Grad-CAM

1. Objective:

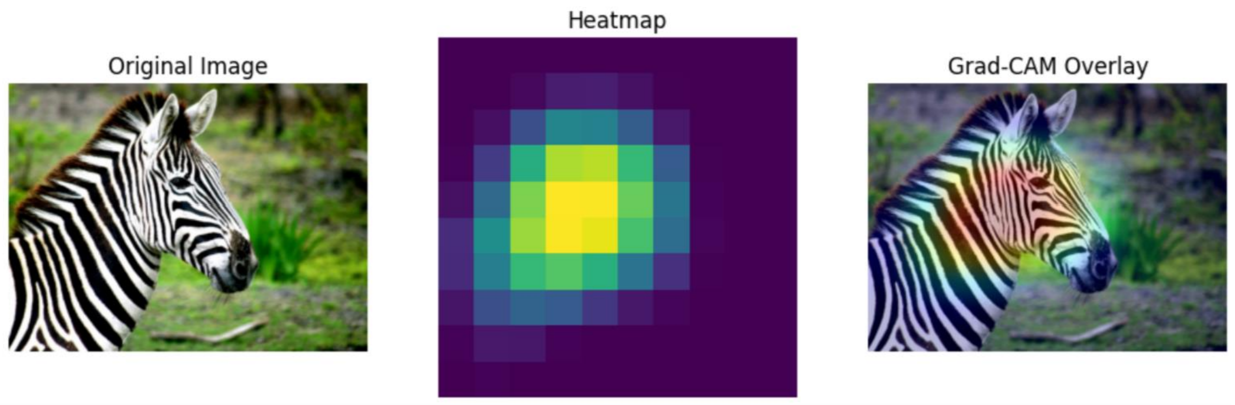
The objective of Grad-CAM (Gradient-weighted Class Activation Mapping) is to provide insights into the decision-making process of neural network models by highlighting regions in the input images that significantly contribute to the model's predictions. This report discusses the implementation and application of Grad-CAM on two different datasets: CIFAR-10 and NEU (Northeastern University) Surface Defects. Implemented a script using the Xception model with Keras and TensorFlow as the backend, a deep learning model pre-trained on ImageNet for image classification. Adapted and implemented the script for image classification tasks on CIFAR-10 and NEU datasets using PyTorch.

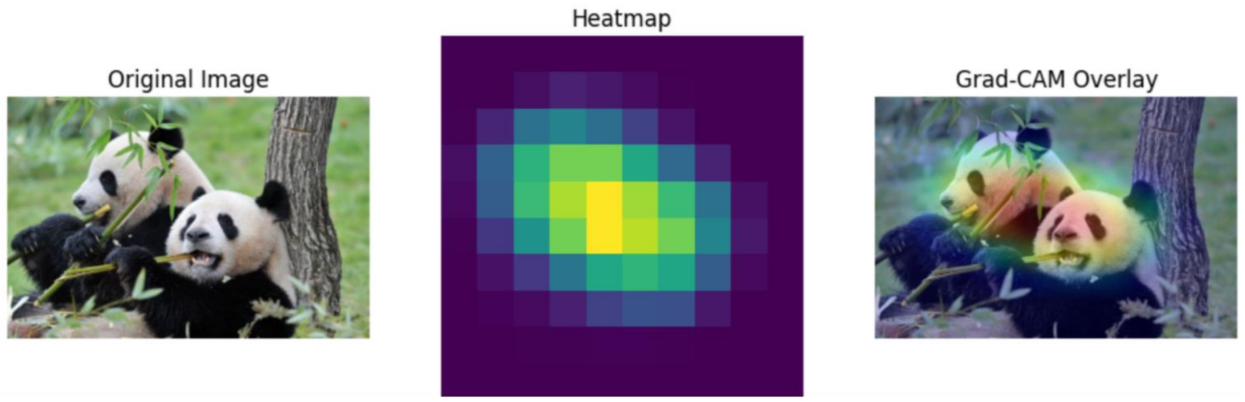
2. History of Grad-CAM:

Grad-CAM was introduced by Selvaraju et al. in the paper titled "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization" (2017). Prior techniques like CAM (Class Activation Mapping) focused on global average pooling, but Grad-CAM introduced a gradient-weighted approach for localization. Grad-CAM's flexibility and ability to work with any differentiable model made it widely adopted in the field of interpretability.

3. Using Keras:

implements the Grad-CAM (Gradient-weighted Class Activation Mapping) technique using the Xception model with Keras and TensorFlow. It loads an image of a zebra, preprocesses it, and generates a heatmap that highlights the regions contributing to the model's prediction. The script then displays the original image, the computed heatmap, and a Grad-CAM overlay for enhanced interpretability. The Xception model, pre-trained on ImageNet, is utilized, showcasing its effectiveness in visualizing the neural network's decision-making process. The resulting visualizations aid in understanding the model's focus areas and contribute to the broader interpretability of deep learning models.





4. Using pytorch:

Grad-CAM workflow:

1. Model Definition:

Begin with a pre-trained or custom neural network model in PyTorch.

2. Hook Registration:

Identify the target layer where you want to visualize activations.

Register forward and backward hooks on this layer to capture activations during the forward pass and gradients during the backward pass.

3. Image Preprocessing:

Preprocess the input image to align with the model's requirements (resizing, normalization, etc.).

4. Forward and Backward Pass:

Perform a forward pass with the preprocessed image through the model.

Compute the loss associated with the predicted class.

Perform a backward pass to calculate gradients.

5. Compute Gradients and Activations:

Use the captured gradients and activations from the hooks to compute the Grad-CAM heatmap.

Calculate pooled gradients and weight the channels in the activations.

6. Post-process Heatmap:

Resize the heatmap to match the original image size.

Optionally, apply smoothing techniques like Gaussian blur for better visualization.

7. Superimpose on Original Image:

Convert the original image and heatmap to the appropriate data types.

Combine the original image and the heatmap using a weighted sum to create the final Grad-CAM visualization.

8. Visualization:

Display the original image, heatmap, and superimposed image for interpretation.

9. Unregister Hooks:

Ensure that you unregister the hooks after visualization to avoid interfering with subsequent computations.

Key Points:

Hooks: They capture intermediate results during forward and backward passes.

Gradients: The gradients highlight important regions for the prediction.

Heatmap: The Grad-CAM heatmap visually emphasizes regions where the model focuses to make predictions.

Superimposition: Overlay the heatmap on the original image to create an interpretable visualization.

This workflow provides insight into the model's decision-making process, helping understand which parts of the input image contribute most to the final prediction. Grad-CAM is a powerful interpretability tool, applicable across various neural network architectures and datasets.

5. Dataset and Model Information:

- **CIFAR-10:**

Dataset: CIFAR-10 consists of 60,000 32x32 color images in 10 different classes. This is a built-in dataset.

Model: A modified ResNet18 is used for CIFAR-10, with the last fully connected layer removed. Train

accuracy of the model used was 84.50%, Validation Accuracy was 92.01%

- **NEU Surface Defects:**

Dataset: The NEU Surface Defects dataset contains images of six different types of surface defects.

Model: A ResNet50 model pre-trained on ImageNet is used as the base model for NEU. Training and

validation accuracies of the model used were 99.69 % and 100%

6. Grad-CAM Implementation:

Common Steps:

Both implementations share common steps:

Model Definition: A modified ResNet-188 model for CIFAR-10 and a ResNet50 model for NEU.

Model Loading: Pretrained models are loaded for both datasets.

Hook Registration: Hooks are registered on the last convolutional layer to capture gradients during forward and backward passes.

Image Preprocessing: Input images are preprocessed to meet the model's requirements.

7. Grad-CAM Visualization:

Common Visualization Steps:

Both implementations visualize the following:

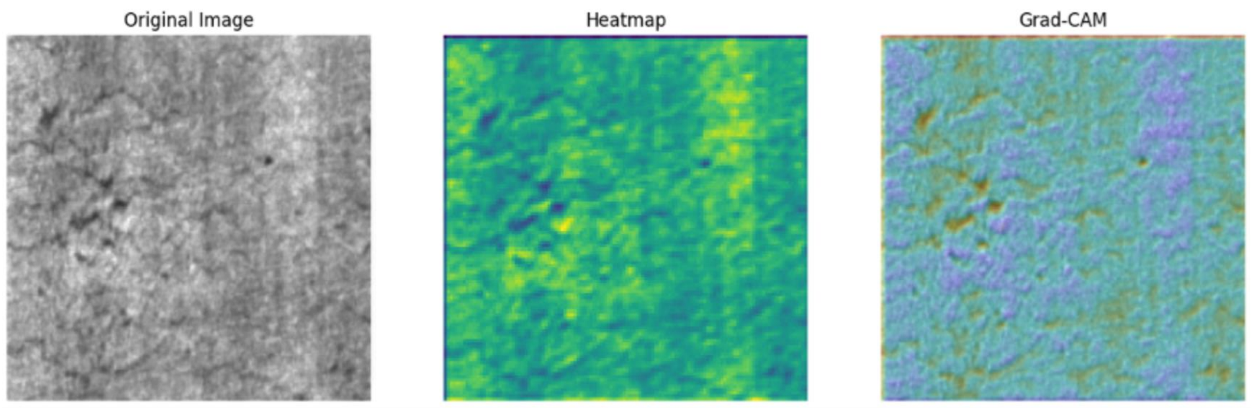
Original Image: The input image used for Grad-CAM analysis.

Heatmap: A heatmap highlighting regions that significantly influence the model's predictions.

Superimposed Image: The original image with the overlaid heatmap for enhanced interpretability.

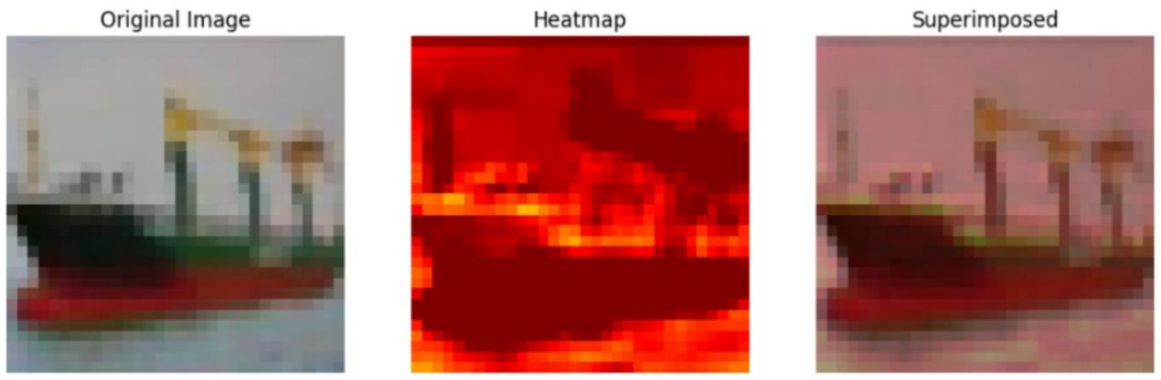
NEU Data:

Used viridis color for the heatmap.



CIFAR-10 Visualization Specifics:

Resizing of input images to (224, 224) to match the expected input size of the pre-trained ResNet18 model. Used color JET for the heatmap.



8. Conclusion:

Grad-CAM is a powerful tool for interpreting neural network models.

The combination of original images, heatmaps, and superimposed images provides a comprehensive understanding of model decision-making.

These insights can be crucial for model debugging, validation, and improving model performance.

Grad-CAM is a valuable technique for model interpretability.

The implementations on CIFAR-10 and NEU datasets showcase their versatility and effectiveness in visualizing the crucial regions in input images.