Exploring Explainability in Deep neural networks

Abstract:

The VGG model is one of the popular Convolution Neural network models for recognizing and classifying the images. However, its complex design, filled with many layers that process the images, makes it hard for people to understand and how it comes up with its results. So, this is the common issue with deep learning where we can't easily see what's happening inside these "black box models". The main focus of our project is to make the VGG model's decision-making clear and understandable. We mainly aim to make it easier for everyone to trust and use this advanced AI, especially in important areas like healthcare imaging and self-driving cars, where knowing how the AI makes decisions is very important. We plan to integrate explainability methods with a VGG model tailored for image classification tasks, making its decision-making process transparent. In addition, we compare the effectiveness of various explainability techniques in demystifying the layers and features used by the VGG model in making predictions. Furthermore, we assess the impact of enhanced explainability on the interpretability and trustworthiness of the VGG model among end-users.

Our approach involves the application and potential adaptation of both model-specific and model-agnostic explainability techniques to a pre-trained VGG model. Techniques such as Local Interpretable Model-agnostic Explanations (LIME), Grad-CAM (Gradient-weighted Class Activation Mapping), Saliency MAP, Integrated Gradients (IG) and SHAP (Shapley Additive Explanations) will be explored for their compatibility and effectiveness with the VGG architecture. The project will evaluate these methods' ability to visually and quantitatively explain which features and patterns within the input images are most influential in the model's classification decisions.

Introduction:

Deep Learning has excellent development in various domains, which is revolutionising the way machines understand and interact with the world. There are many architectures driving this transformation, the VGG model has excellent improvement in the field of image recognition and classification because of its composition of various convolution layers. This VGG model can capture complex patterns and features in visual data, which often surpasses human performance in accuracy. Yet, this sophistication introduces a layer of complexity that obscures the model's decision-making processes from human understanding. The convolutional operations and transformations within these networks are often referred to as "black box" models due to their lack of transparency, presenting a significant barrier to trust and usability in critical applications such as healthcare imaging and autonomous vehicles.

The nature of DL models hides their broader acceptance, especially in sensitive areas where understanding the vision behind an AI'a decision is as important as the decision itself. For example, in health care imaging, a radiologist must understand why a model classifies a scan as indicative of a disease to trust and act upon its recommendation. Likewise in autonomous driving, understanding the models's decision making process is crucial for safety verification and gaining regulatory and public approval. The need for model transparency is not just a technical necessity but also a societal imperative.

Our project aims to describe the layers of convolution neural networks. Particularly the VGG model, and render their decision-making transparent. Finally we focus on explainability methods that can show details of inner workings of these DL models. Explainability in AI is an interdisciplinary effort to make the outcomes of DL models interpretable to humans, effectively bridging a gap between complex models predictions and actionable human understanding.

The main objectives of the project are: First, we aim to integrate state-of-the art explainability methods into the VGG model to describe its image classification tasks. Second, we plan to conduct a comparative study on the effectiveness of various techniques. These techniques are Local Interpretable Model-agnostic Explanations (LIME), Grad-CAM (Gradient-weighted Class Activation Mapping), Saliency MAP, Integrated Gradients (IG) and SHAP (Shapley Additive Explanations), in demonstration the features and patterns the VGG model that is used for predictions. Third, we assess the impact of these explanabilty enhancements on the interpretability and trustworthiness of the VGG model among end-users.

In our methodology, we tend to apply and adapt both model-specific and model 's explanability techniques to VGG trained models. The CIFAR-10 dataset, comprising 60,000 colour images across ten distinct classes, serves as view point for our evaluations, offering a diverse and comprehensive collection of images for a robust assessment of the VGG model's performance and the effectiveness of applied explanaabilty techniques.

By accomplishing our objectives, we anticipate contributing significantly to the field of interpretable AI. We aim to not only enhance the trustworthiness of the DL model but also show a way for their deployment in areas where the need for transparency are high. Our project aspires for development of AI systems that are not only intelligent but also interpretable, accountable and trustworthy.

Objectives

- To integrate explainability methods with a VGG model tailored for image classification tasks, making its decision-making process transparent.
- To compare the effectiveness of various explainability techniques in demystifying the layers and features used by the VGG model in making predictions

VGG Model:

We have constructed a VGG13 model for our dataset. Below are the details:

```
----------------------------------------
VGG13(
  (conv_layers): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU()
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU()
    (14): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (fc_layers): Sequential(
    (0): Linear(in_features=4096, out_features=1024, bias=True)
    (1): ReLU()
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=1024, out_features=1024, bias=True)
    (4): ReLU()
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=1024, out_features=10, bias=True)
  )
)
----------------------------------------------------------------
      Layer (type)               Output Shape         Param #
================================================================
          Conv2d-1         [-1, 64, 64, 64]           1,792
            ReLU-2         [-1, 64, 64, 64]               0
          Conv2d-3         [-1, 64, 64, 64]          36,928
            ReLU-4         [-1, 64, 64, 64]               0
       MaxPool2d-5         [-1, 64, 32, 32]               0
          Conv2d-6        [-1, 128, 32, 32]          73,856
            ReLU-7        [-1, 128, 32, 32]               0
          Conv2d-8        [-1, 128, 32, 32]         147,584
            ReLU-9        [-1, 128, 32, 32]               0
      MaxPool2d-10        [-1, 128, 16, 16]               0
         Conv2d-11        [-1, 256, 16, 16]         295,168
           ReLU-12        [-1, 256, 16, 16]               0
         Conv2d-13        [-1, 256, 16, 16]         590,080
           ReLU-14        [-1, 256, 16, 16]               0
      MaxPool2d-15          [-1, 256, 8, 8]               0
         Linear-16                [-1, 1024]       4,195,328
```

ReLU-17	[-1, 1024]	0
Dropout-18	[-1, 1024]	0
Linear-19	[-1, 1024]	1,049,600
ReLU-20	[-1, 1024]	0
Dropout-21	[-1, 1024]	0
Linear-22	[-1, 10]	10,250

==========

Total params: 6,400,586
Trainable params: 6,400,586
Non-trainable params: 0

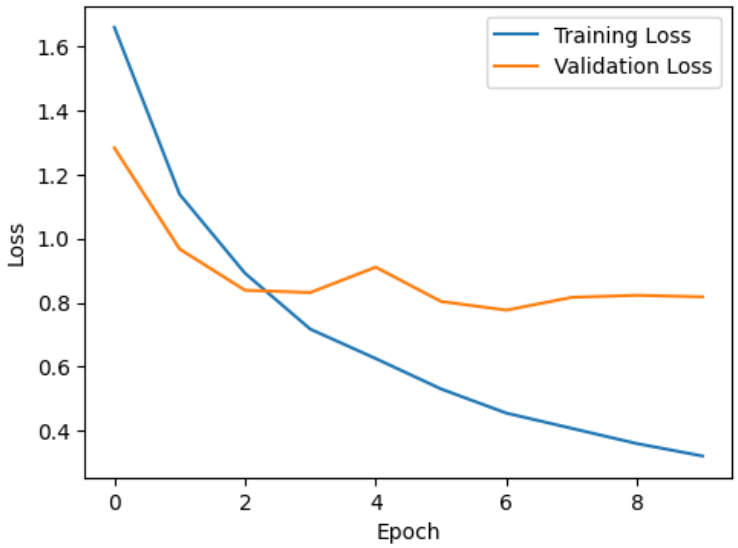
Input size (MB): 0.05
Forward/backward pass size (MB): 14.92
Params size (MB): 24.42
Estimated Total Size (MB): 39.39

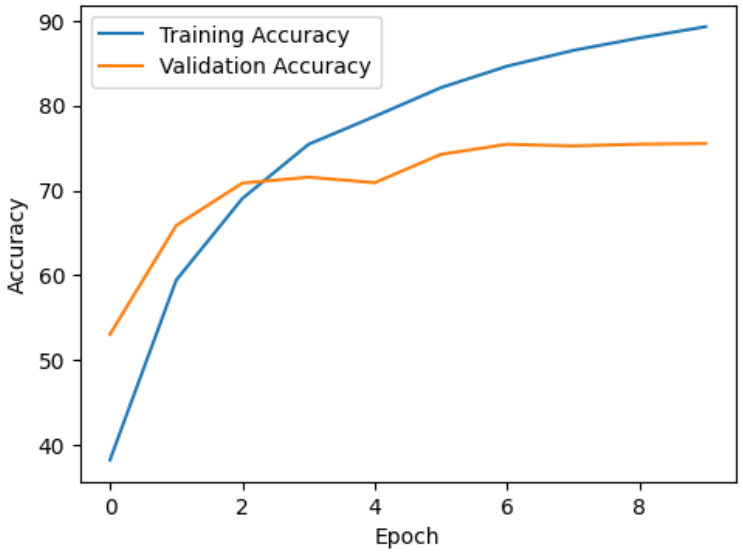
Learning rate: 0.001
Loss function: Cross Entropy loss
Optimizer: Adam
Dropout probability: 0.5
Number of epochs: 10

Performance:

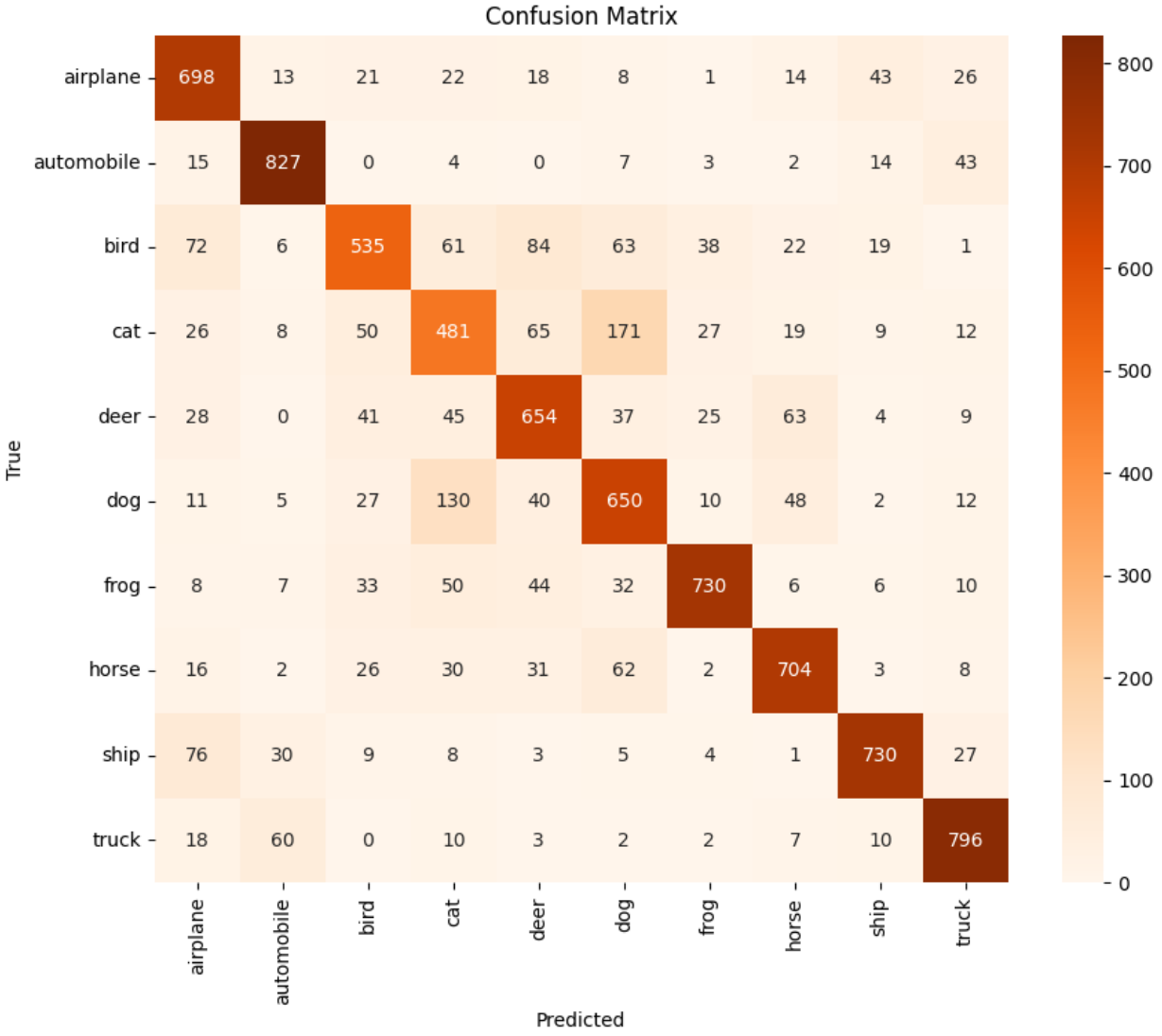
Train accuracy: 89.29%
Train loss: 0.32
Validation accuracy: 75.52%
Validation loss: 0.81
Test accuracy: 75%
Test Loss: 0.80

Evaluation Metrics:
Precision: 0.76
Recall: 0.76
F1 Score: 0.76





Confusion Matrix:



Base Model + L2 Regularization:

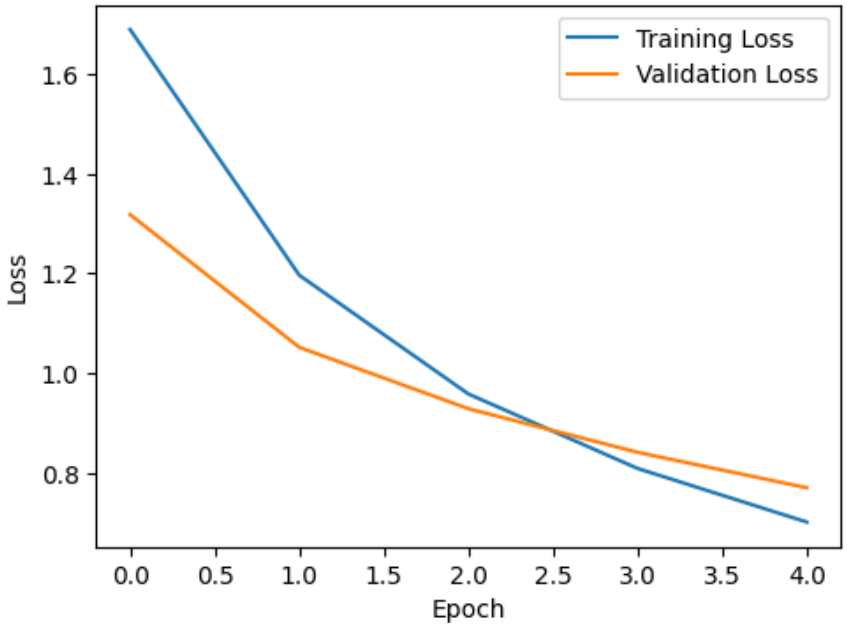
Hyper Parametrers:

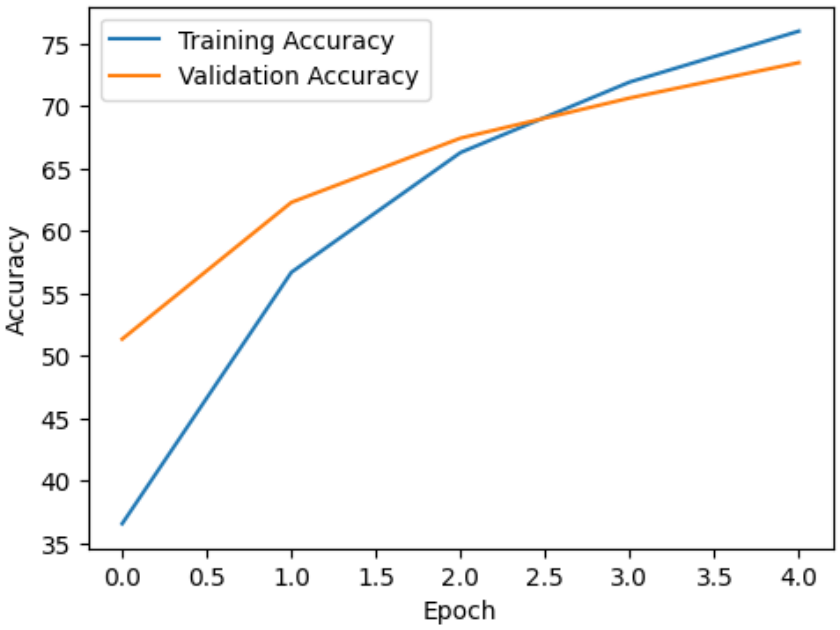
Learning rate: 0.001
Loss function: Cross Entropy loss
Optimizer: Adam
Weight decay: 1e-5
Dropout probability: 0.5
Number of epochs: 10

Performance:

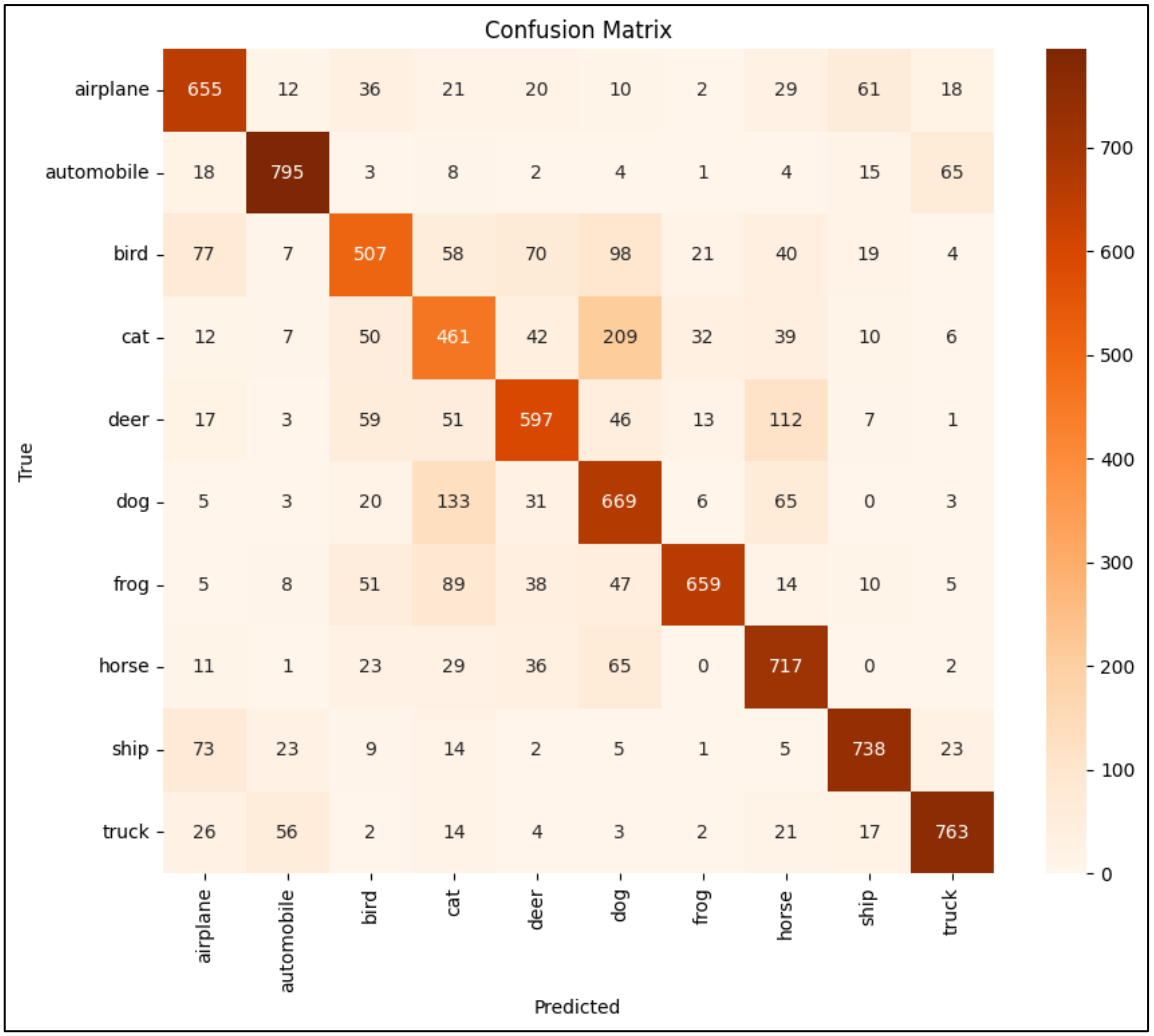
Train accuracy: 76.0%
Train loss: 0.70
Validation accuracy: 73.48%
Validation loss: 0.77
Test accuracy: 72%
Test Loss: 0.79

Evaluation Metrics:
Precision: 0.75
Recall: 0.75
F1 Score: 0.75





Confusion Matrix:

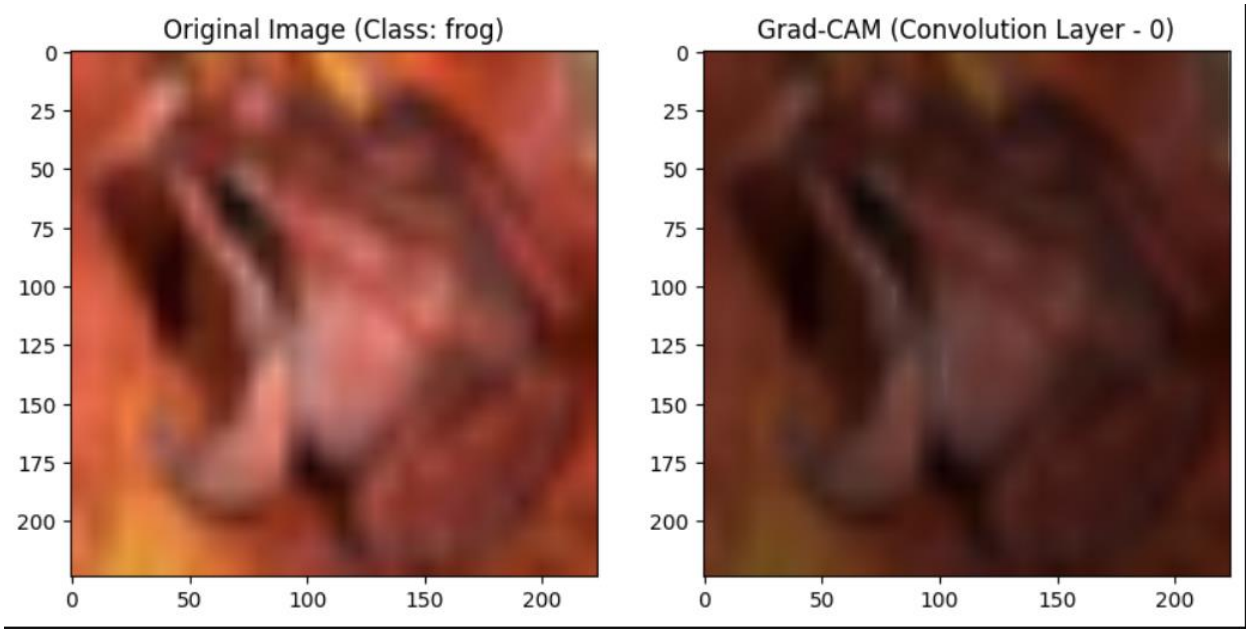

Confusion Matrix
True
Predicted
airplane
automobile
bird
cat
deer
dog
frog
horse
ship
truck
655 12 36 21 20 10 2 29 61 18
18 795 3 8 2 4 1 4 15 65
77 7 507 58 70 98 21 40 19 4
12 7 50 461 42 209 32 39 10 6
17 3 59 51 597 46 13 112 7 1
5 3 20 133 31 669 6 65 0 3
5 8 51 89 38 47 659 14 10 5
11 1 23 29 36 65 0 717 0 2
73 23 9 14 2 5 1 5 738 23
26 56 2 14 4 3 2 21 17 763
700
600
500
400
300
200
100
0

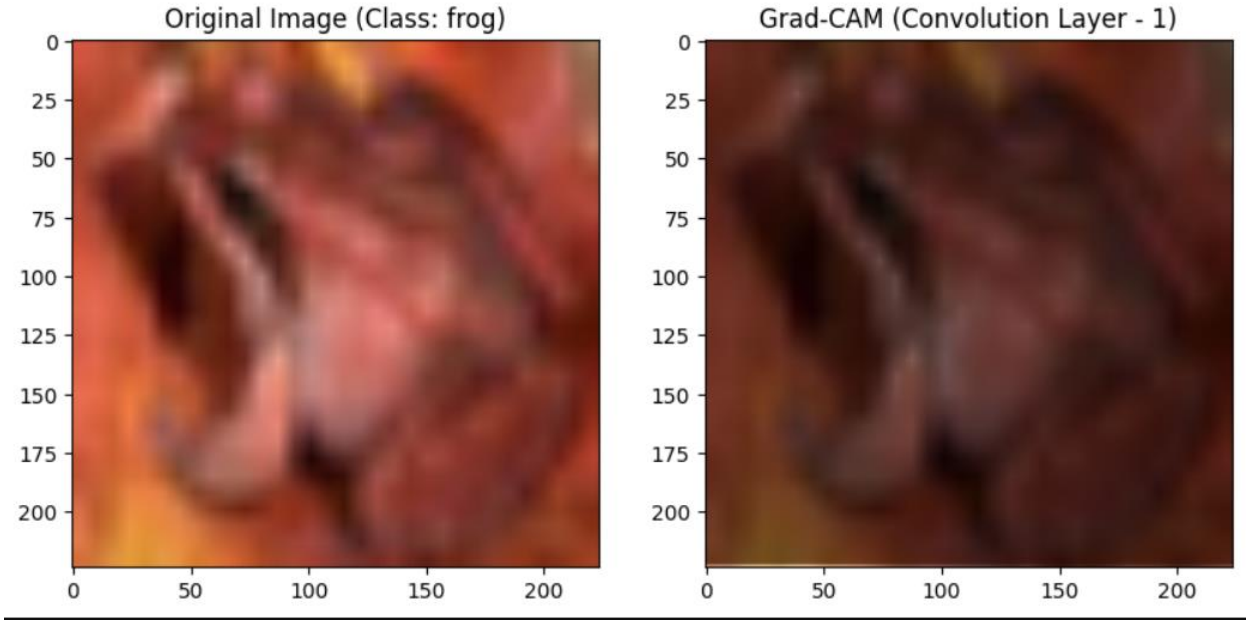
Exploring Grad-CAM technique for model explainability:

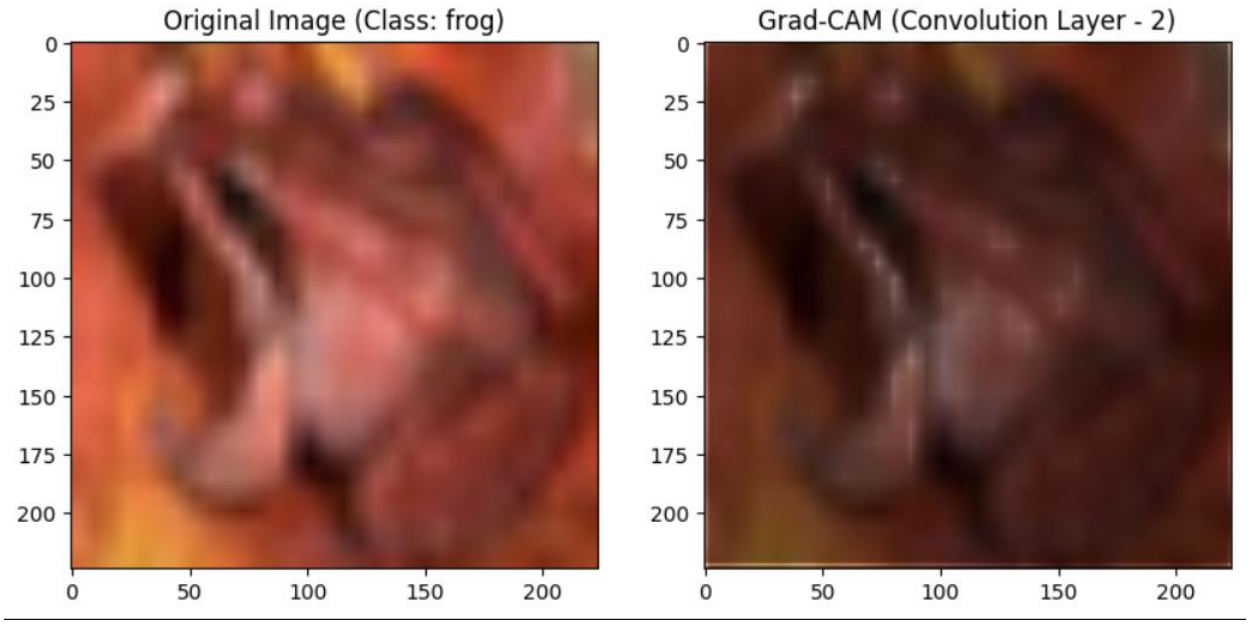
Grad-CAM was chosen for its simplicity and effectiveness in localizing the important regions in the input image. We utilized the gradients flowing into each layer of the trained VGG13 model to create coarse heat maps. This allowed us to visualize which parts of an image led to a particular output, offering a window into the model's layered processing.

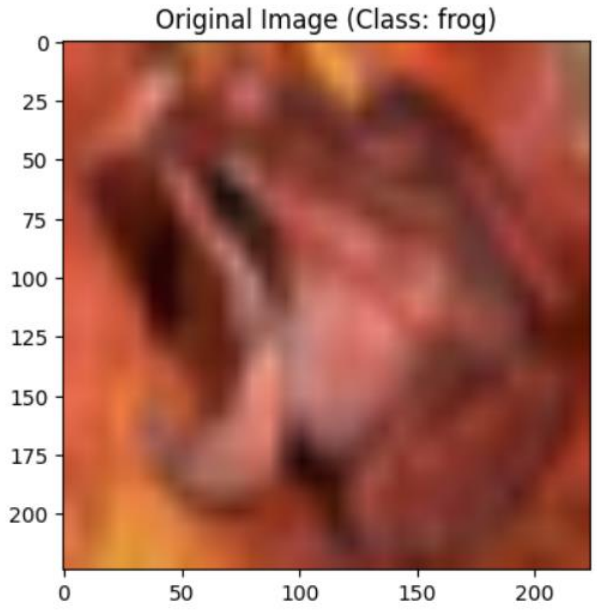
Implementation and Results:

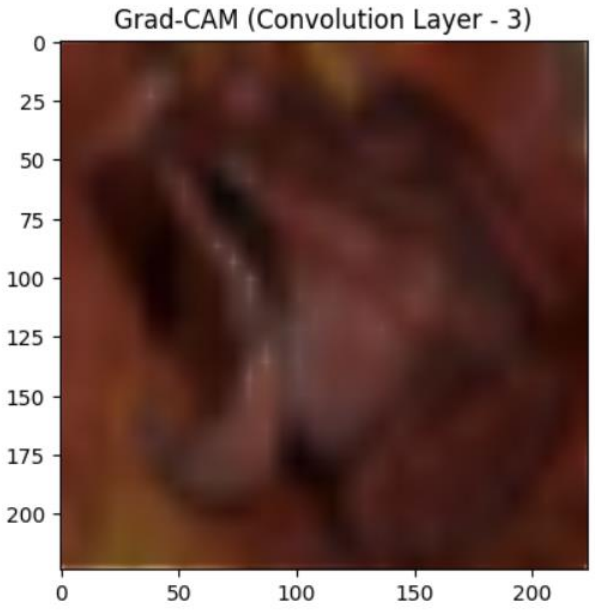
To make our project work, we had to tweak how we prepare images so they fit what our VGG13 model needs. We also adjusted the Grad-CAM method to work smoothly with our model's setup. The cool thing we got out of this was a set of heatmaps, which are like special maps that show us where the model is looking in an image. These maps tell a story about how the model starts by noticing simple things like edges and textures and then moves on to recognize more complicated stuff in the deeper parts of the model.

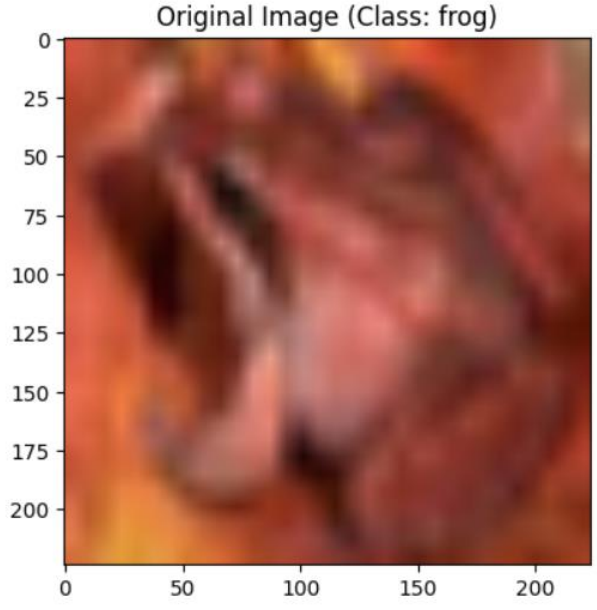

Original Image (Class: frog)
Grad-CAM (Convolution Layer - 0)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

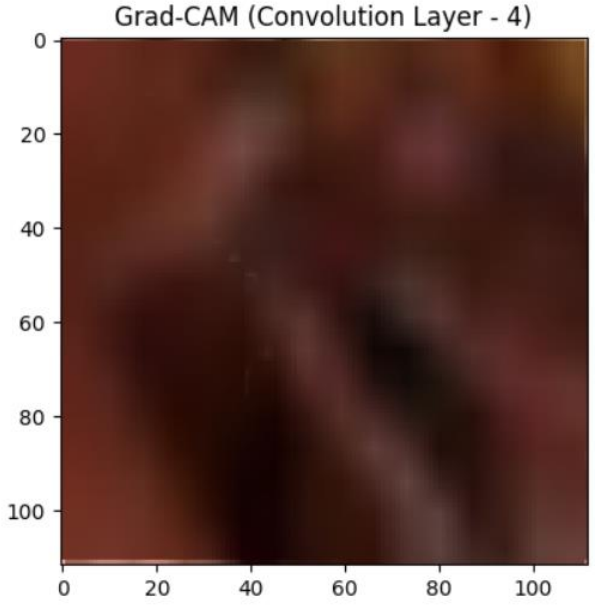

Original Image (Class: frog)
Grad-CAM (Convolution Layer - 1)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

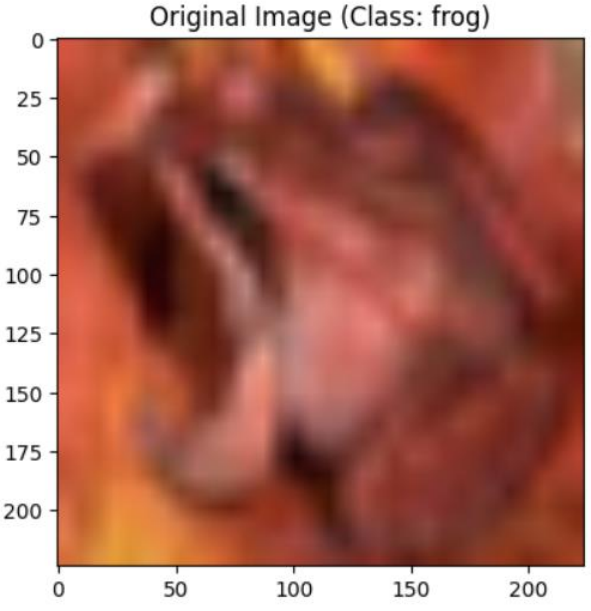

Original Image (Class: frog)
Grad-CAM (Convolution Layer - 2)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

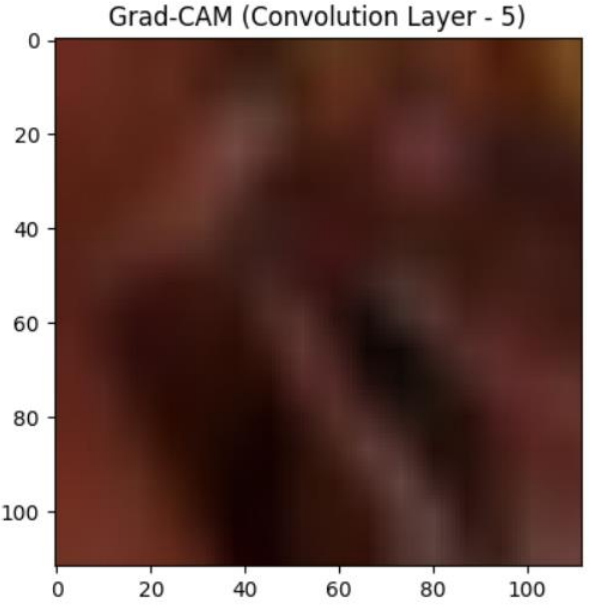

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

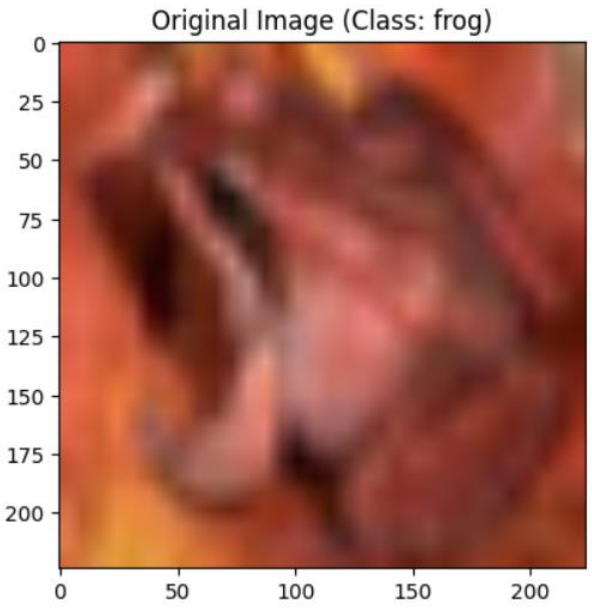

Grad-CAM (Convolution Layer - 3)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

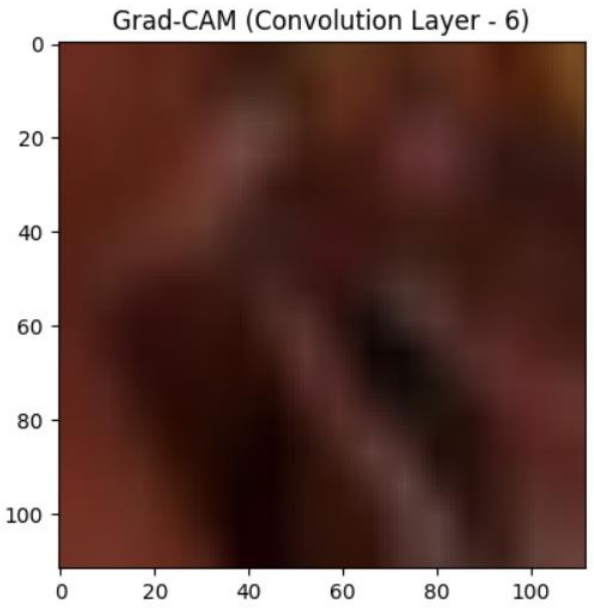

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

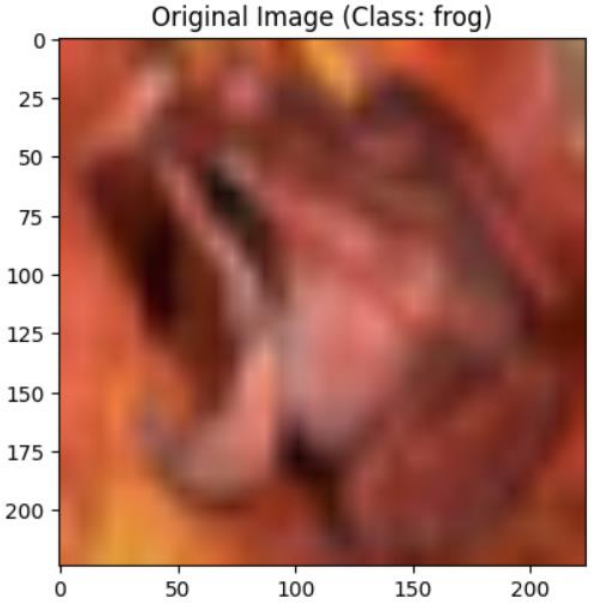

Grad-CAM (Convolution Layer - 4)
0
20
40
60
80
100
0
20
40
60
80
100

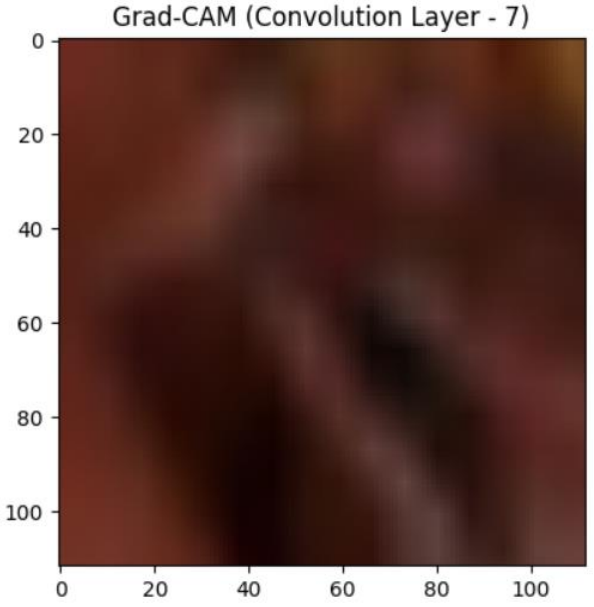

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

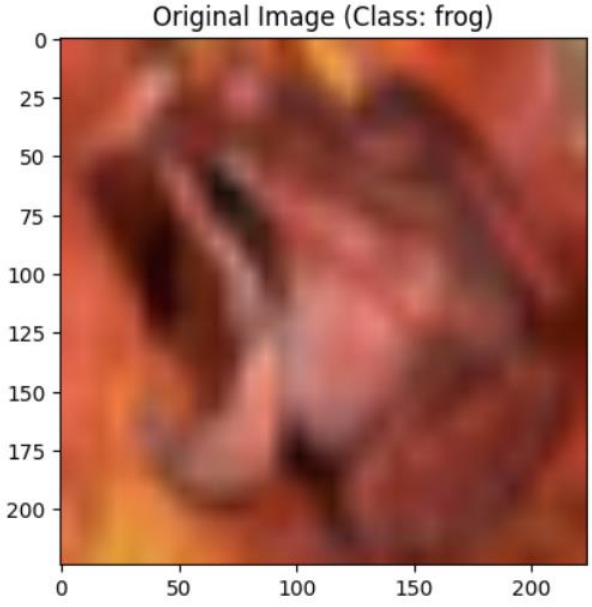

Grad-CAM (Convolution Layer - 5)
0
20
40
60
80
100
0
20
40
60
80
100

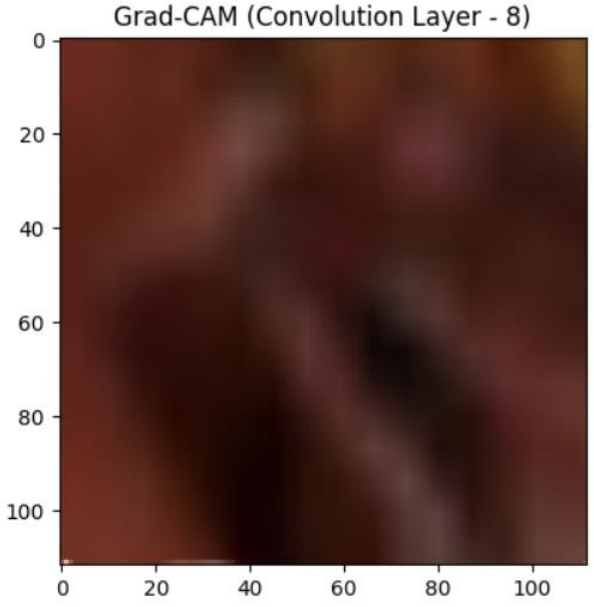

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

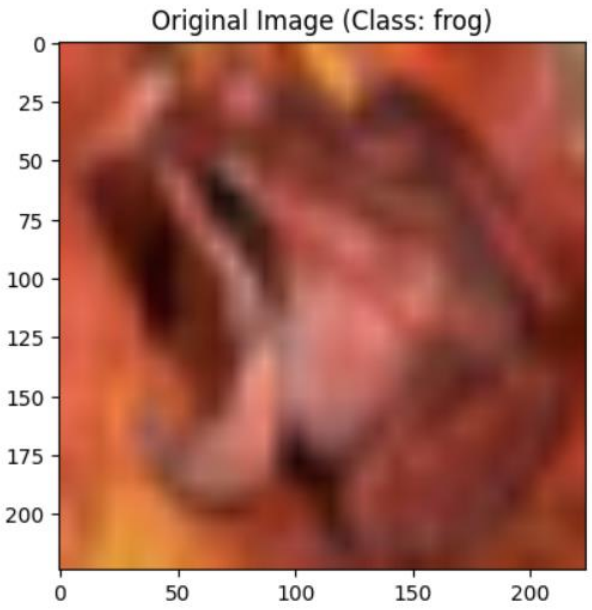

Grad-CAM (Convolution Layer - 6)
0
20
40
60
80
100
0
20
40
60
80
100

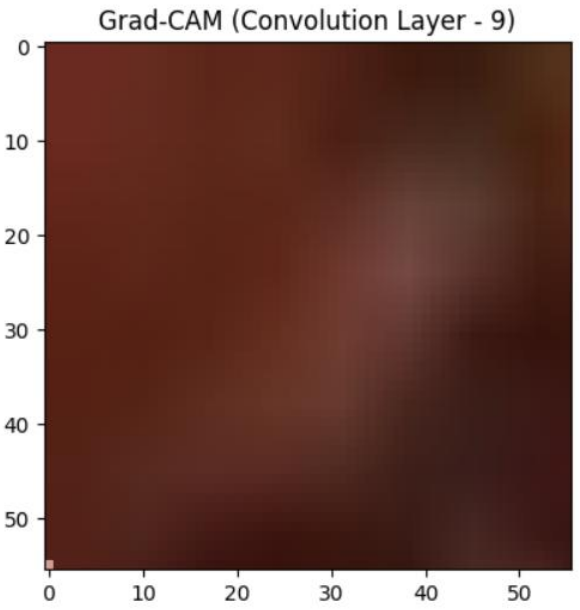

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

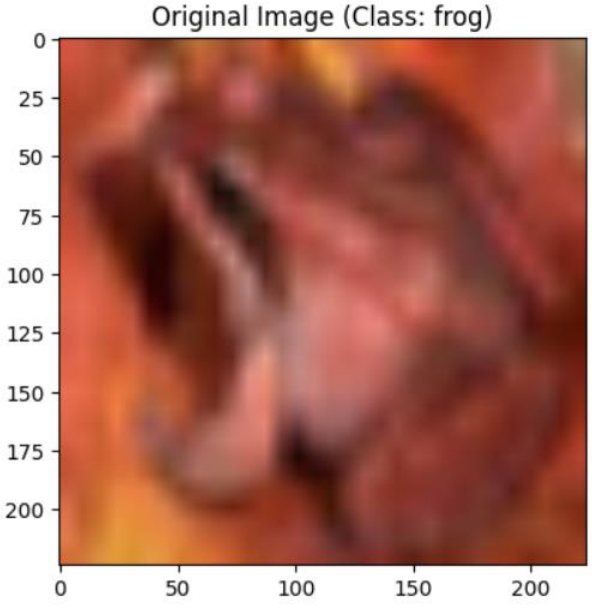

Grad-CAM (Convolution Layer - 7)
0
20
40
60
80
100
0
20
40
60
80
100

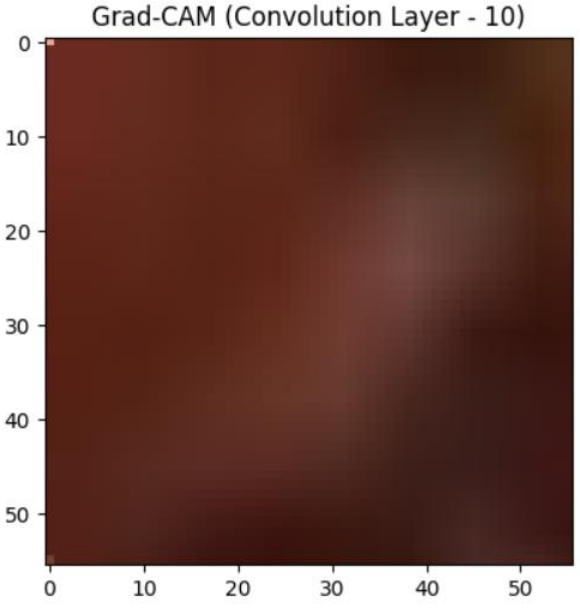

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

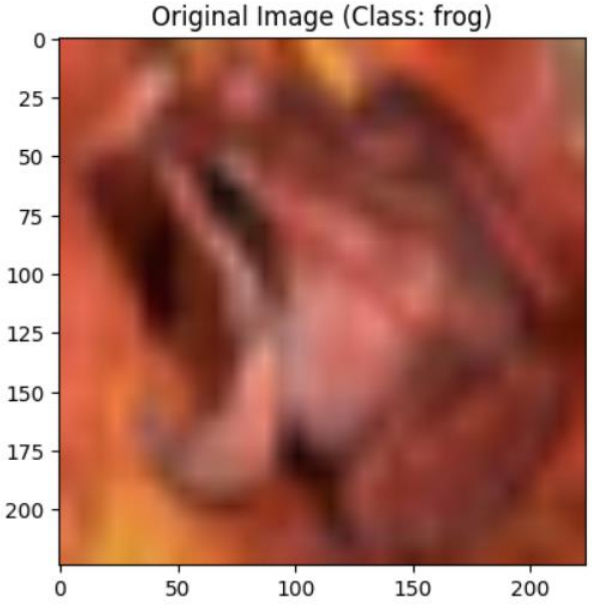

Grad-CAM (Convolution Layer - 8)
0
20
40
60
80
100
0
20
40
60
80
100

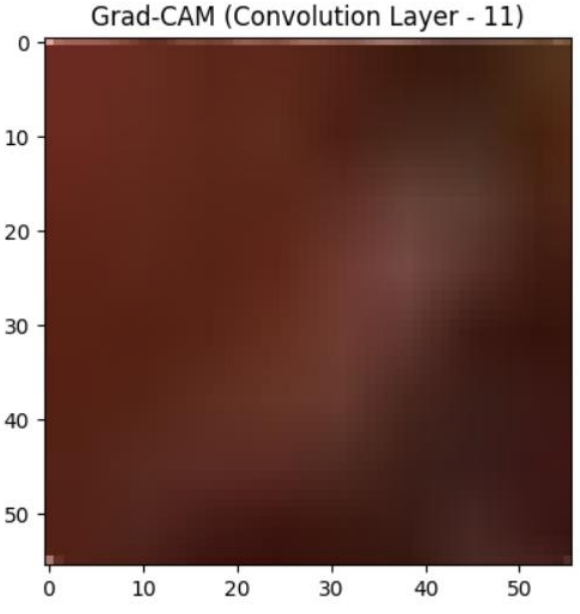

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

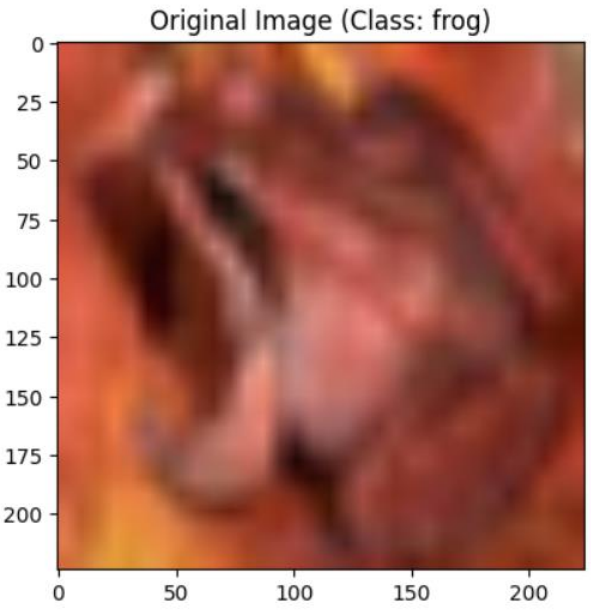

Grad-CAM (Convolution Layer - 9)
0
10
20
30
40
50
0
10
20
30
40
50

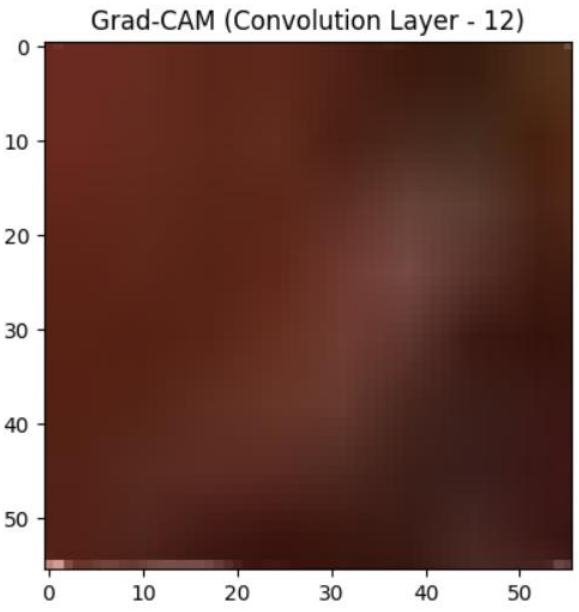

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

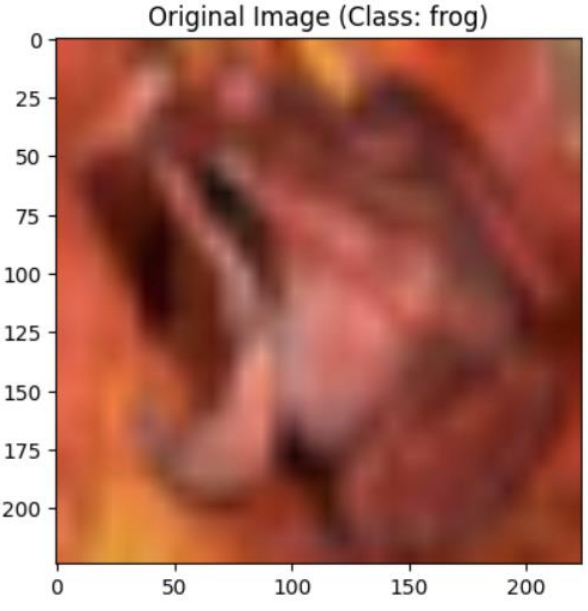

Grad-CAM (Convolution Layer - 10)
0
10
20
30
40
50
0
10
20
30
40
50

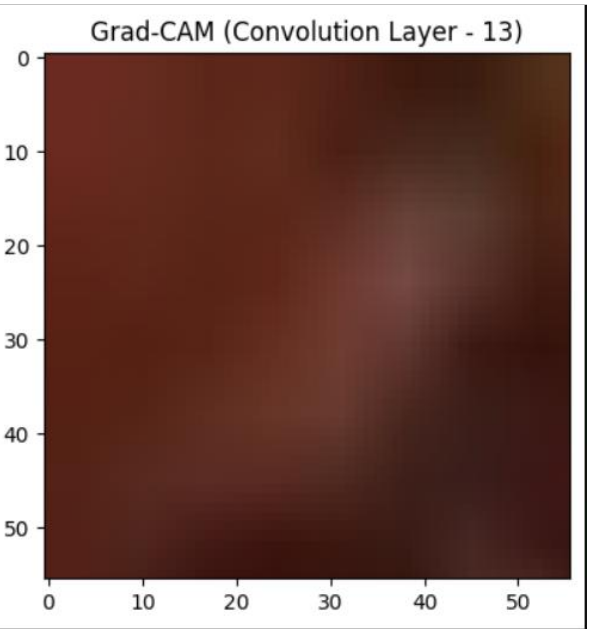

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200

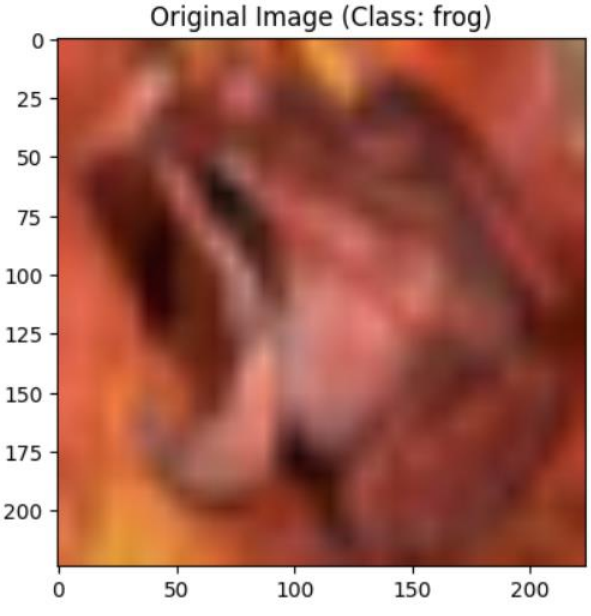

Grad-CAM (Convolution Layer - 11)
0
10
20
30
40
50
0
10
20
30
40
50

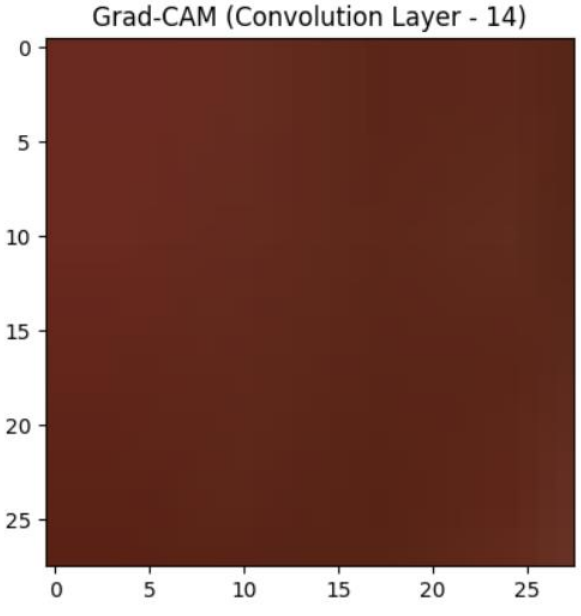

Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200


Grad-CAM (Convolution Layer - 12)
0
10
20
30
40
50
0
10
20
30
40
50


Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200


Grad-CAM (Convolution Layer - 13)
0
10
20
30
40
50
0
10
20
30
40
50


Original Image (Class: frog)
0
25
50
75
100
125
150
175
200
0
50
100
150
200


Grad-CAM (Convolution Layer - 14)
0
5
10
15
20
25
0
5
10
15
20
25

Interpreting the GradCAM Heatmap:

The above image shows the example of a frog. The GradCam heatmaps show the regions of the image that are most influential for the VGG13 model when classifying the image as “frog”. We ran the GradCam heatmap for all 16 convolution layers. Below is the layer-by-layer interpretation:

- **Layers 0 and 1:** In these initial convolution layers, we observe that the heatmap appears quite blurred indicating that the model is focusing on broad features and general patterns across the image rather than specific details.

- **Layers 2 and 3:** As we progress to these layers, we observe that the focus areas start to become more defined although it is still broad. This indicates that the network begins to focus on more specific patterns that are relevant to the classification.

- **Layers 4 to 6:** In these intermediate layers, we can observe that the highlighted areas start to become more concentrated and focused. We see that the Grad-CAM heatmap is beginning to reveal the network's attention to certain regions of the image which are likely important for making the classification decision.

- **Layers 7 and 8:** When we look at the further layers, we observe that the highlighted regions are even more focused, and we see that the model is paying close attention to certain parts of the image. Here, we observe that the model is likely identifying the upper body features of the frog in this example like its head, eyes, and its gills.

- **Layers 9 to 13:** In these layers, we observe that the highlighted regions are most intensely focused on the body texture of the frog. We can observe that the frog is magnified to a maximum extent covering the entire heatmap region and the model is likely identifying and learning textures on certain body parts of the frog.

- **Layer 14:** As we proceed to the final convolution layer, we observe that the heatmap is completely dark. Therefore, the model may have abstracted the features to such a high level that Grad-CAM cannot easily visualize the importance of these features in the spatial domain of the original image. It's also possible that the critical features for classification are so abstract at this point that they are not easily represented in a 2D heatmap.

Limitations of GradCam Explainability:

As we observe the GradCam heatmaps, we need to consider certain limitations in interpreting GradCam. They are:

- **Diminishing Spatial Relevance**: Deep layers of a CNN tend to extract highly abstract features that may represent complex structures pertinent to the classification task, like object parts or entire objects, and may not be directly related to specific spatial locations in the image.
- **Visualization Limitations**: Grad-CAM and similar visualization techniques sometimes struggle to highlight meaningful patterns in the deepest layers because the

activations have been abstracted away from the pixel space.

- **Contextual Information**: Deep layers incorporate more contextual information which is essential for distinguishing among classes but is not always visually interpretable.
- **Model Interpretation**: If the final layers' heatmaps are not informative, it could suggest that the model is relying on features that are not easily interpretable or that Grad-CAM is not the right tool for visualizing these particular model decisions.

Conclusion:

Using Grad-CAM with our VGG13 model has given us exciting outcomes in making it easier to understand how deep learning works. The special heatmaps we created show just how cleverly the model can pick out and focus on important parts of images. This approach is a big step forward in helping us all get a clearer picture of how AI makes its choices.

Exploring Saliency Maps Technique for model explainability:

Saliency maps are visualization tools used in the field of machine learning particularly in the context of deep neural networks to interpret the behavior of deep learning models. Like GradCam, Saliency Maps are also used to interpret the behavior of convolution neural networks particularly in tasks like image classification. However, they both differ in their approach and the type of insights they provide.

Unlike GradCam which focuses on the importance of different feature maps within convolution layers, saliency maps focus on the individual pixels in the image. Saliency maps focus on identifying which pixels in the input image most affect the output prediction by computing the gradient of the output for the input image.

Saliency maps produce finer detail at the pixel level compared to gradcam, but they can be very noisy, especially in deeper or very complex models.

Computing Saliency Maps:

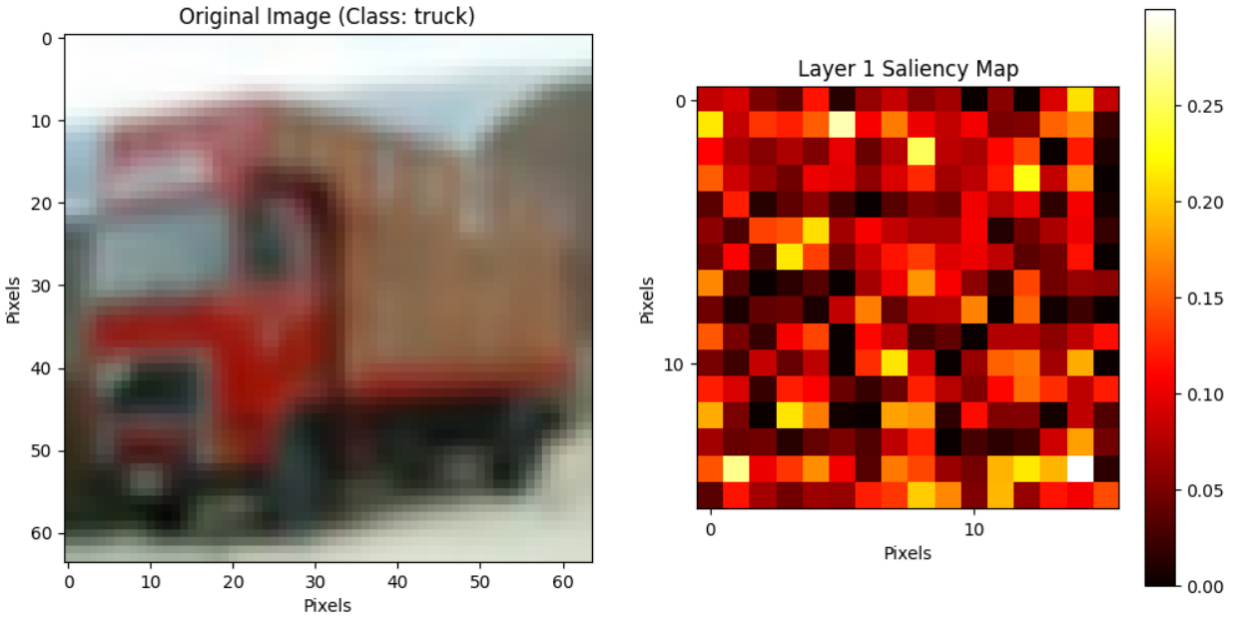
Here is a brief overview of how saliency maps are computed:

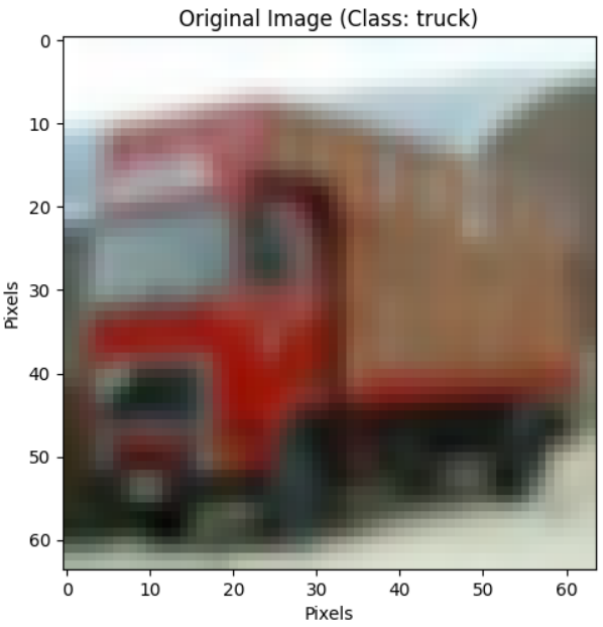
- **Gradient Computation**: The basic idea is to compute the gradient of the output for the input data. This gradient measures how changes in each input pixel (for image data) or feature influence the model’s prediction.
- **Backpropagation**: Using backpropagation, these gradients are computed for a specific output class (for classification tasks) or the predicted value (for regression tasks).
- **Highlighting Influential Features**: The absolute values of these gradients are then used to create a map where higher values indicate more influence on the model's output. This map is overlaid on the input image or data to show the "hot spots" or important regions.

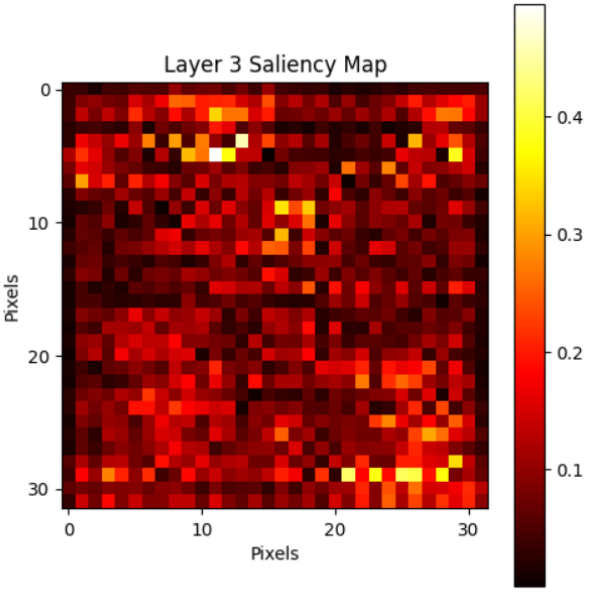
Implementation and Results:

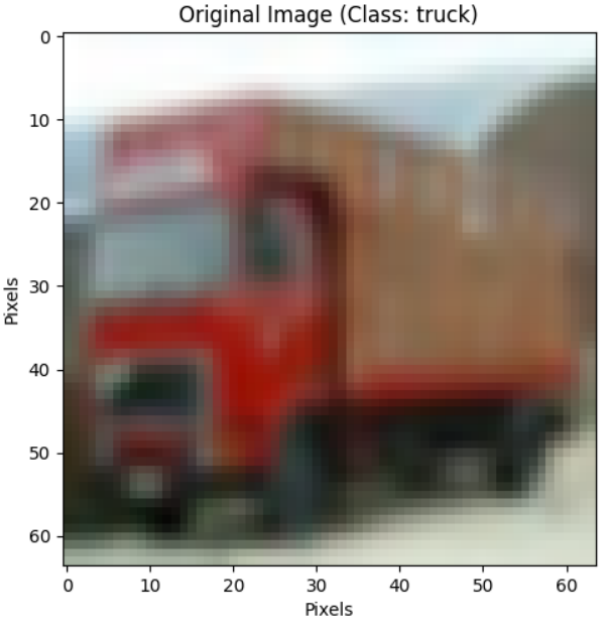
To implement saliency maps in our project, we adjusted the input image processing to align with the specific requirements of our neural network model. We employed hooks to track activations and gradients across the model's layers as it processed images. Most importantly, we created saliency maps, which are visual representations showing the areas of the image most influential to the model's predictions. These maps effectively illustrate how the model focuses on different image features, from basic details like edges in early layers to more complex patterns in deeper layers, revealing the model's internal decision-making process.

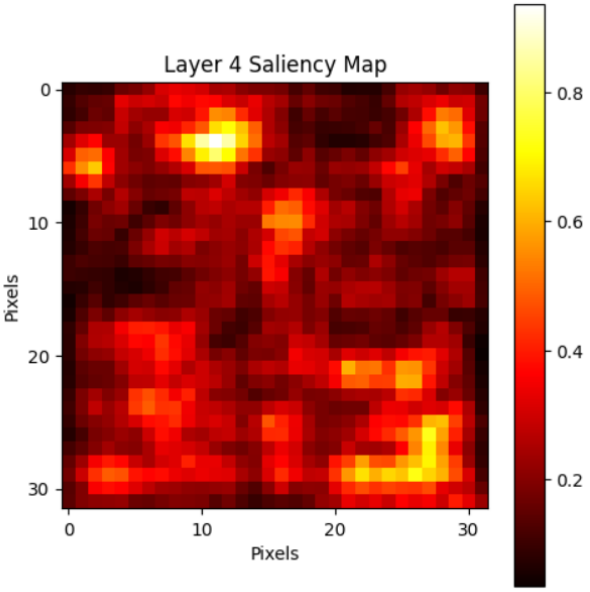
Below are the saliency map visualizations for different layers in the model:

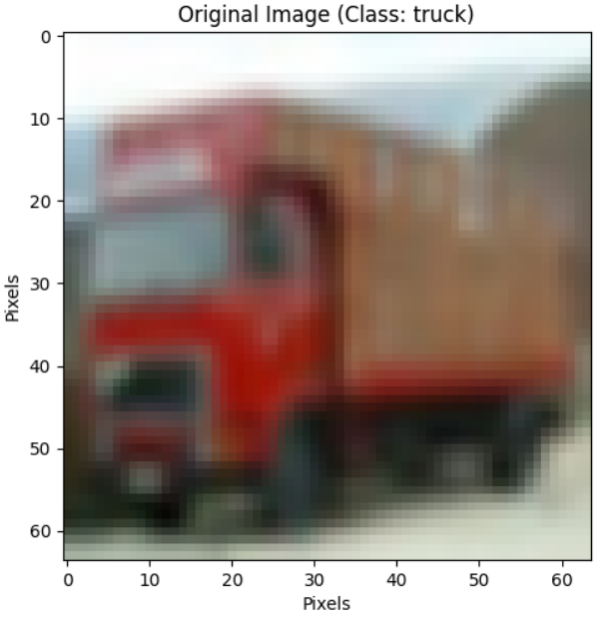


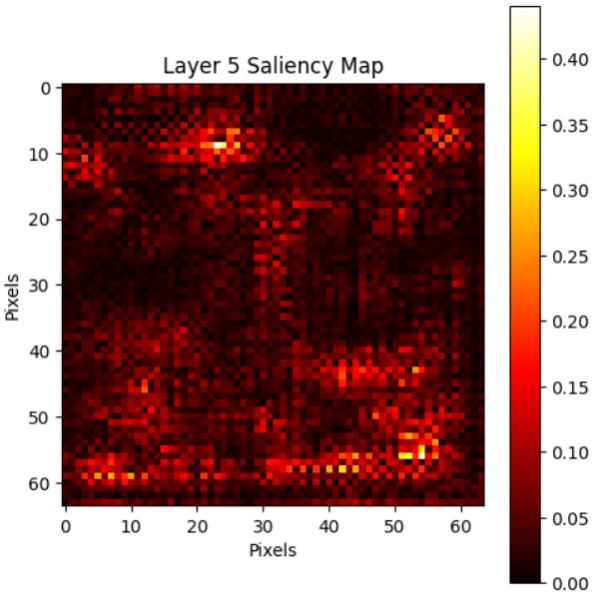

Original Image (Class: truck)
Pixels
Pixels

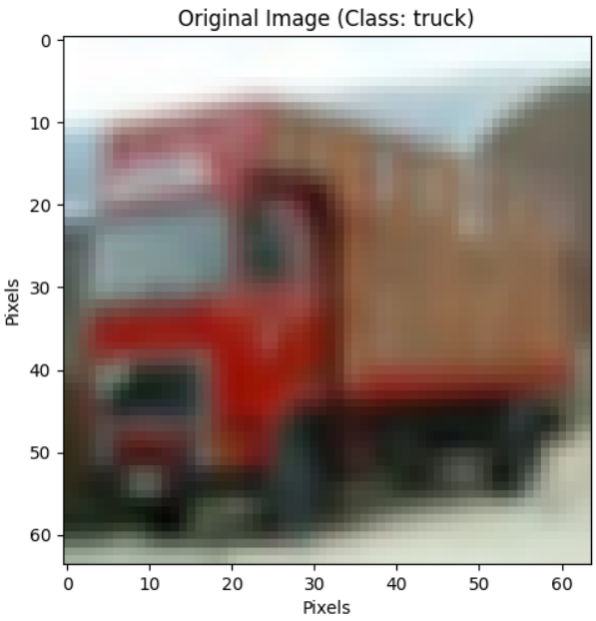

Layer 3 Saliency Map
Pixels
Pixels

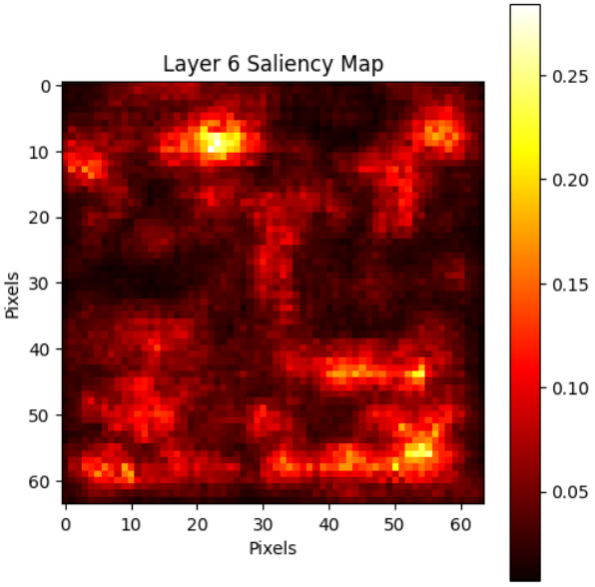

Original Image (Class: truck)
Pixels
Pixels

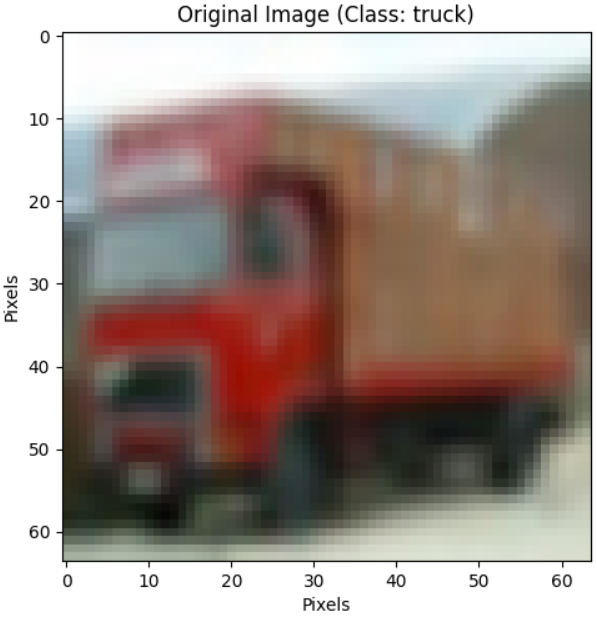

Layer 4 Saliency Map
Pixels
Pixels

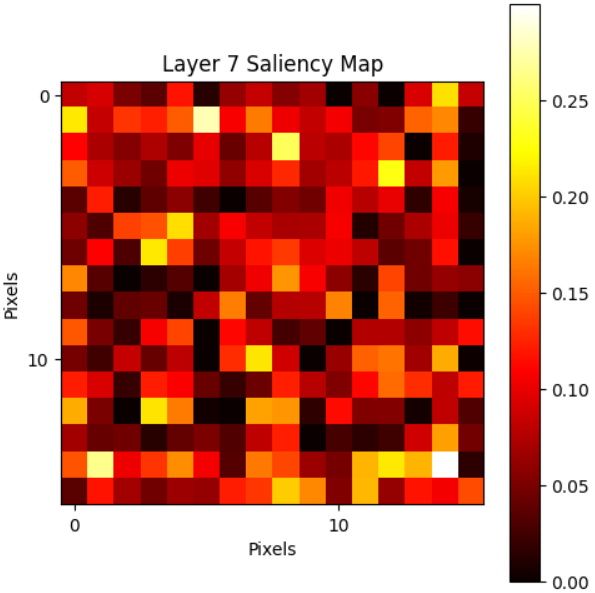

Original Image (Class: truck)
Pixels
Pixels

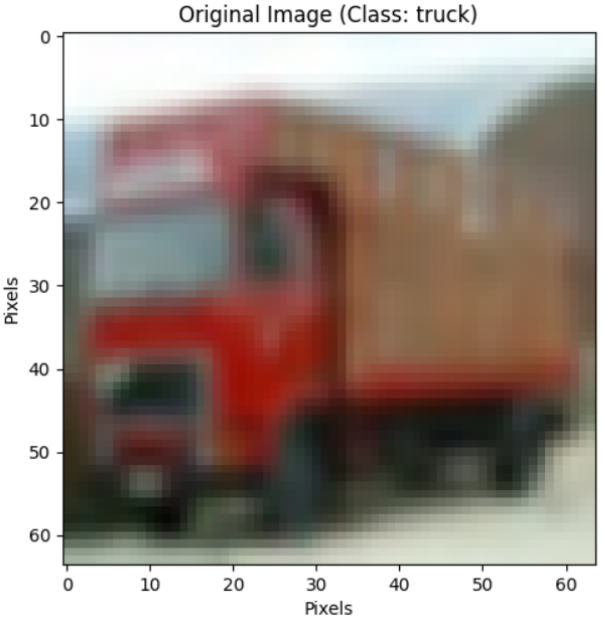

Layer 5 Saliency Map
Pixels
Pixels

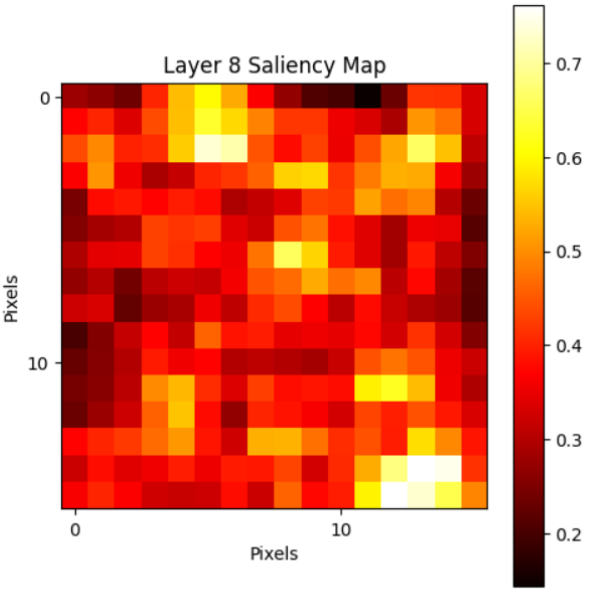

Original Image (Class: truck)
Pixels
Pixels

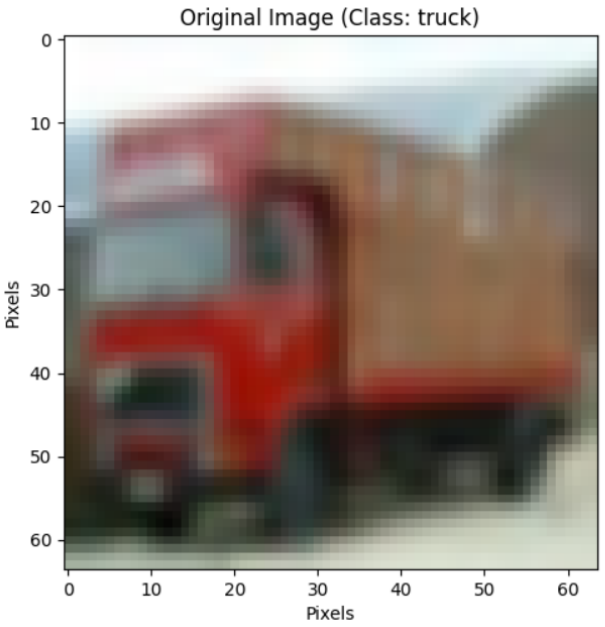

Layer 6 Saliency Map
Pixels
Pixels

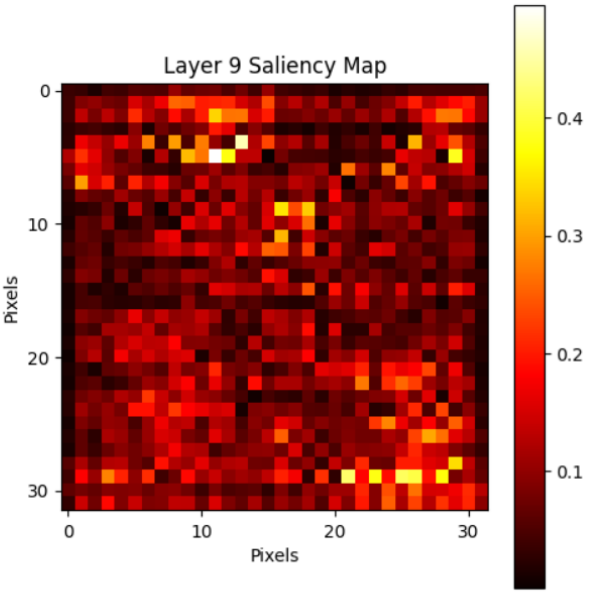

Original Image (Class: truck)
Pixels
Pixels

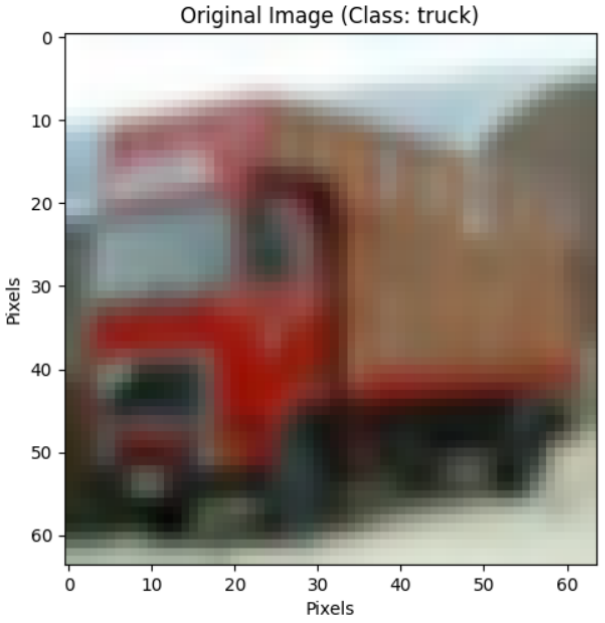

Layer 7 Saliency Map
Pixels
Pixels

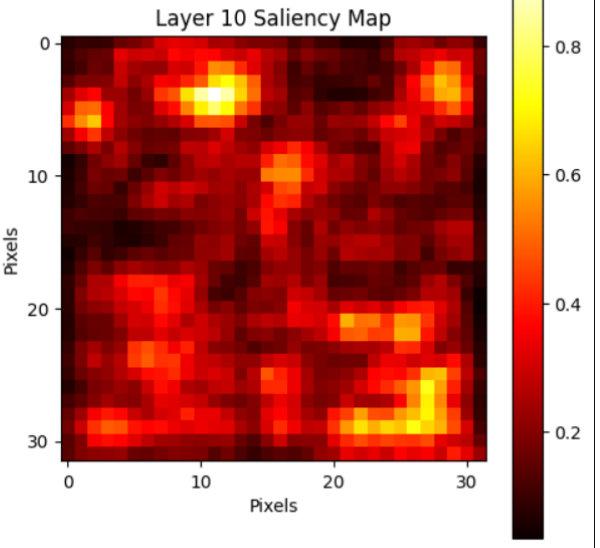

Original Image (Class: truck)
Pixels
Pixels

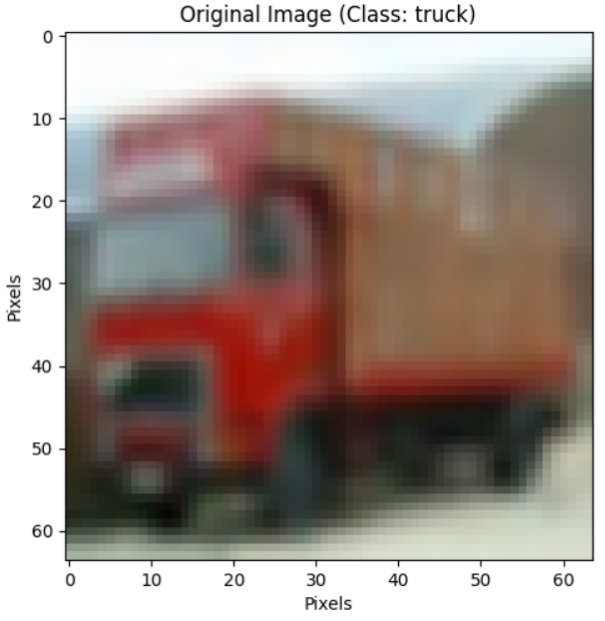

Layer 8 Saliency Map
Pixels
Pixels

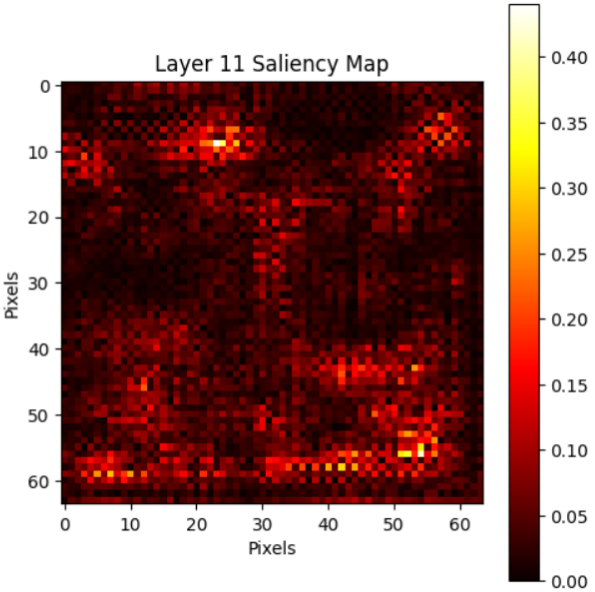

Original Image (Class: truck)
Pixels
Pixels

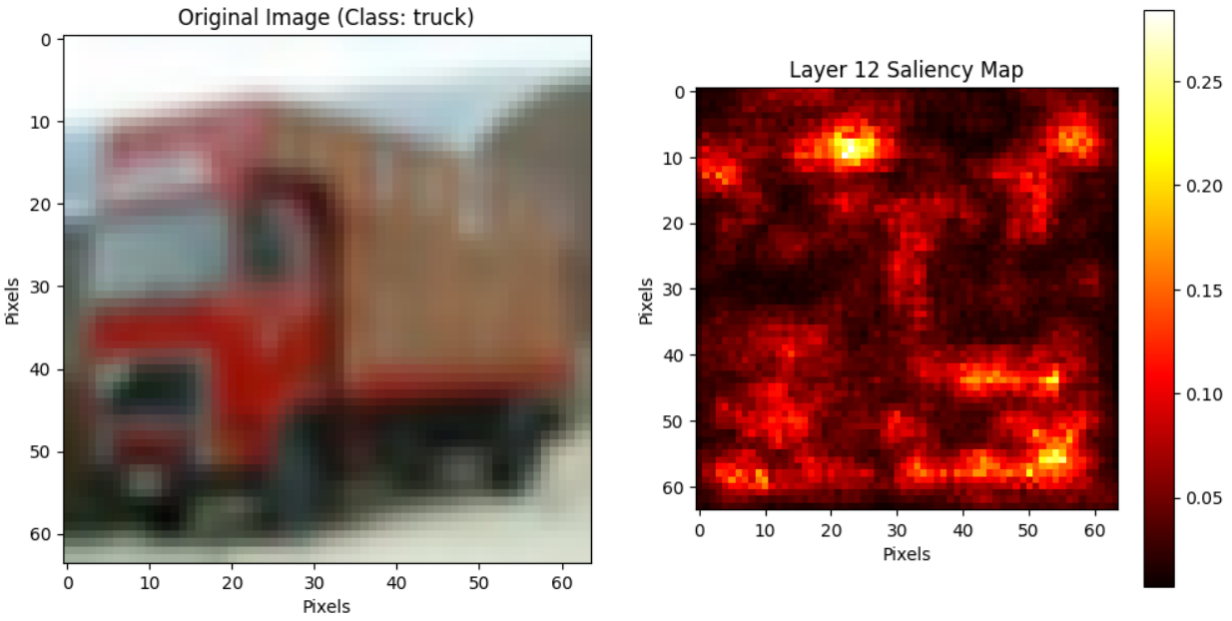

Layer 9 Saliency Map
Pixels
Pixels


Original Image (Class: truck)
Pixels
Pixels


Layer 10 Saliency Map
Pixels
Pixels


Original Image (Class: truck)
Pixels
Pixels


Layer 11 Saliency Map
Pixels
Pixels



The above visualizations show the saliency map for different layers for the image of the truck. The saliency map visualizations show what regions of the pixels in the image are influential in different layers while classifying the above image as a "truck".

From layers 1 to 12, we can see a progression in the saliency map representations. Early layers often detect simple features like edges and basic textures, while deeper layers tend to represent more abstract features. Below is the layer-by-layer interpretation:

Layers 1 & 2: In these layers, we can observe that the saliency seems to be quite distributed, with emphasis on broader regions. This might indicate that the network is focusing on general shapes and boundaries in the image.

Layers 3 to 5: In these intermediate layers, we observe that the saliency is becoming more focused on specific areas, suggesting that the network is beginning to hone in on more detailed features that are important for identifying the truck. We see that the saliency maps are getting more defined, indicating the identification of more complex features.

Layers 6 to 8: Here, we can observe that the saliency maps are much more concentrated in particular regions. These layers likely represent the parts of the truck that are most distinctive for classification, such as the corners, the cab of the truck, or distinctive color patterns.

Layers 9 & 10: In these layers, we observe that the saliency maps are very focused, with bright spots showing where the network is paying the most attention. This could correspond to high-level features that define the truck class, such as the structure of the cargo area, wheels, or other unique identifiers.

Layers 11 & 12: The saliency map visualization shows the finest level of pixel details in these layers. Their saliency maps show a few highly focused areas of bright color, indicating where the network is concentrating its attention most strongly. Since these are the deepest layers, the saliency map is less about individual pixels and more about the complex relationships and patterns the network has learned to recognize as being indicative of a truck.

Limitations of Saliency Maps:

While Saliency maps are certainly useful if we need to analyze finer details in terms of pixels, they have some limitations:

- Saliency maps can sometimes be noisy or misleading, focusing on areas that seem unintuitive to humans.
- Saliency map interpretation is relatively simplistic and might not fully capture the complex dependencies or interactions in the data that the model is leveraging.

Conclusion:

Implementing saliency maps in our model has produced fantastic insights into the neural network's inner workings. These visual guides map out the regions of an image that most captivate the model's attention,' enabling a peek into the opaque decision-making process of deep learning. By highlighting the areas that significantly impact the model's predictions, saliency maps serve as a window into the AI's thought process. This tool not only deepens our understanding of AI's interpretive strategies but also moves us closer to demystifying the complex layers of machine 'perception.' It's a considerable stride towards transparent AI, allowing us to witness the intricate dance of pixels that inform a model's final decision.

Exploring LIME Technique for model explainability:

Local Interpretable Model-agnostic Explanations (LIME) is a technique that helps us understand how machine learning models make their decisions. Essentially, LIME simplifies the model's process in a specific area to make it easier to interpret. It works by making small changes to the input data and observing how these changes affect the model's predictions. This helps to pinpoint which features in the data are most important for the model's decision, making it a valuable tool for interpreting complex models like deep neural networks, where it's usually hard to see what's driving the predictions.

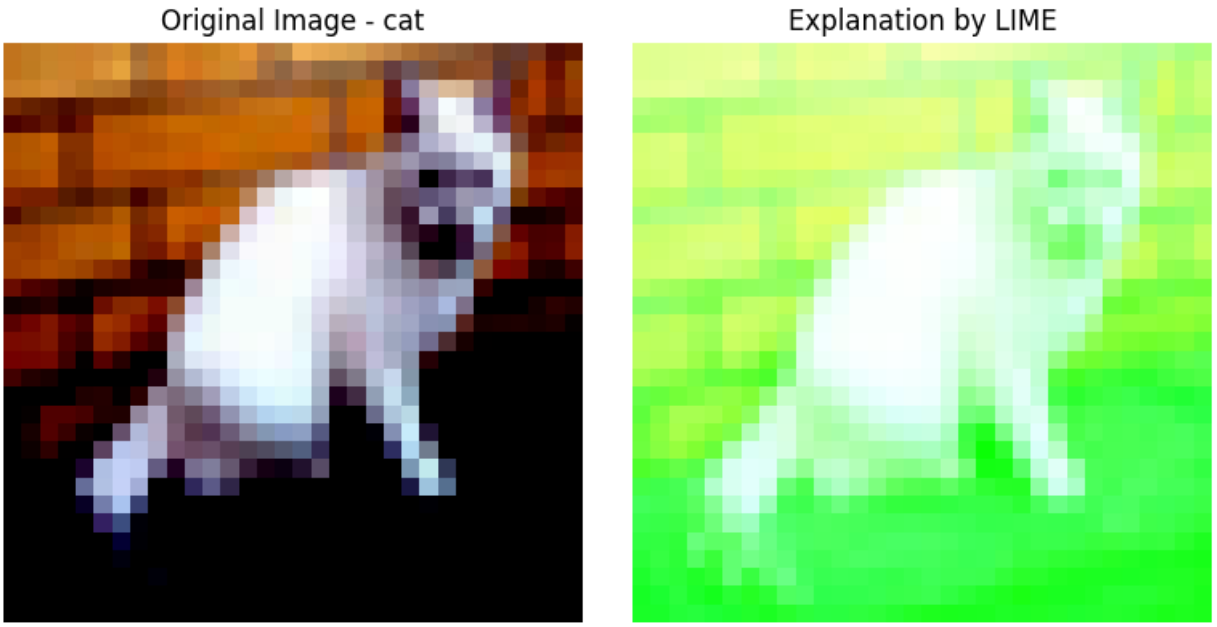
Implementation and results:

We normalized the images using defined mean and standard deviation values to ensure input to the model is standardized. We then applied LIME to a sample image. The technique generated perturbations and used the model's responses to these to train a local model. This model's coefficients then provided insight into the importance of different pixels in the classification decision.

The first image is the actual image as perceived by the human eye. In this case, it's a pixelated image of a cat.

The second image is a heat map overlay on the original image. This visualization colors different regions of the image based on their importance to the model's prediction. The most influential regions are highlighted, indicating where the model is looking to make its decision.

In the Lime explanation image, areas in green around the cat's body and face suggest these regions are most influential in the model's decision-making process. The intensity of the green indicates the strength of the influence.



Limitation of Lime technique:

- Although LIME works with any model, the details of the model can still affect its explanations. The insights LIME provides for one set of data may not apply to other parts of the data.
- The explanations LIME gives, especially with complex data like images might not always be straightforward to use.
- Creating explanations with LIME can be resource-intensive when dealing with many data variations and complicated models like deep neural networks.

Conclusion:

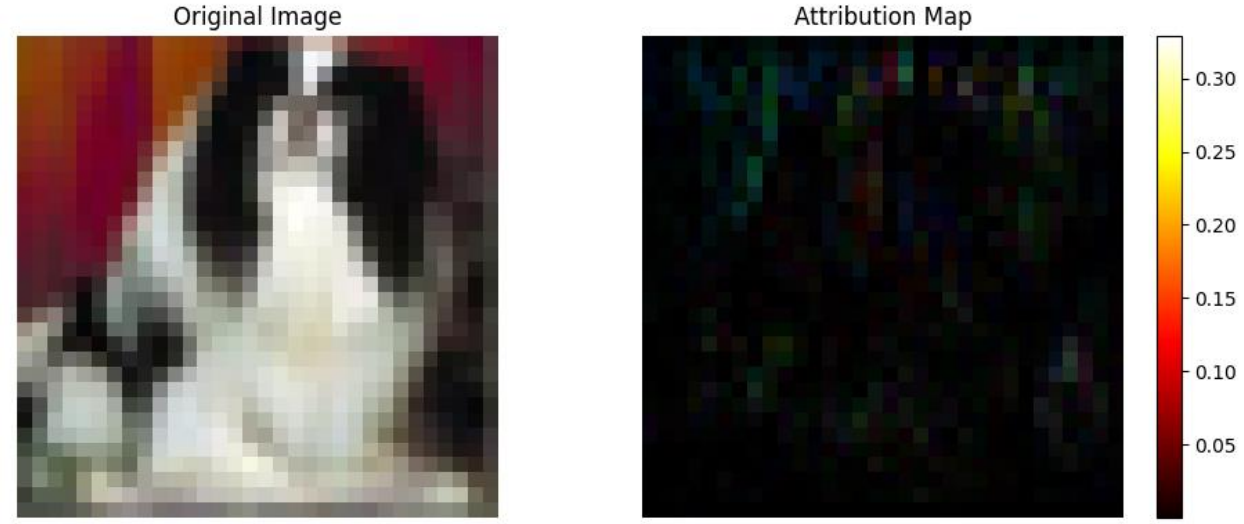
The LIME explanation helped in understanding the reasoning behind the VGG13 model's classification of an image. By highlighting which features are most influential, it aids in assessing whether the model is focusing on sensible, recognizable features of the cat. This can be particularly useful for debugging the model (by identifying any focus on irrelevant features) and for guiding further model improvements and training strategies to enhance accuracy and reliability in classifications.

Exploring Integrated Gradient Technique for model explainability:

Integrated Gradients (IG) is useful in understanding the contribution of each input feature to the final prediction. This technique is widely used to ensure transparency and reliability in model decisions, especially in sensitive fields.

Implementation and results:
We instantiated Integrated Gradients with the model to decompose the prediction into contributions of each input feature.
Loaded a batch of data from train_loader, and the relevant inputs to the computation device and Integrated Gradients computed the attributions by interpolating the input and its baseline and averaging the gradients at all interpolation points. The number of steps for the interpolation was set to 50, to provide a balance between computational efficiency and granularity of the explanation. Also, we normalized the attributions to visualize the data effectively.

The left image in the output is the input image. The right image is the attribution map with hot color map to represent the intensity of attributions, where warmer colors indicate higher importance. This image displays the normalized importance of each pixel in determining the model's prediction. Several regions of the image are highlighted in varying intensities of red and yellow. These highlights suggest that the model considers these areas crucial for identifying the target class. This insight can help in understanding the model's behavior, verifying if the model is focusing on sensible features, and diagnosing any potential biases or shortcomings in the model's learning.



Limitations:

- The baseline we choose can greatly impact the results. Using the wrong baseline might lead to incorrect conclusions.
- For complex data types like images, it can be difficult to clearly understand the results. This is because images contain many overlapping features that can complicate the interpretation.

Conclusion:

We were able to generate detailed insights into how our model processes and interprets its input data. The technique enhances model transparency by highlighting how each part of the input contributes to the output. This method is particularly valuable for validating the model's decision-making process, ensuring that the model behaves as expected, and identifying any areas for improvement.

Exploring SHAP Technique for model explainability:

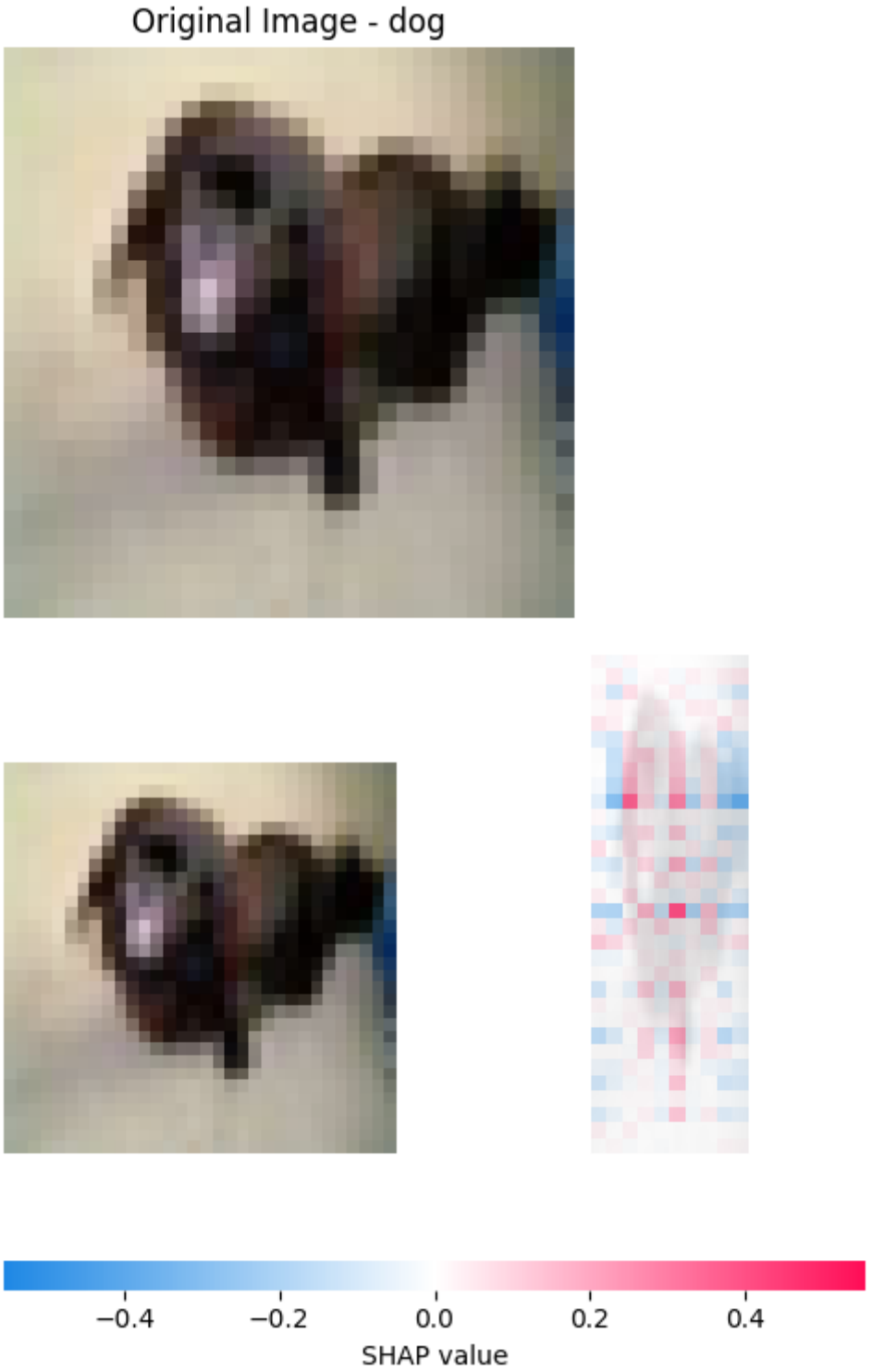
SHAP (SHapley Additive exPlanations) is a powerful interpretability technique based on game theory, specifically the Shapley values, which quantify the contribution of each feature to the prediction of a machine learning model. This method is particularly helpful in providing insights into the decision-making process of complex models.

Implementation and Results:
We selected a background dataset from our training loader to set a reference for the model and is critical for SHAP's calculations. We then utilized SHAP's DeepExplainer, which is tailored for

deep learning models, using our model and the background dataset to initialize the explainer. SHAP values were calculated for a subset of the background data to understand the impact of each feature in the model's predictions. Adjusted output images and SHAP values by reversing the normalization process applied during model training, ensuring that the visualizations are meaningful and comparable to the original image data.

In the output, original image was displayed to provide a baseline for comparison with the SHAP values. SHAP values were visualized using SHAP's image plot, which overlays the SHAP values on the original image. This overlay uses a color-coded scheme where blue represents a negative impact (decreasing the likelihood of the prediction) and red indicates a positive impact (increasing the likelihood of the prediction).

The SHAP visualization effectively highlights the areas in the image that significantly influence the model's prediction. For the image of the dog, specific regions such as parts of the dog's face and body were highlighted. These areas are crucial in identifying the image as a dog, according to the model's learned features.



Limitations:

- SHAP values can be sensitive to the distribution of data in the training set. If the background dataset used for reference does not represent the overall data distribution well, the SHAP values might not accurately reflect the influence of features.
- SHAP provides a quantitative measure of feature importance but translating these values into actionable insights can be challenging.
- Depending on the implementation, SHAP explanations may lack consistency across different runs, especially when the method involves sampling or approximations.

Conclusion:

Our implementation of the SHAP technique provides deep insights into the contributing factors behind the model's predictions, enhancing the transparency of our machine learning process. By identifying which features are most influential, we can better understand and trust our model's decisions. This method is invaluable for debugging and improving the model, ensuring that it focuses on relevant features and performs well in practical applications.

Final Conclusion:

In this project, we attempted to explain and implement some popular explainability techniques to understand the working of deep learning models. As we conclude this project, we attempt to explain the role played by Explainability techniques to enhance the trustworthiness of AI models:

1. **Transparency:**

    - Explainability helps demystify the decision-making process of AI models by showing which features influence outcomes. This transparency allows developers and end-users to understand how and why specific decisions are made.
    - By providing insights into the model's reasoning, stakeholders can verify whether the AI is making decisions based on relevant features or biases, thus ensuring the model behaves as expected.

2. **Accountability:**

    - Explainable AI can help pinpoint when and why errors occur, which is crucial for diagnosing problems and improving model performance. This ability is especially valuable in high-stakes areas such as healthcare, finance, and law.
    - Increasingly, regulations require transparency in AI systems (eg: EU's GDPR). Explainability aids in compliance by providing the necessary documentation and reasoning for decisions made by AI, which is required for legal accountability.

3. **Fairness:**

    - Explainability tools can reveal whether a model's decisions are unfairly biased towards certain groups. Understanding these biases allows developers to make necessary adjustments to data or model architectures to promote fairness.
    - By ensuring AI systems operate without unintended or discriminatory bias, explainability supports the ethical deployment of AI technologies.

4. **User Confidence and Trust:**

    - When users understand how AI models make decisions, they are more likely to trust and rely on AI-driven recommendations. This confidence is crucial for user adoption in sectors like autonomous vehicles and personal finance.
    - Explainable models allow users to provide feedback on AI decisions, facilitating iterative improvements. This feedback loop not only improves model accuracy but also aligns AI operations with human values and expectations.

Contribution Summary:

Team Member	Project Part	Contribution(%)
Kusha Kumar Dharavath	Model Training, GradCAM technique and Report	100
Kishan Nagaraja	Implemented Saliency Map, LIME and Report	100
Dharma Acha	Implemented Integrated Gradient, SHAP and Presentation	100

References:

- https://arxiv.org/abs/1708.08296
- Lundberg, S.M., Lee, S.I. (2017). A Unified Approach to Interpreting Model Predictions. NIPS.
- Diagnostics | Free Full-Text | Explainable AI for Retinoblastoma Diagnosis: Interpreting Deep Learning Models with LIME and SHAP (mdpi.com)
- Grad-CAM: Visual Explanations From Deep Networks via Gradient-Based Localization (thecvf.com)
- Integrated gradients | TensorFlow Core
- Visualizing Deep Networks by Optimizing with Integrated Gradients (thecvf.com)
- Explain an Intermediate Layer of VGG16 on ImageNet — SHAP latest documentation
- Applied Sciences | Free Full-Text | Exploring the Capabilities of a Lightweight CNN Model in Accurately Identifying Renal Abnormalities: Cysts, Stones, and Tumors, Using LIME and SHAP (mdpi.com)
- 1703.00152 (arxiv.org)
- dharmaac_kushakum_assignment1
- kushakum code demo