aggregation can be visualized as a heatmap where areas that significantly influence the prediction are highlighted.
4. **Visualize the Explanation** The SHAP values can be visualized to show the impact of each region of the image. For example, regions with high SHAP values that positively affect the prediction (like the presence of whiskers or ears) will be highlighted, indicating their importance in the classification decision.
5. **Interpret the Results** The visualization helps us understand which features (or image regions) are most influential. If SHAP values show high positive values for areas capturing the cat's ears and whiskers, it suggests these features are key determinants in recognizing the image as a cat.

Images from Dashboard: For the SHAP image classification, We have used 2 pre-trained models ResNet50 and VGG16. We have chosen the ResNet50 as the pre-trained model and cat as our dataset image. The output, it's segmented into pixels, here all the pixels will be considered for the overall prediction of the model.

The below bar values represent the Shapely values, originates from cooperative game theory and is used to fairly distribute the total payoff to players depending on their contribution to the overall game. In the context of machine learning, Shapley values are used to determine the contribution of each feature in a model's prediction.

Figure 7: SHAP DASHBOARD

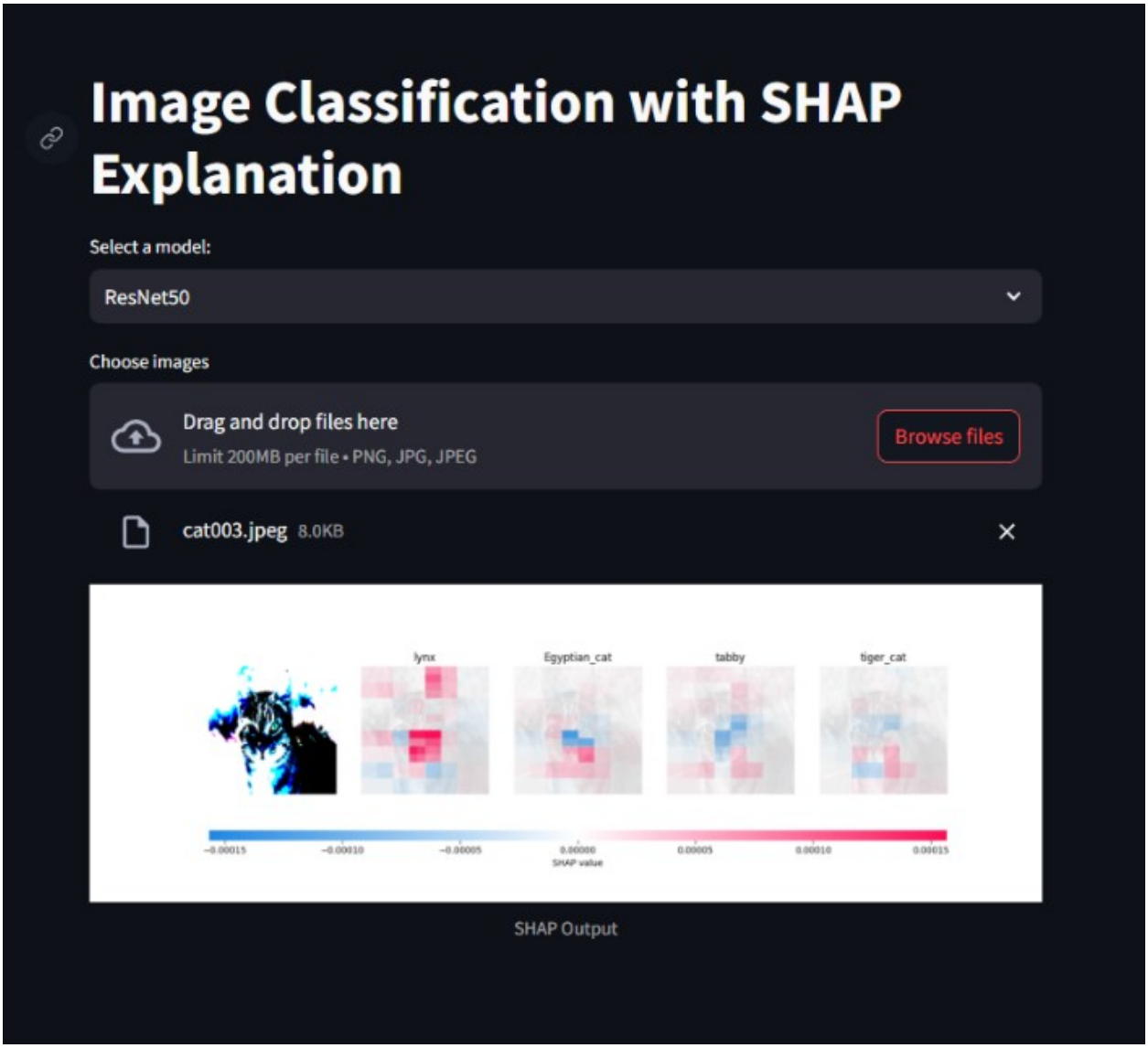
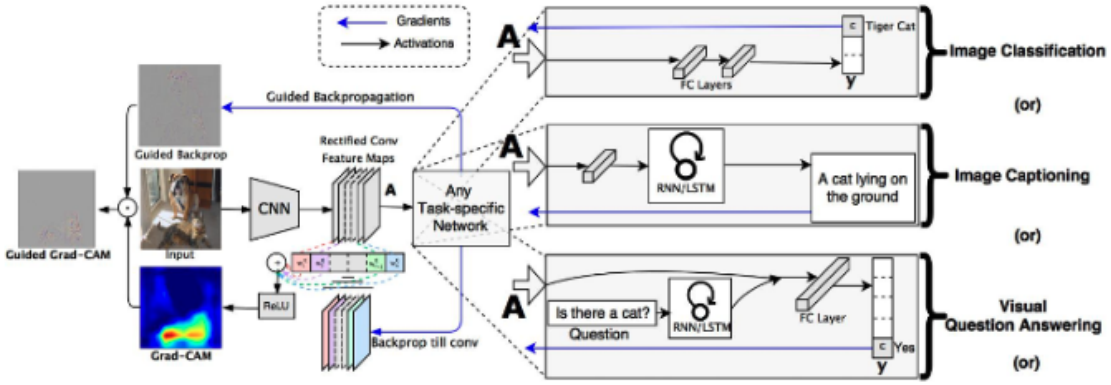


Figure 8: SHAP DASHBOARD

4.5 GRAD CAM:

Description: Grad-CAM leverages the activations of the last convolutional layer in the network. These activations capture high-level features learned by the network(Pretrained - Resnet50 in our case) during training, which are crucial for making predictions. Grad-CAM relies on the ability of a neural network to learn and extract features from input images.



Above architecture shows how the Gradcam works.

Below are the steps on how the GradCAM algorithm works:

Forward Pass: Feed the input image through the CNN to obtain the feature maps of the last convolutional layer. Let A_k be the activations of the last convolutional layer for the input image, where k is the index of the feature map.

Gradient Calculation: Compute the gradients of the target class score Y_c with respect to the activations A_k of the last convolutional layer. (This makes the main difference between CAM and GradCAM)

Global Average Pooling (GAP): Compute the importance of each feature map by taking the global average of the gradients.

$$\alpha_k = \frac{1}{Z} \sum_i \sum_j \frac{\partial Y_c}{\partial A_{ij}^k}$$

Figure 9: GAP

Weighted Combination: Weight the feature maps by their rank (according to weights). Then apply ReLU to the weighted combination and upsample it to match the dimensions of the input image. This produces a weighted combination of the feature maps, emphasizing the regions that are most relevant for predicting the target class.

Visualization: Overlay the heatmap onto the input image to visualize the important regions. Typically, the heatmap is scaled and blended with the input image to produce a visually interpretable result. The heatmap indicates which parts of the input image are most influential in the network's decision-making process for the target class.

Figure 10: Input Image for GradCam

Results from Dashboard: **Input Image**:

Output Image:

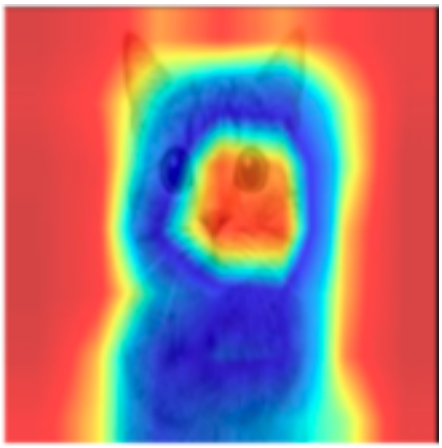


Figure 11: Output Image from GradCam

As we can identify from the pictures that to identify for the model as Tabby (a breed in card), from the gradcam is from the features in Face.

4.6 Occlusion Sensitivity:

Description: The use of Occlusion Sensitivity is to understand the importance of different parts of an image in the model's decision-making process.

Results from Dashboard:

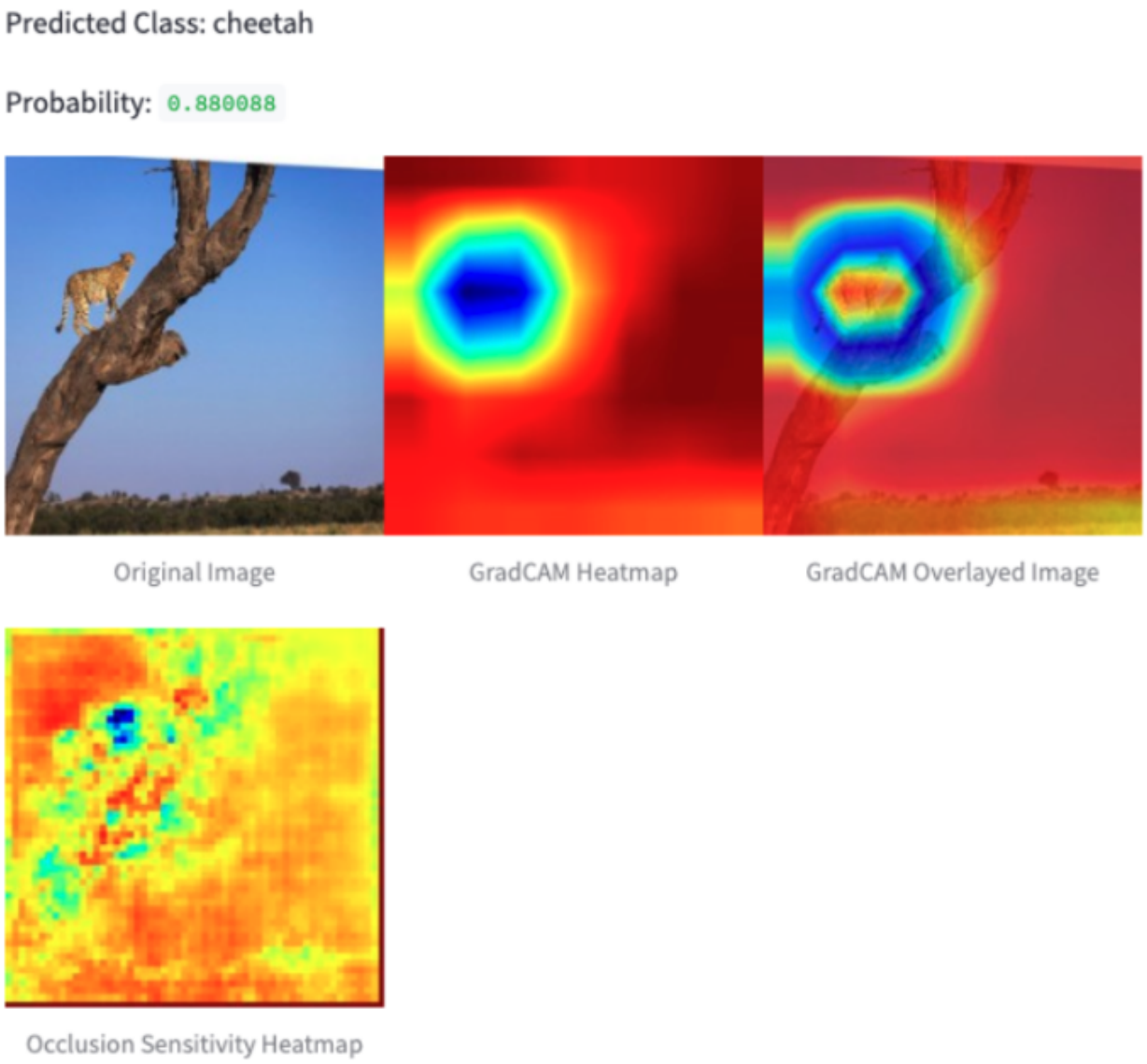


Figure 12: Occlusion Sensitivity Results

How Occlusion Sensitivity Works:

1. The occlusion value is used to define the value that replaces occluded regions of the image, allowing the algorithm to assess the importance of different parts of the image for the model's predictions.
2. We will occlude different parts of the image by sliding the occlusion window across the image with a given stride. (This means that part of the image is effectively hidden or covered.)
3. The occluded image is then fed through the model, and the model's prediction is observed.
4. For each patch in the image, the model's prediction probability for the target class is calculated after occluding that patch. The probability represents the model's confidence in the presence of the target class when that patch is occluded. The higher the probability, the more important that patch is for the model's prediction of the target class.
5. These probabilities are stored in a heatmap where each pixel's value represents the model's confidence when the corresponding patch is occluded. The heatmap is then normalized to values between 0 and 255 for visualization.
6. By comparing the model's predictions with and without occlusion, the algorithm can understand how important each part of the image is for the model's decision-making process.

5. Conclusion:

In this report, we discussed several explainability techniques used in the field of deep learning. From Class Activation Maps (CAM) to Occlusion Sensitivity, each technique offers unique insights into how deep neural networks make decisions and provides valuable interpretability for model predictions.

Class Activation Maps (CAM) allow us to visualize the regions of an image that contribute most to a particular class prediction, providing valuable insights into the model's attention mechanism. Grad-CAM further refines this approach by incorporating gradient information to produce more localized heatmaps.

Deep Feature Factorization (DFF) takes interpretability a step further by extracting meaningful concepts from the learned representations of neural networks. By decomposing the feature space into interpretable concepts, DFF enhances transparency and enables visualization of abstract concepts present in the feature space.

Local Interpretable Model-agnostic Explanations (LIME) and SHapley Additive exPlanations (SHAP) offer model-agnostic approaches to explain individual predictions. LIME approximates the complex model locally with an interpretable model, while SHAP provides a measure of feature importance based on cooperative game theory, offering insights into the contribution of each feature to the model's prediction.

Finally, Occlusion Sensitivity helps us understand the importance of different parts of an image in the model's decision-making process by occluding regions of the image and observing the effect on the model's predictions.

By employing these explainability techniques, researchers and practitioners can gain deeper insights into deep learning models, enhance transparency, and build more trustworthy AI systems for real-world applications.

6. References:

1. Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In Proceedings of the IEEE International Conference on Computer Vision (ICCV) (pp. 618–626). Venice, Italy.
2. Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning Deep Features for Discriminative Localization. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2921–2929). Las Vegas, NV, USA.

3. Petsiuk, V., Das, A., & Saenko, K. (2018). Rise: Randomized Input Sampling for Explanation of Black-box Models. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 346–363). Munich, Germany.

4. Lundberg, S. M., & Lee, S. I. (2017). A Unified Approach to Interpreting Model Predictions. In Advances in Neural Information Processing Systems (NIPS) (pp. 4765–4774). Long Beach, CA, USA.

5. Zeiler, M. D., & Fergus, R. (2014). Visualizing and Understanding Convolutional Networks. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 818–833). Zurich, Switzerland.

6. Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). "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 (KDD) (pp. 1135–1144). San Francisco, CA, USA.