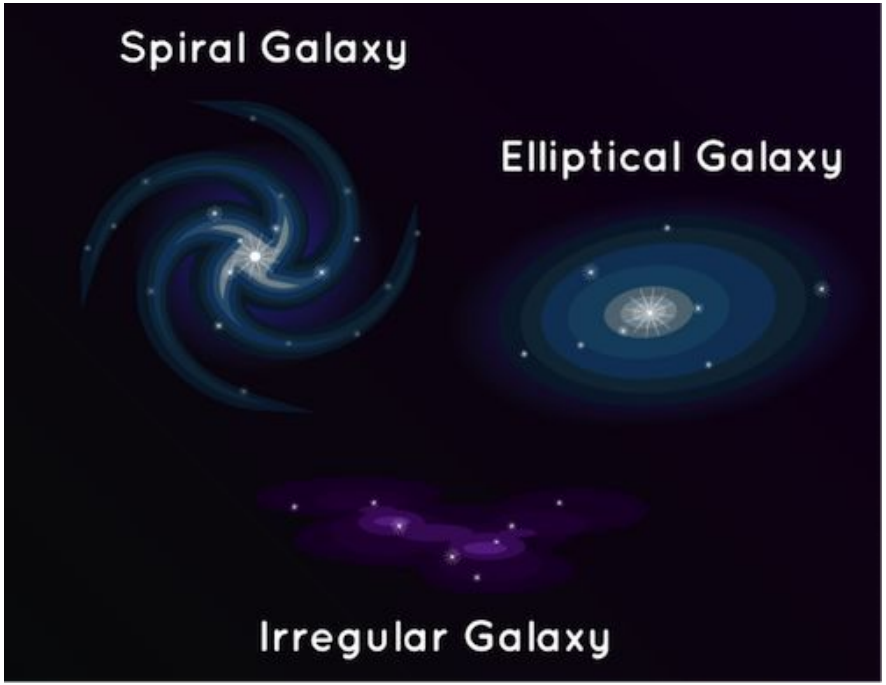
Explainable Transformer based Approach for Galaxy classification

What is GAMA ?

the GAMA Elliptical Galaxy Classification project is dedicated to cataloging and analyzing the morphological types of galaxies, with a specific focus on **elliptical galaxies,** using data from the GAMA survey. This work helps improve our understanding of galaxy formation and evolution by providing a detailed and accurately classified sample of galaxies

Elliptical Galaxy ?



Why It is helpful ?

Categorizing elliptical galaxies from other types helps astronomers understand galaxy formation and evolution, study the influence of environment on galaxy morphology, analyze **stellar populations** and **gas content**, investigate **dark matter** distribution, and refine **cosmological models**. This classification enhances our knowledge of the universe and supports more accurate astronomical theories.

Data Loading and Preprocessing

Loading the image using **Keras**.

Resizing the image to **300x300** pixels for uniformity.

Converting the image to a **NumPy** array using **Keras** :
tf.keras.preprocessing.image.img_to_array.

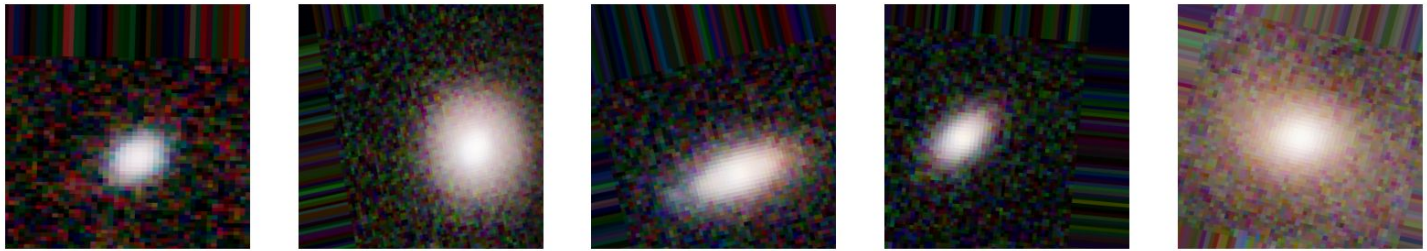
Data Augmentation in Machine Learning

Definition: Data augmentation is a technique used to artificially expand the size of a training dataset by creating modified versions of the original data. This is done by applying various transformations such as rotation, scaling, translation, and flipping to the original data samples.

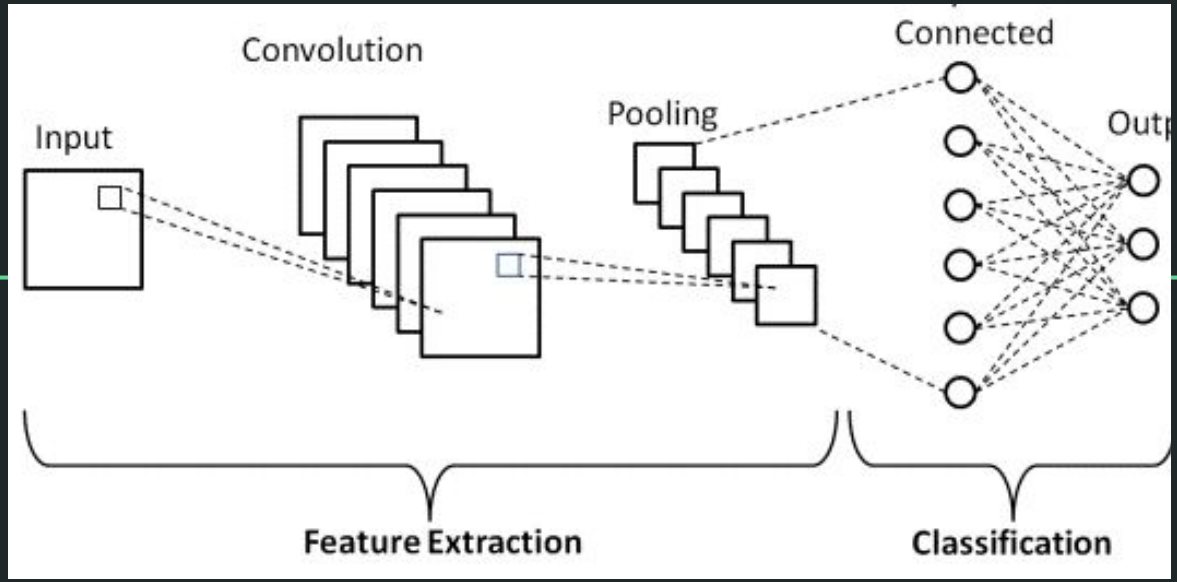
Purpose: The main goal of data augmentation is to improve the generalization capability of machine learning models, particularly deep learning models, by providing a more diverse set of training examples. This helps the models to be more robust and perform better on unseen data.

Dive in Code

```
# Data augmentation

datagen = ImageDataGenerator

rotation_range=20, # Randomly rotate images by up to 20 degrees

width_shift_range=0.2, # Randomly shift images horizontally by 20%

height_shift_range=0.2, # Randomly shift images vertically by 20%

shear_range=0.2, # Apply shearing transformations

zoom_range=0.2, # Randomly zoom in and out by 20%


horizonta-nearer # #R do new pipes with tre nearest pixel value
```



Convolutional Neural Network (CNN)

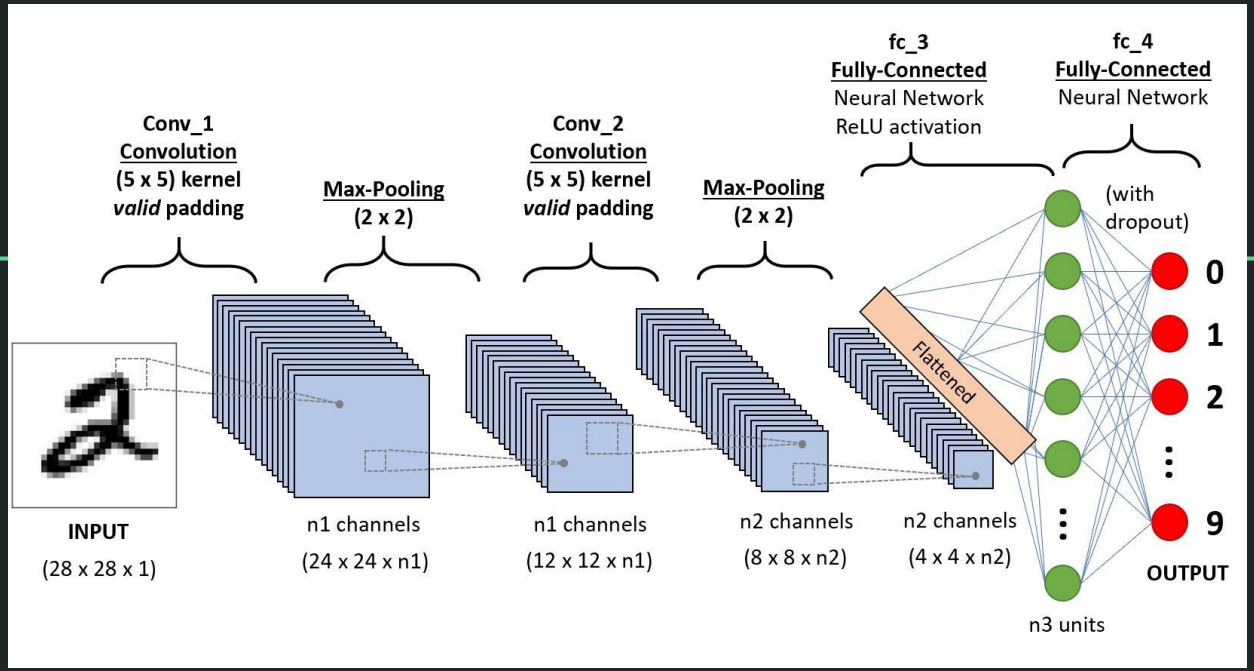


A Convolutional Neural Network (CNN) works by processing images through several layers:

- **Input Layer:** Takes the image as input.
- **Convolutional Layers:** Detect features like edges by applying filters. The result is a feature map.
- **Activation Function:** Adds non-linearity to help learn complex patterns.
- **Pooling Layers:** Reduce the size of the feature maps, making the network more efficient.
- **Fully Connected Layers:** Combine features to make final predictions.

The network learns by adjusting filter weights during training to minimize prediction errors.

A simple CNN processes images through multiple layers to identify features and make predictions. It starts with an **input layer** that takes the image, followed by convolutional layers that apply filters to detect features like edges and textures, creating **feature maps**. These maps are passed through **activation functions** (e.g., ReLU) to introduce non-linearity. **Pooling layers** then reduce the spatial dimensions, making the network more efficient. Finally, **fully connected layers** combine these features to make the final prediction, such as classifying the image. The network learns by adjusting the filter weights during training to minimize prediction errors.



First CNN Model Architecture

- **Input:** 300x300x3 (RGB Image)
- **Conv Layer 1:**
  - Conv2D: 32 filters, 3x3 kernel, ReLU activation
  - MaxPooling2D: 2x2 pool size
- **Conv Layer 2:**
  - Conv2D: 64 filters, 3x3 kernel, ReLU activation
  - MaxPooling2D: 2x2 pool size
- **Conv Layer 3:**
  - Conv2D: 128 filters, 3x3 kernel, ReLU activation
  - MaxPooling2D: 2x2 pool size
- **Fully Connected Layers:**
  - Flatten
  - Dense: 512 units, ReLU activation
  - Dropout: 0.5
- **Output Layer:**
  - Dense: 1 unit, Sigmoid activation

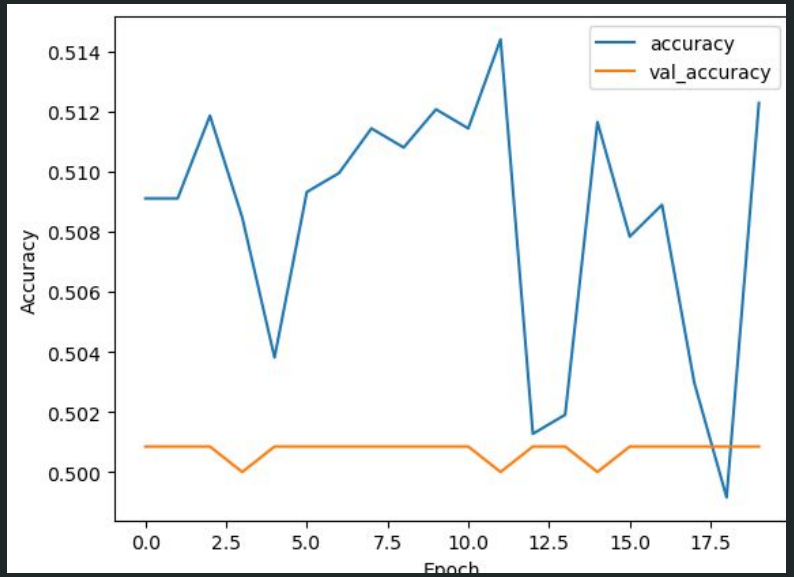
Optimizer: Adam
Loss Function: Binary Cross Entropy

Training Details:

Epochs: 20
Batch Size: 32
Results:

Training Loss: Decreased from 9.9256 to 0.6970
Training Accuracy: Around 50.91%

Validation Loss: Stabilized around 0.6933
Validation Accuracy: Consistently around 50.08%
Conclusion:



The model did not improve significantly, with accuracy around 50%, indicating no better than random guessing. Further investigation and model tuning are required.

First Try With CNN

Batch Normalization: Normalizes the activations of the previous layer, improving the stability and speed of training.

Dropout: Randomly sets a fraction (25%) of the input units to zero during training to prevent overfitting.

Second CNN Model Architecture

First CNN Model:

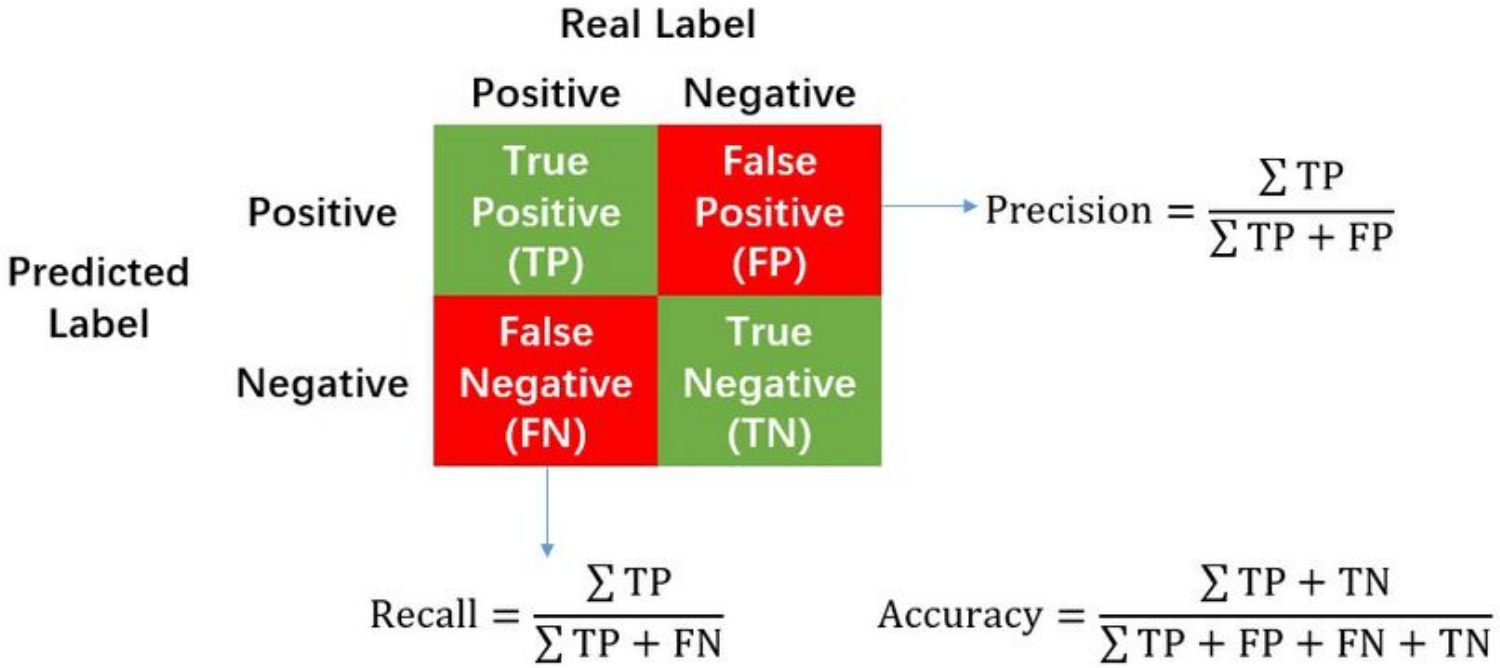
- No data augmentation
- No class weights

Second CNN Model:

- Data augmentation (rotation, shift, shear, zoom, flip)
- Class weights to handle imbalance

- **Conv Layer 1:**
  - Conv2D: 32 filters, 3x3 kernel, ReLU activation
  - **BatchNormalization**
  - MaxPooling2D: 2x2 pool size
  - Dropout: 0.25
- **Conv Layer 2:**
  - Conv2D: 64 filters, 3x3 kernel, ReLU activation
  - **BatchNormalization**
  - MaxPooling2D: 2x2 pool size
  - Dropout: 0.25
- **Conv Layer 3:**
  - Conv2D: 128 filters, 3x3 kernel, ReLU activation
  - **BatchNormalization**
  - MaxPooling2D: 2x2 pool size
  - Dropout: 0.25
- **Fully Connected Layers:**
  - Flatten
  - Dense: 512 units, ReLU activation
  - **BatchNormalization**
  - Dropout: 0.5
- **Output Layer:**
  - Dense: 1 unit, Sigmoid activation

Optimizer: Adam
Loss Function: Binary Cross Entropy



$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

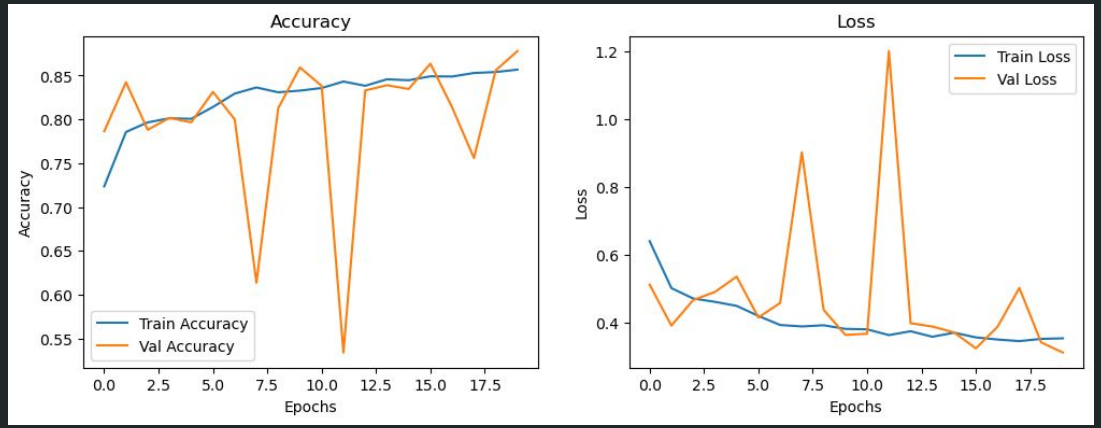
Support: This is the number of **actual occurrences** of the **class** in the dataset. It represents the number of true instances for each label.

Epoch 1:

- Training Loss: 0.6394
- Training Accuracy: 72.35%
- Validation Loss: 0.5114
- Validation Accuracy: 78.64%

Epoch 20:

- Training Loss: 0.3532
- Training Accuracy: 85.68%
- Validation Loss: 0.3112
- Validation Accuracy: 87.80%



The CNN model performed well in distinguishing between "Elliptical" and "Other" classes, with high precision, recall, and F1-scores for both classes. The consistent improvement in accuracy and reduction in loss over the epochs indicates effective learning and generalization. The final validation accuracy of 87.80% and balanced classification metrics demonstrate the model's robustness in this binary classification task.

	precision	recall	f1-score	support
Elliptical	0.89	0.87	0.88	598
Other	0.87	0.89	0.88	582
accuracy			0.88	1180
macro avg	0.88	0.88	0.88	1180
weighted avg	0.88	0.88	0.88	1180

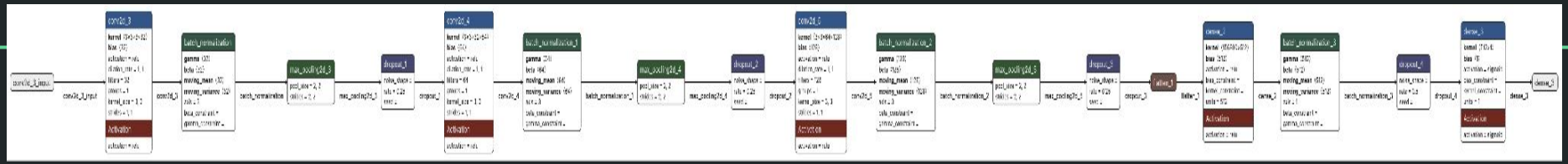
Second CNN Results

Complexity of the model

Counting the Layers of Second CNN

- **Input Layer:** 1
- **Conv2D Layers:** 3
- **BatchNormalization Layers:** 3
- **MaxPooling2D Layers:** 3
- **Dropout Layers:** 4 (1 in each convolutional block and 1 in the fully connected block)
- **Flatten Layer:** 1
- **Dense Layers:** 2 (1 in the fully connected block and 1 in the output layer)

Total Layers: 17



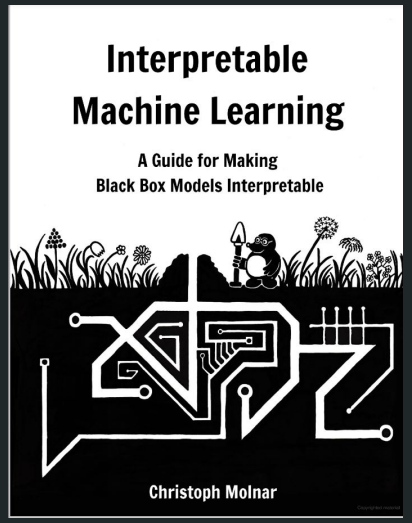
Overall, this model can be considered moderately complex. It is more sophisticated than very simple CNNs with fewer layers and parameters, but it is not as deep or complex as state-of-the-art models like VGG, ResNet, or Inception, which can have dozens or even hundreds of layers. This level of complexity is appropriate for the task at hand, balancing performance with computational efficiency.

Overfitting occurs when a model learns the details and noise in the training data to such an extent that it performs poorly on new, unseen data. Underfitting happens when a model is too simple to capture the underlying patterns in the data, leading to poor performance on both training and validation data.

1. **Dropout Layers:**
   - **Purpose:** Dropout layers randomly drop a fraction of the neurons during training, forcing the network to learn more robust features that are not reliant on any particular neurons.
   - **Implementation:** Our model has four dropout layers with varying rates:
     - Dropout (0.25) after each convolutional block (three in total).
     - Dropout (0.5) after the dense layer in the fully connected block.
2. **Batch Normalization:**
   - **Purpose:** Batch normalization normalizes the activations of each layer, which helps stabilize and accelerate the training process. It also provides some regularization, although not as strong as dropout.
   - **Implementation:** Our model includes batch normalization layers after each convolutional layer and the dense layer.
3. **Data Augmentation:**
   - **Purpose:** Data augmentation generates new training examples by applying random transformations (e.g., rotation, shifting, shearing, zooming, and flipping). This increases the diversity of the training data and helps the model generalize better.
   - **Implementation:** Our model uses `ImageDataGenerator` for data augmentation with several transformations.
4. **Class Weighting:**
   - **Purpose:** When dealing with imbalanced datasets, class weighting helps by assigning a higher penalty to misclassifications of the minority class, encouraging the model to learn equally well for all classes.
   - **Implementation:** Our model calculates and applies class weights based on the training labels.

Handling Overfitting and Underfitting

Explainable AI



Explainable AI (XAI) refers to methods and techniques in artificial intelligence (AI) that make the outputs of AI systems understandable to humans. The goal of XAI is to create AI models whose actions and decisions can be easily interpreted, providing transparency and building trust among users.

GDPR and Transparency

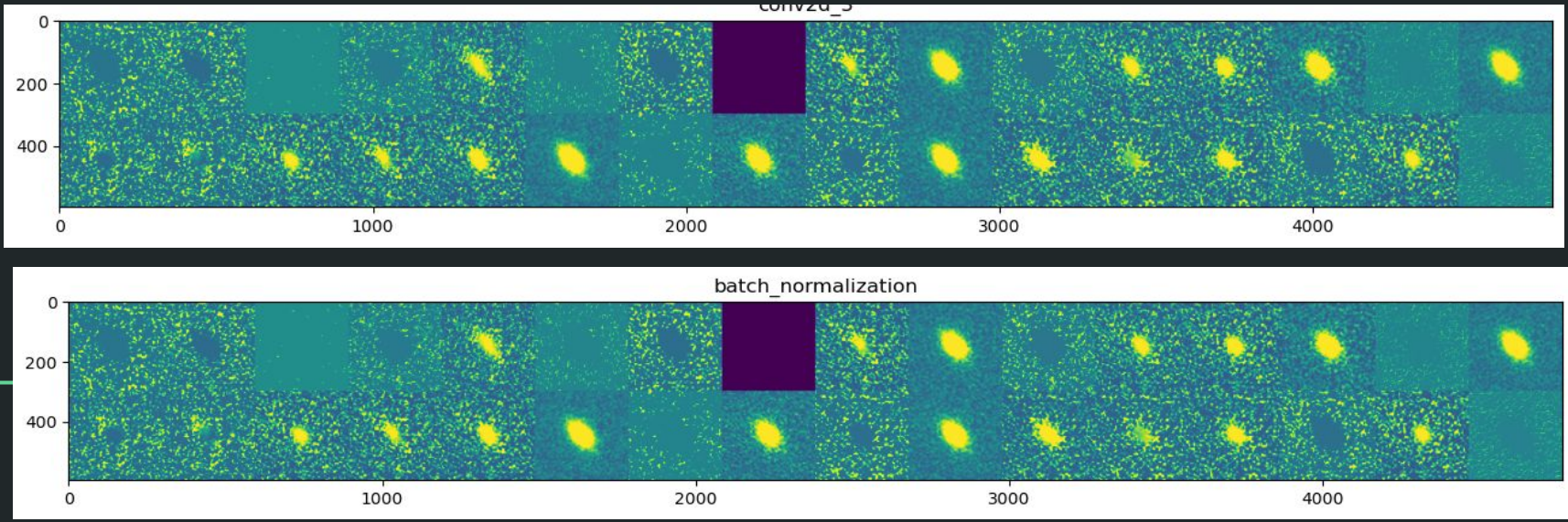
The General Data Protection Regulation (GDPR) is a comprehensive EU data protection law that impacts transparency and Explainable AI (XAI).

Right to Explanation: GDPR's Article 22 grants individuals the right to avoid decisions made solely by automated processing, including profiling, if these decisions significantly affect them. If such decisions occur, individuals are entitled to meaningful information about the logic behind them.

Visualizing activations in Explainable AI (XAI) typically involves understanding which parts of the input data (e.g., an image) activate certain parts of a neural network.

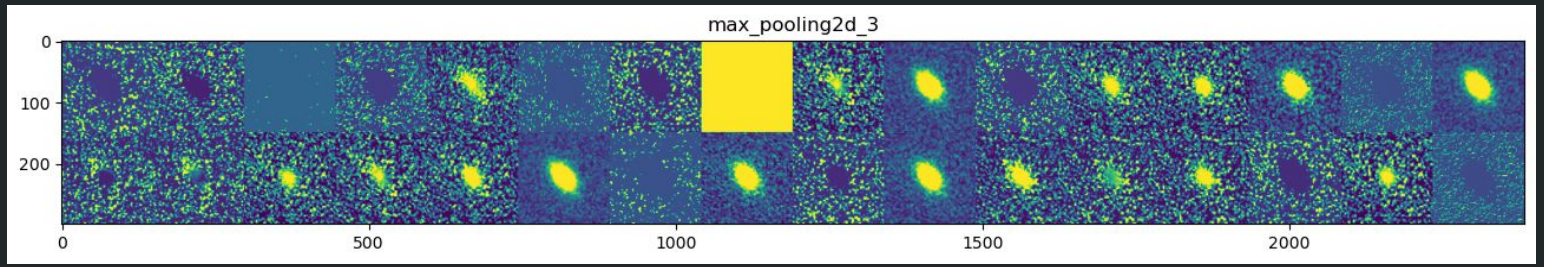
Visualizing activations

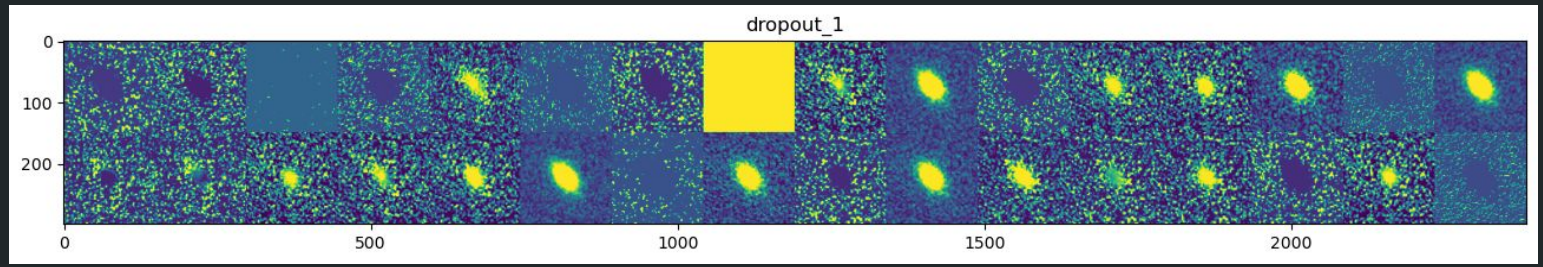
This method visualizes the activations of the first 12 layers of a convolutional neural network (CNN) model when a sample image is passed through it. The model, designed for galaxy classification

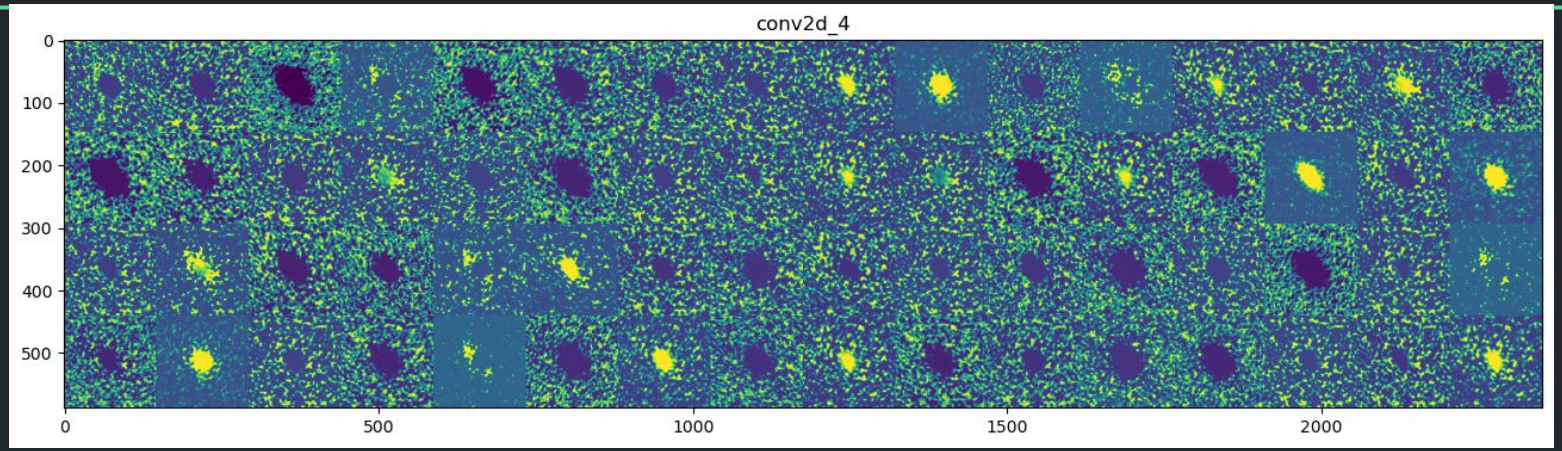


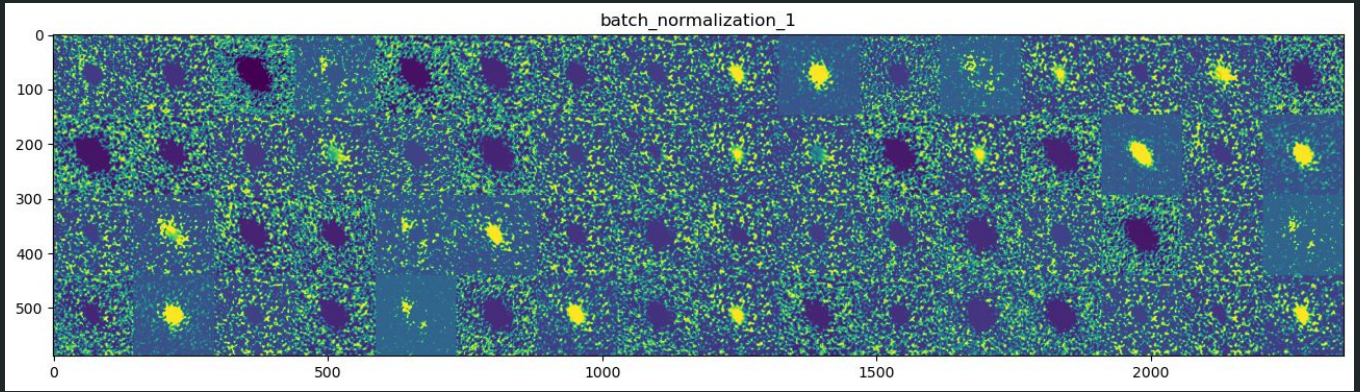
Visualizing activations is a powerful method of Explainable AI (XAI) that helps in understanding how neural networks make decisions. This technique involves examining the internal layers of a neural network to see how different inputs activate neurons within the network.

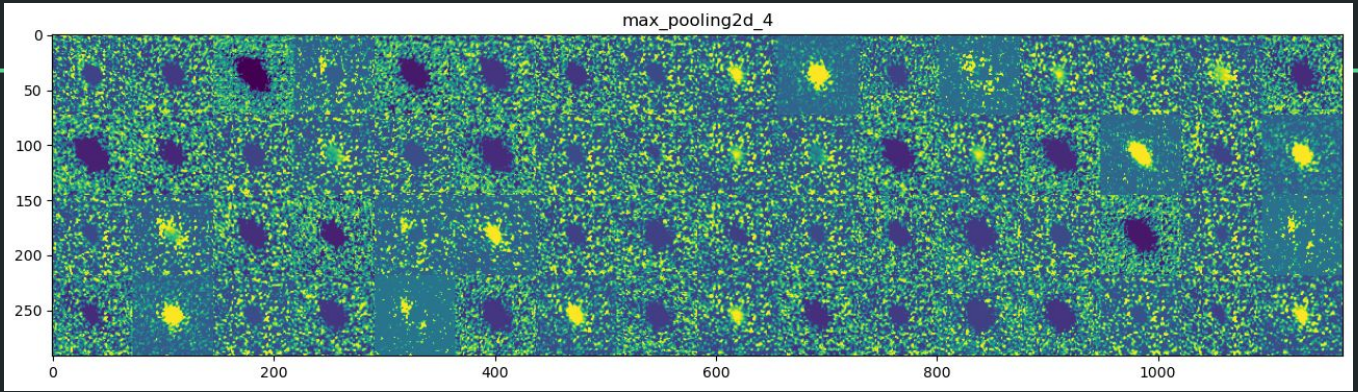
First Output : Visualizing Activations in Explainable AI

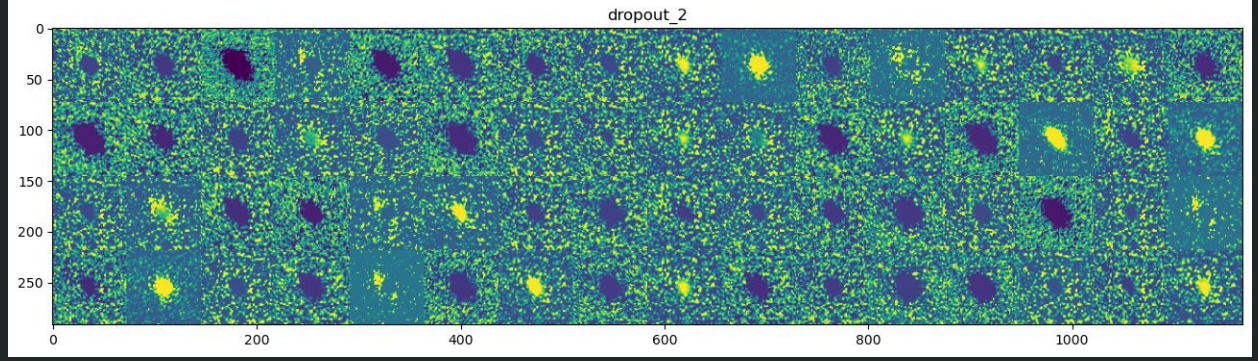

max_pooling2d_3
0
100
200
0
500
1000
1500
2000

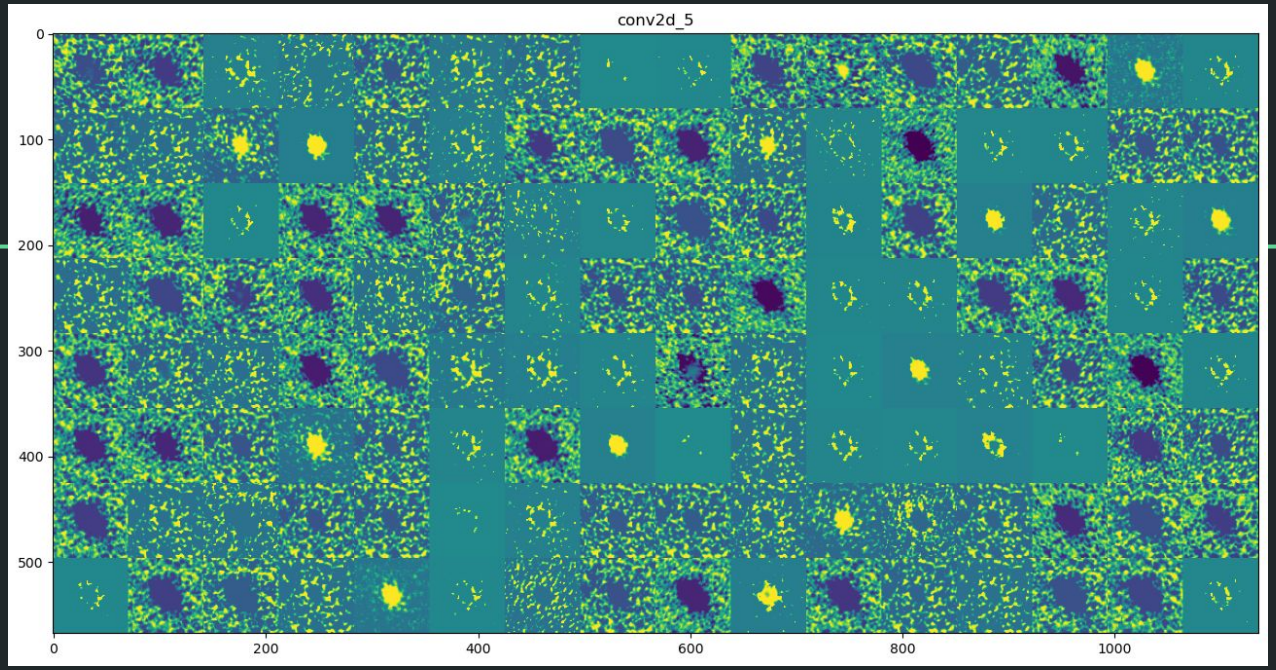

dropout_1
0
100
200
0
500
1000
1500
2000

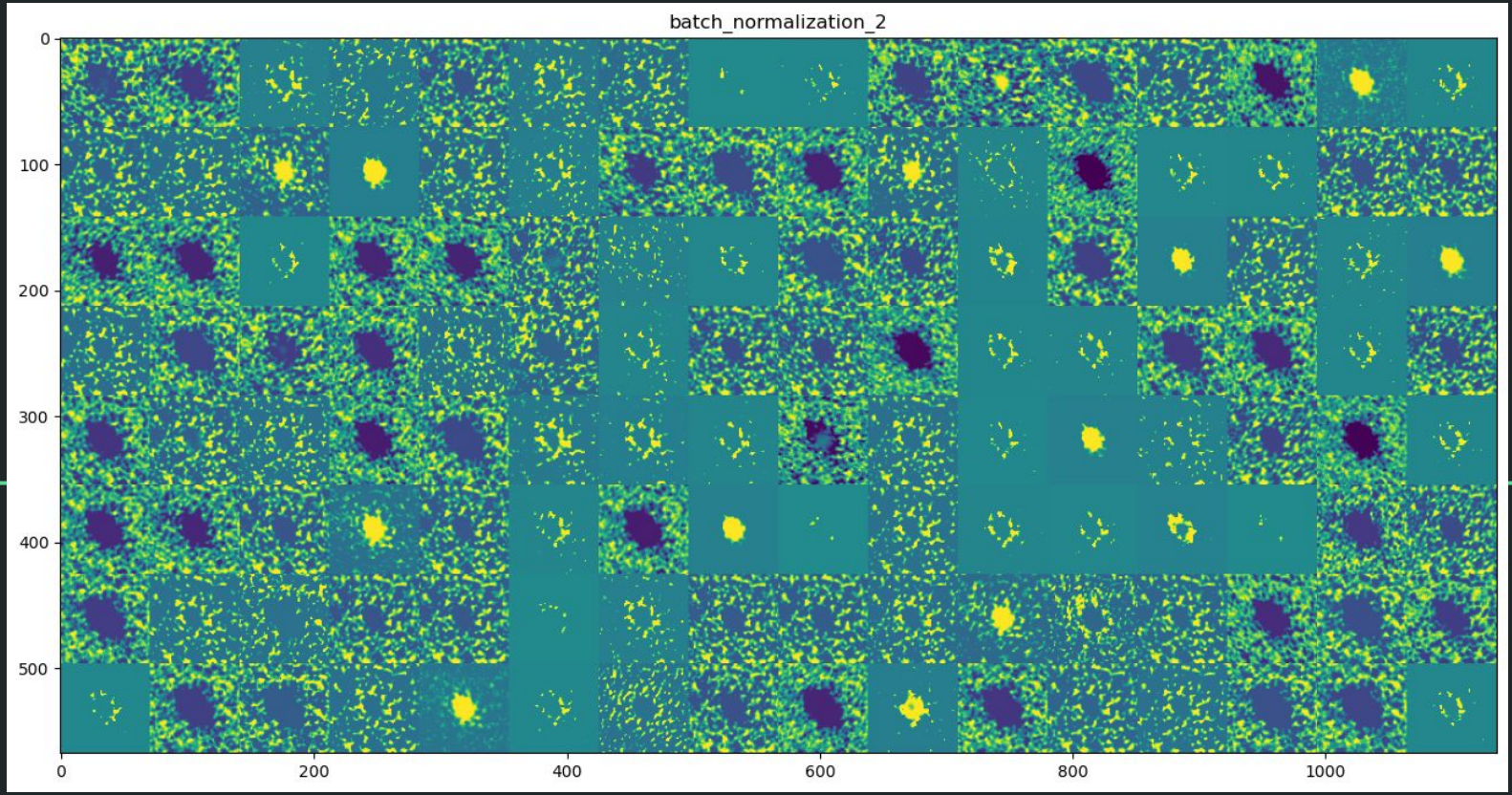

conv2d_4
0
100
200
300
400
500
0
500
1000
1500
2000

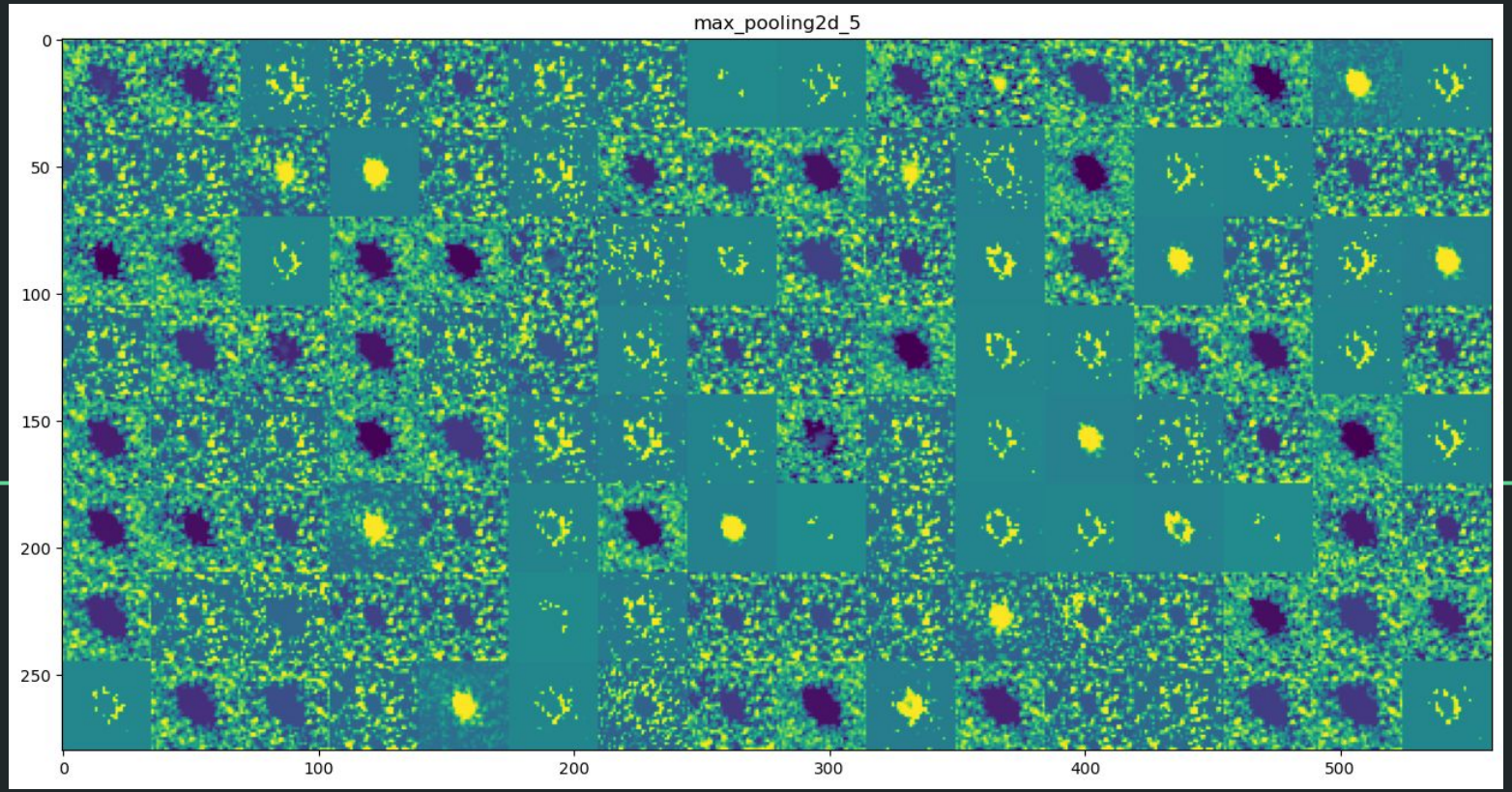

batch_normalization_1
0
100
200
300
400
500
0
500
1000
1500
2000

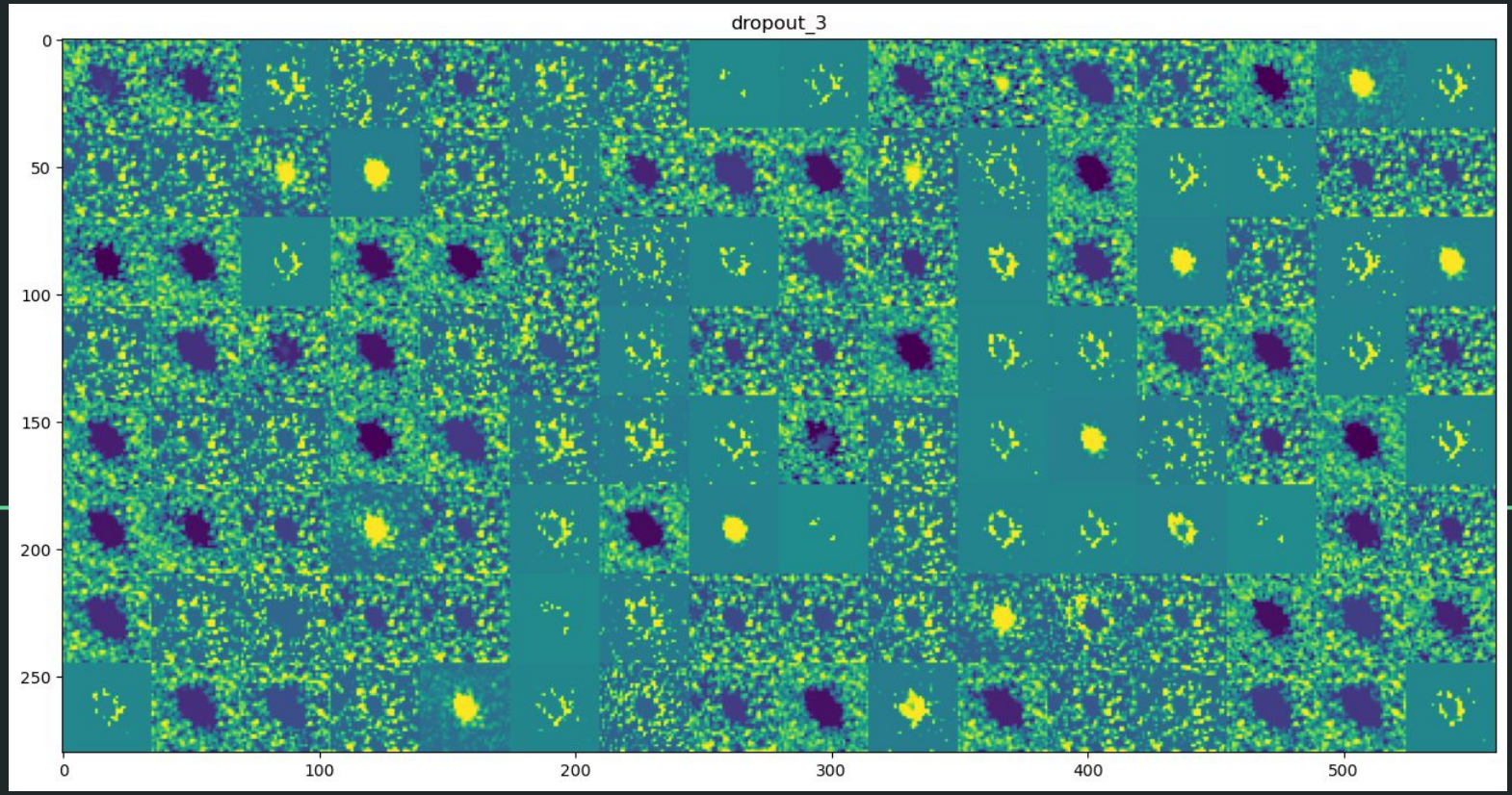

max_pooling2d_4
0
50
100
150
200
250
0
200
400
600
800
1000


dropout_2
0
50
100
150
200
250
0
200
400
600
800
1000


conv2d_5
0
100
200
300
400
500
0
200
400
600
800
1000


batch_normalization_2
0
100
200
300
400
500
0
200
400
600
800
1000


max_pooling2d_5
0
50
100
150
200
250
0
100
200
300
400
500


dropout_3
0
50
100
150
200
250
0
100
200
300
400
500

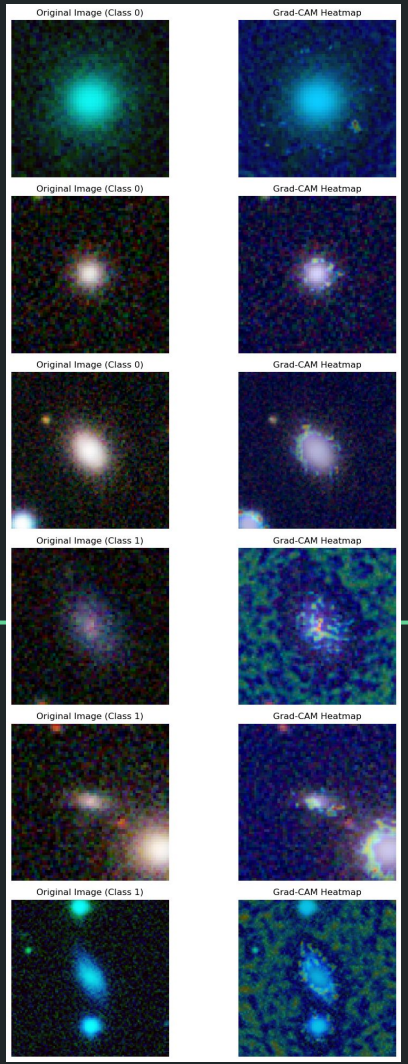
Superimposing the Heatmap on the Input Image

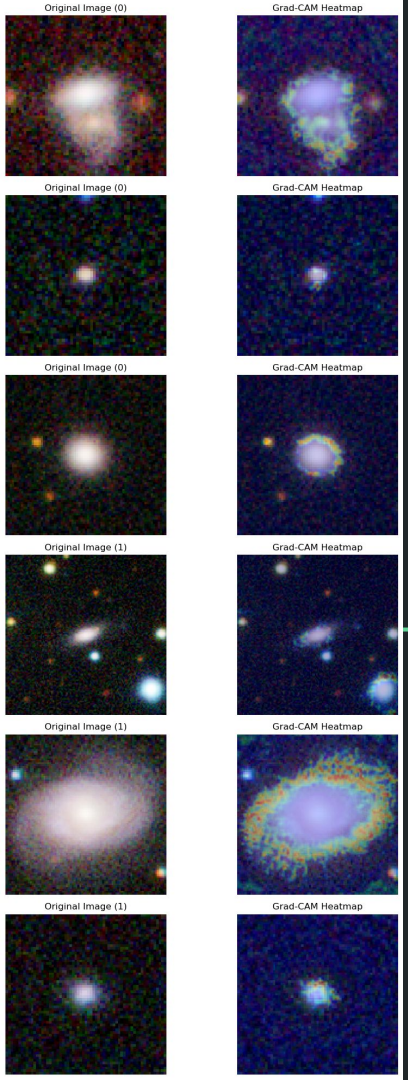
The function `superimpose_heatmap` superimposes the generated Grad-CAM heatmap onto the original input image. This helps in visually interpreting which parts of the image are most significant for the model's prediction.

- **Heatmap Processing**: The heatmap values are scaled and converted to a color map (e.g., "jet").
- **Superimposition**: The color map is resized to match the dimensions of the input image and blended with the input image to highlight the important regions.

Second Try with XAI : Grad-CAM

Grad-CAM is a technique that generates visual explanations for deep learning models, particularly CNNs. It uses the gradients of the target class flowing into the final convolutional layer to produce a coarse localization map, highlighting the important regions in the image. This helps in understanding which parts of the image contribute most to the model's prediction.


Original Image (Class 0)
Grad-CAM Heatmap
Original Image (Class 0)
Grad-CAM Heatmap
Original Image (Class 0)
Grad-CAM Heatmap
Original Image (Class 1)
Grad-CAM Heatmap
Original Image (Class 1)
Grad-CAM Heatmap
Original Image (Class 1)
Grad-CAM Heatmap


Original Image (0)
Grad-CAM Heatmap
Original Image (0)
Grad-CAM Heatmap
Original Image (0)
Grad-CAM Heatmap
Original Image (1)
Grad-CAM Heatmap
Original Image (1)
Grad-CAM Heatmap
Original Image (1)
Grad-CAM Heatmap

Transformers

Feature Extraction

In feature extraction, the pre-trained model's weights are used as they are, up to the output of one or more layers. The steps involved in feature extraction are:

Use Pre-Trained Model: Start with a pre-trained model trained on a large dataset (like ImageNet) with millions of images. Examples include models like VGG, ResNet, or Inception.

Freeze Layers: Freeze all the layers of the pre-trained model. Freezing means that the weights of these layers are not updated during training.

Custom Classifier: Add a new classifier (typically a few dense layers) on top of the pre-trained model. Train only these new layers on your specific dataset (which is smaller and usually different from the original dataset).

Training: Train the entire model (pre-trained layers + new classifier) on your dataset. **Only the weights of the custom classifier are updated; the pre-trained layers remain fixed.**

Fine-Tuning

Fine-tuning goes a step further than feature extraction by unfreezing some (or all) of the layers of the pre-trained model and training them alongside the new classifier. The steps involved in fine-tuning are:

Use Pre-Trained Model: Start with a pre-trained model trained on a large dataset.

Unfreeze Layers: Optionally, unfreeze some or all of the layers of the pre-trained model. This allows the weights of these layers to be updated during training.

Custom Classifier: Add a new classifier on top of the pre-trained model.

Training: Train the entire model (both the pre-trained layers and the new classifier) on your dataset. During training, **both the weights of the pre-trained layers and the new classifier are updated.**

```python
from tensorflow.keras.applications import VGG16

vgg16_base = VGG16(weights='imagenet', include_top=False, input_shape=(300, 300,
3))
```

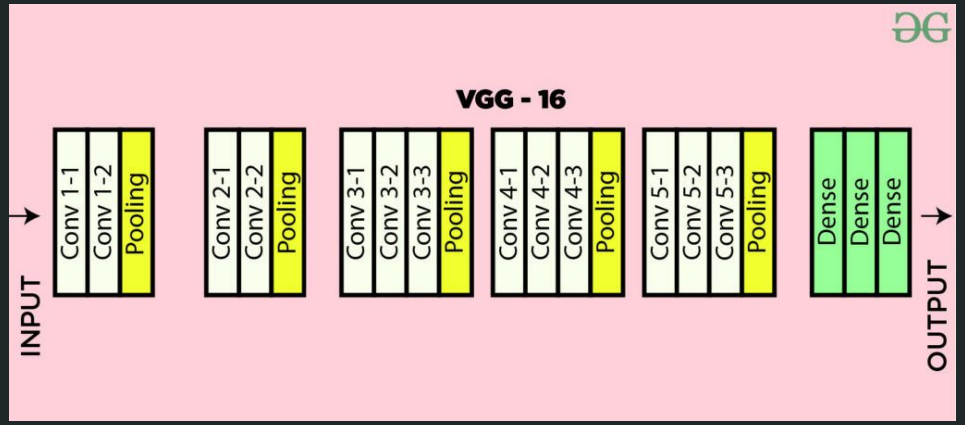
```python
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dense, BatchNormalization,
Dropout
model = Sequential([
    vgg16_base,
    Flatten(),
    Dense(512, activation='relu'),
    BatchNormalization(),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])
```

VGG16 : We employed the VGG16 model, which has been pre-trained on the **ImageNet** dataset, to classify images into two categories: Elliptical and Other. Leveraging the principle of **feature extraction**, we capitalize on the rich and generalized representations learned by VGG16 and adapt them to our specific classification task.

VGG16 Validation Loss: 0.4054112732410431

VGG16 Validation Accuracy: 0.7949152588844299

	precision	recall	f1-score	support
Elliptical	0.91	0.65	0.76	584
Other	0.73	0.93	0.82	596
accuracy			0.79	1180
macro avg	0.82	0.79	0.79	1180
weighted avg	0.82	0.79	0.79	1180



The validation accuracy of approximately 79.49% suggests that the VGG16 model performed reasonably well on the classification task. However, there is room for improvement, especially in terms of identifying elliptical galaxies, where the recall was relatively lower.

Further optimization strategies, such as fine-tuning the model or using different architectures, could potentially enhance the model's performance.

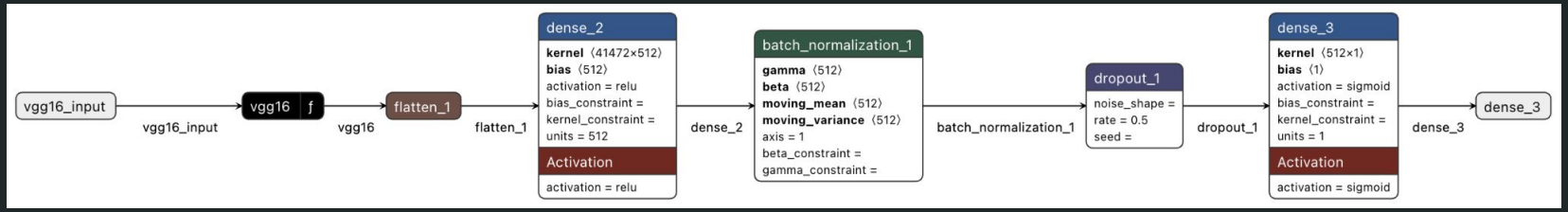
Overall, the VGG16 model demonstrated promising results, laying a solid foundation for further refinement and experimentation in galaxy classification tasks.

Fine Tuning

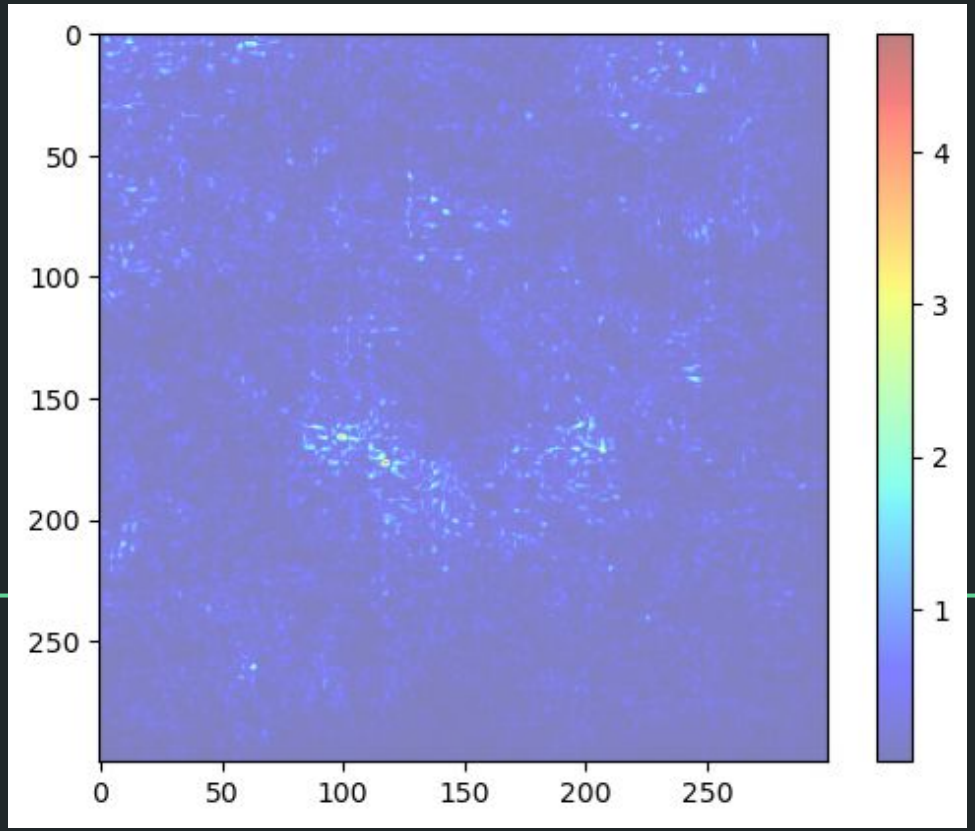
VGG16 Validation Loss: 0.2714722454547882
VGG16 Validation Accuracy: 0.8855932354927063

37/37 [==============================] - 140s 4s/step

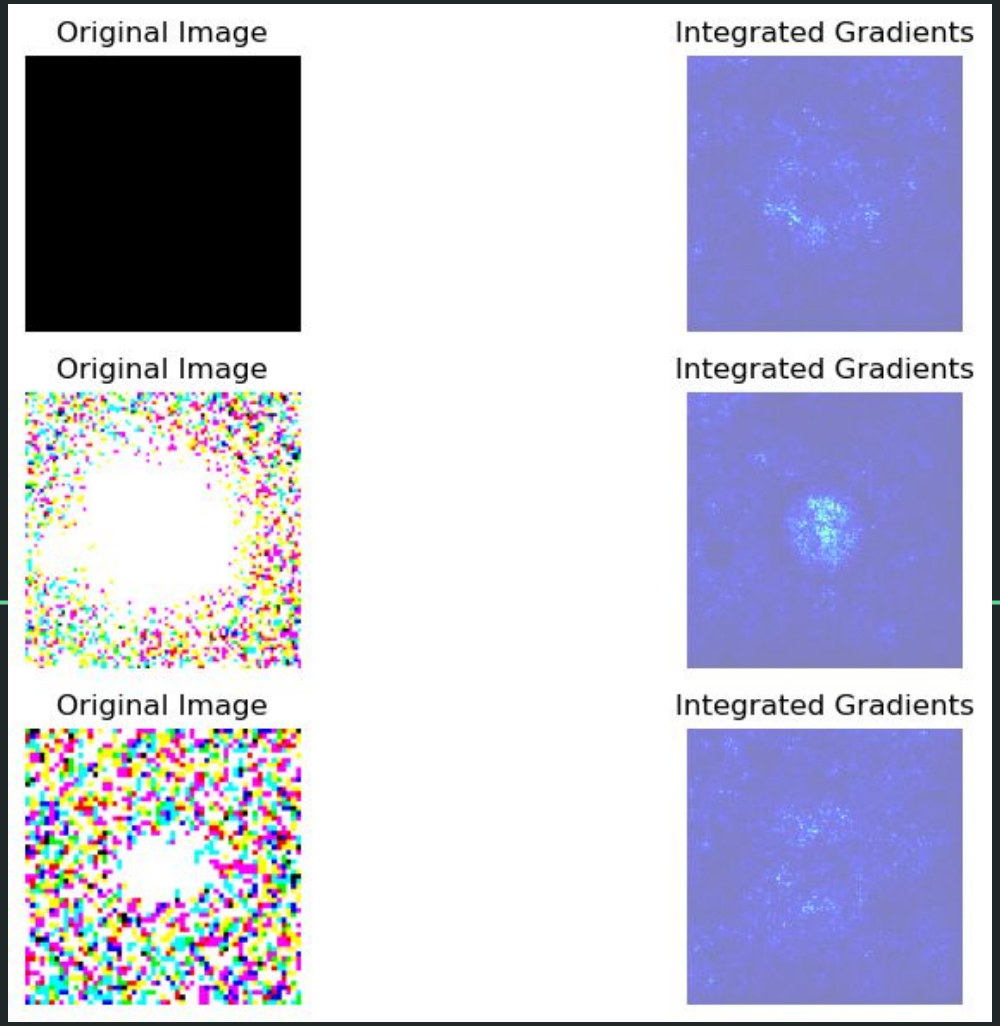
	precision	recall	f1-score	support
Elliptical	0.93	0.83	0.88	584
Other	0.85	0.94	0.89	596
accuracy			0.89	1180
macro avg	0.89	0.89	0.89	1180
weighted avg	0.89	0.89	0.89	1180



Integrated Gradients



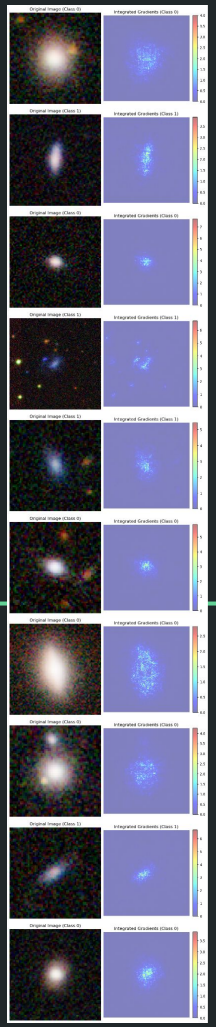
Integrated gradients is an interpretability technique used to **understand the contribution of each pixel in** an input image towards the model's prediction. It calculates the gradients of the model's output with respect to the input image across a series of interpolated steps between a baseline (usually an all-black image) and the input image.



Preprocessing of Images for Display:

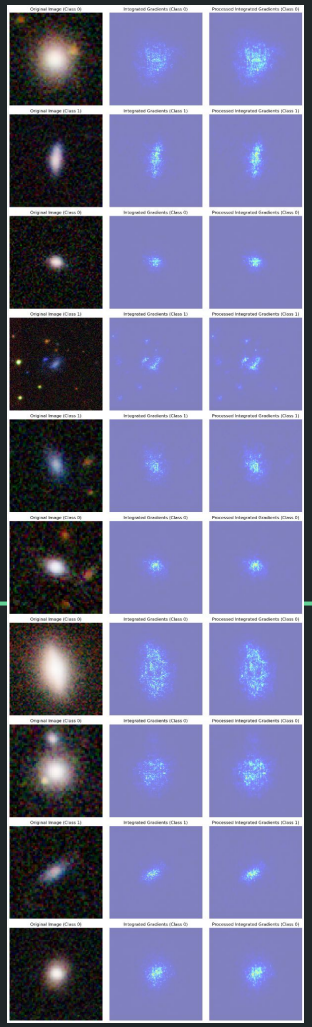
The images in the left column appeared incorrect because they were being preprocessed using **tf.keras.applications.vgg16.preprocess_input.** This preprocessing step alters the images to be suitable for the VGG16 model but makes them look unusual when displayed directly.
Solution: Display the original images without preprocessing.
Normalization of Integrated Gradients:

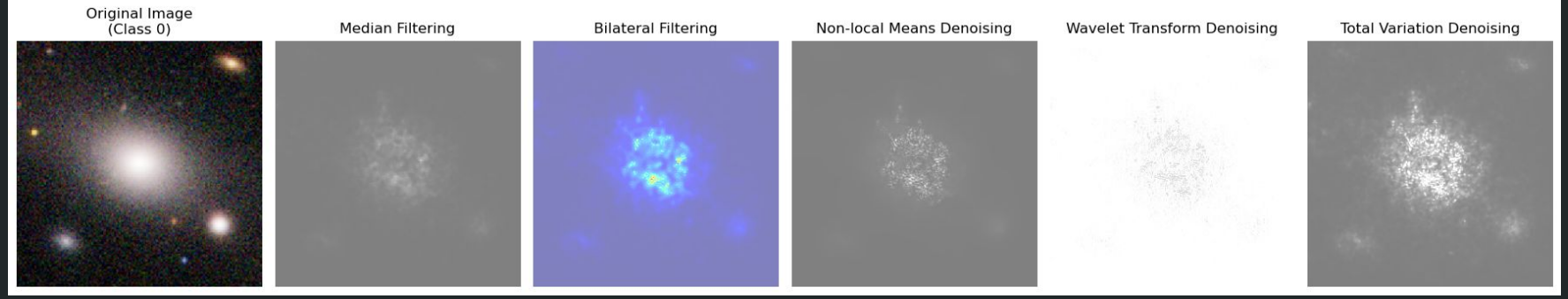
The integrated gradients might not have been normalized properly for visualization, leading to unclear or uninformative visualizations.
Solution: Ensure that the integrated gradients are normalized appropriately before plotting.



Noise Reduction

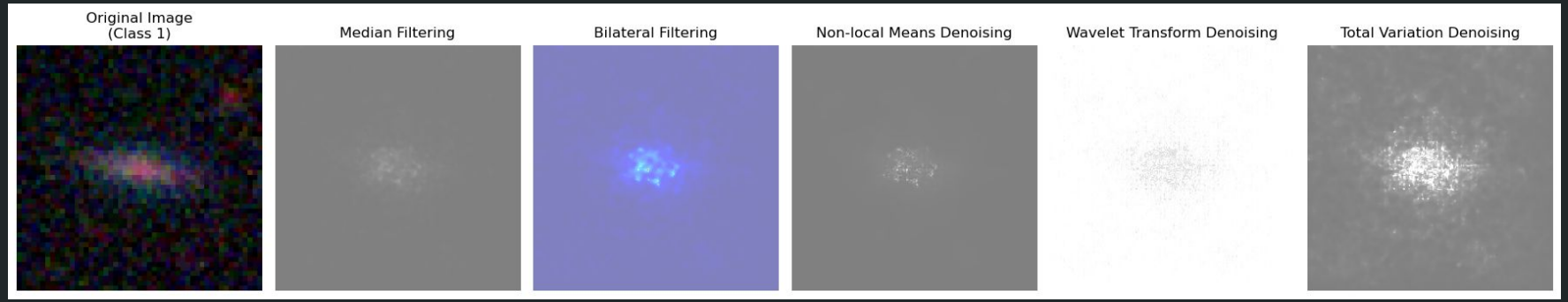
- **Effect of Noise Reduction:**
  - The noise reduction using **Gaussian blur** has successfully smoothed out the noise in the integrated gradients explanations.
  - The key areas of importance in the images are still highlighted, but the noise is significantly reduced, making it easier to interpret the explanations.

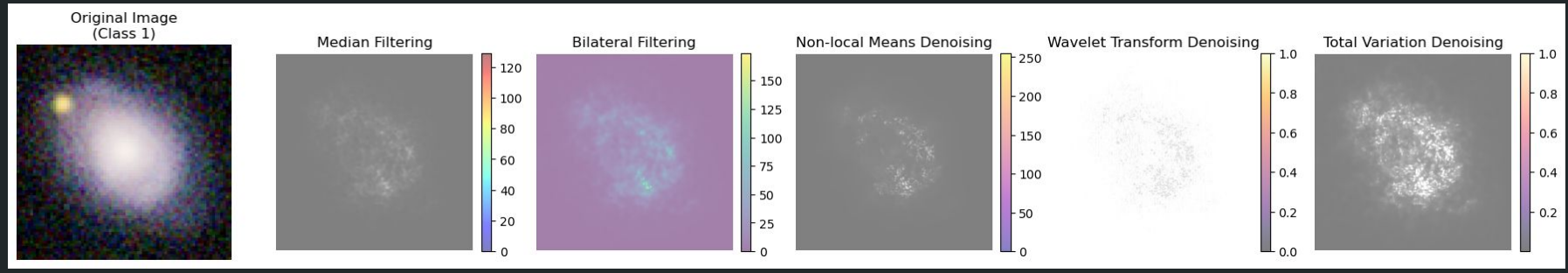


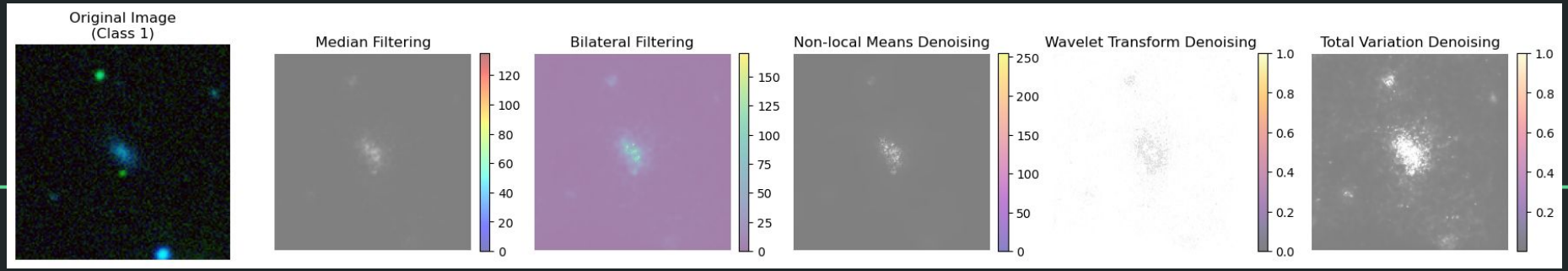


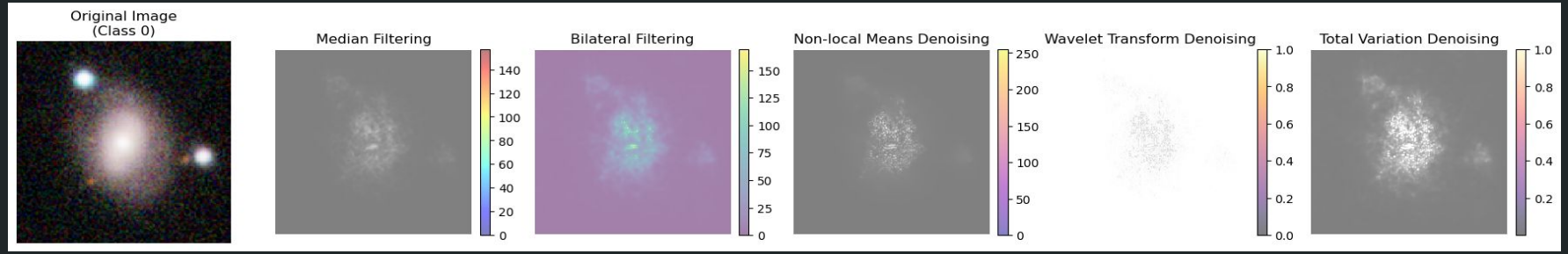
Noise Reduction Techniques

Median Filtering: Effective for removing salt and pepper noise, with moderate preservation of edges.
Bilateral Filtering: Excellent at reducing noise while preserving edges and details.
Non-local Means Denoising: Superior in maintaining textures and fine details while reducing noise.
Wavelet Transform Denoising: Good at isolating and removing high-frequency noise while preserving important features.
Total Variation Denoising: Effective in reducing noise while maintaining sharp edges and fine details.




Original Image
(Class 1)
Median Filtering
Bilateral Filtering
Non-local Means Denoising
Wavelet Transform Denoising
Total Variation Denoising


Original Image
(Class 1)
Median Filtering
Bilateral Filtering
Non-local Means Denoising
Wavelet Transform Denoising
Total Variation Denoising


Original Image
(Class 0)
Median Filtering
Bilateral Filtering
Non-local Means Denoising
Wavelet Transform Denoising
Total Variation Denoising

1. **Median Filtering:**
    - Reduces "salt and pepper" noise by replacing each pixel value with the median value of the neighboring pixels.
    - Effective for removing small-scale noise but can blur edges slightly.
2. **Bilateral Filtering:**
    - Reduces noise while preserving edges by considering both spatial proximity and pixel value differences.
    - Maintains edge sharpness better than median filtering.
3. **Non-local Means Denoising:**
    - Reduces noise by averaging pixels with similar intensity patterns, even if they are far apart.
    - Preserves textures and fine details.

---

4. **Wavelet Denoising:**
    - Reduces noise by applying wavelet transform, thresholding the wavelet coefficients, and then reconstructing the image.
    - Effectively removes high-frequency noise while preserving important features.
5. **Total Variation Denoising:**
    - Reduces noise by minimizing the total variation of the image, promoting piecewise-smooth solutions.
    - Preserves edges and reduces noise uniformly across the image.

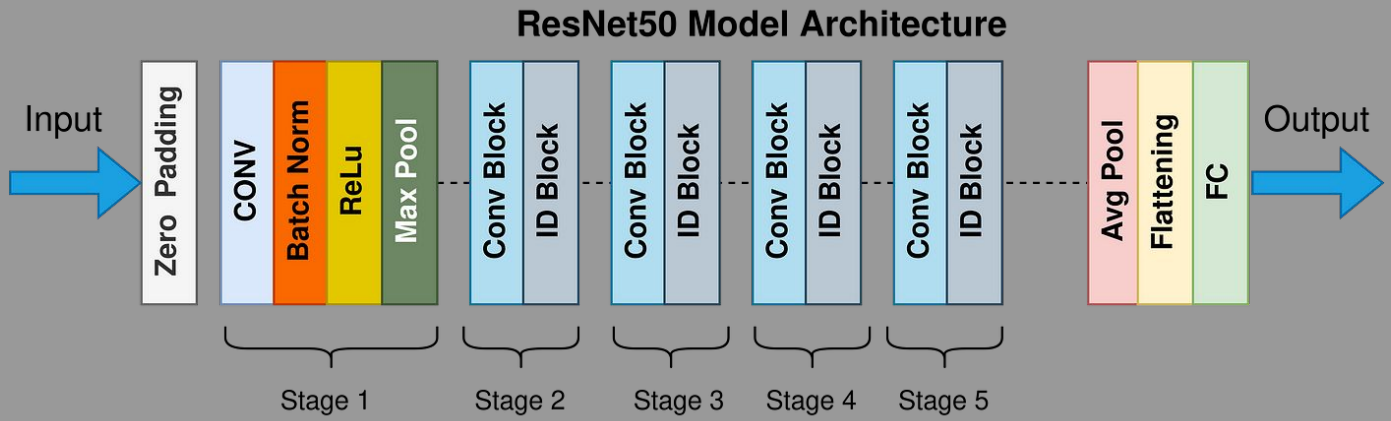
Summary of Noise Reduction Techniques

Complexity:

Finetuned Model: More complex than both the custom CNN and the VGG16 model with feature extraction only.
Increased Complexity: Fine-tuning increases complexity by training more layers, which also increases the number of trainable parameters.
Interpretability:
Custom CNN: Easier to interpret with tools like Grad-CAM because it is simpler and designed from scratch for our specific task.

VGG16 (Feature Extraction + Fine-Tuning): Harder to interpret. Feature extraction alone is somewhat easier but still complex due to pre-trained layers. Fine-tuning adds another layer of complexity, making interpretability harder.

Summary of Interpretability and Complexity



RESNET50

. Transfer Learning:
Definition: Transfer learning is a machine learning technique where a model developed for a particular task is reused as the starting point for a model on a second task. This is particularly useful when the second task has limited training data.
Why Use It?: Training deep learning models from scratch requires a large amount of data and computational resources. Transfer learning allows leveraging pre-trained models, which have already learned features from a large dataset, to improve performance on a new task with less data and computation.

Initial ResNet50 Model Training (Epochs 1-10):

Train Accuracy: Improved from ~80% to 85%
Validation Accuracy: Started around 85%, fluctuating but generally improving to ~88-89%
Train Loss: Decreased from ~0.50 to ~0.33
Validation Loss: Decreased from 0.33 to ~0.27, with some fluctuations

---

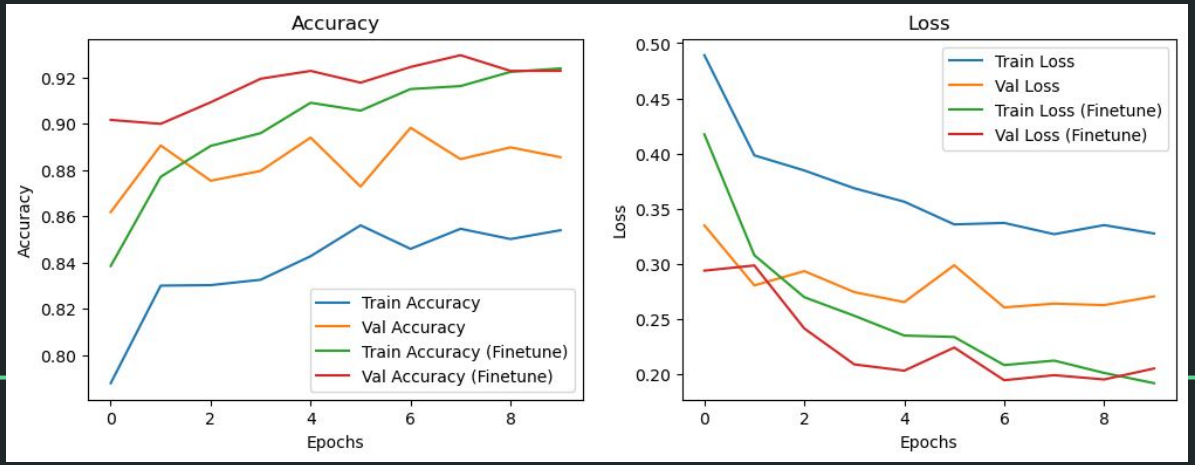
Fine-tuned ResNet50 Model Training (Epochs 1-10):

Train Accuracy: Improved from ~84% to 92-93%
Validation Accuracy: Improved significantly to ~92-93%
Train Loss: Decreased from ~0.42 to ~0.19
Validation Loss: Decreased consistently to ~0.20

ResNet50 Fine-tuned Validation Loss: 0.20537687838077545
ResNet50 Fine-tuned Validation Accuracy: 0.9228813648223877

37/37 [==============================] - 48s 1s/step

	precision	recall	f1-score	support
Elliptical	0.91	0.93	0.92	567
Other	0.94	0.91	0.92	613
accuracy			0.92	1180
macro avg	0.92	0.92	0.92	1180
weighted avg	0.92	0.92	0.92	1180

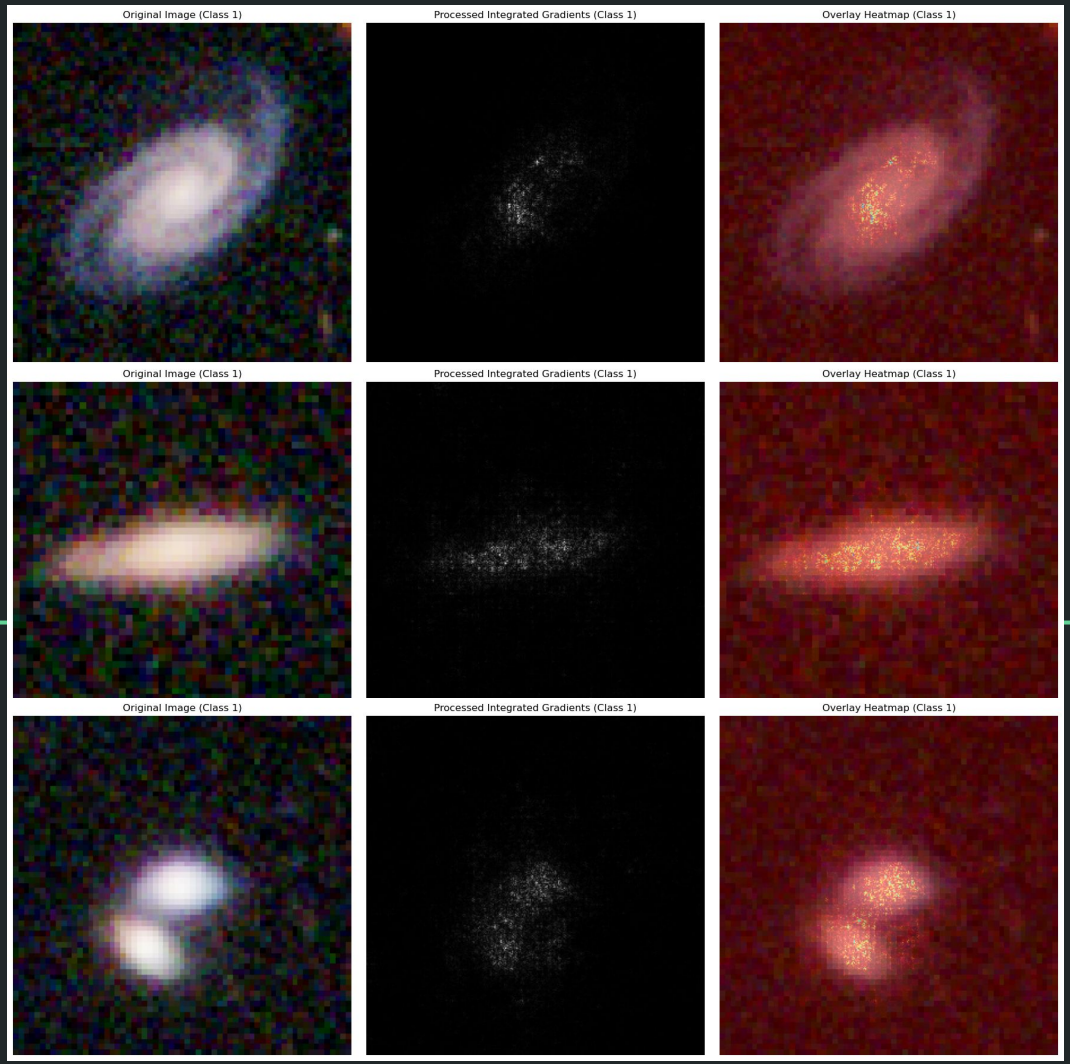


ResNet50 Architecture:

- **Residual Blocks**: ResNet50 consists of residual blocks, which have skip connections that bypass one or more layers. This helps mitigate the vanishing gradient problem, making it possible to train very deep networks.
- **Architecture**: ResNet50 has 50 layers, including convolutional layers, batch normalization layers, activation layers, and fully connected layers. The key innovation is the residual block, where the input of a block is added to its output, allowing gradients to flow through the network more easily.

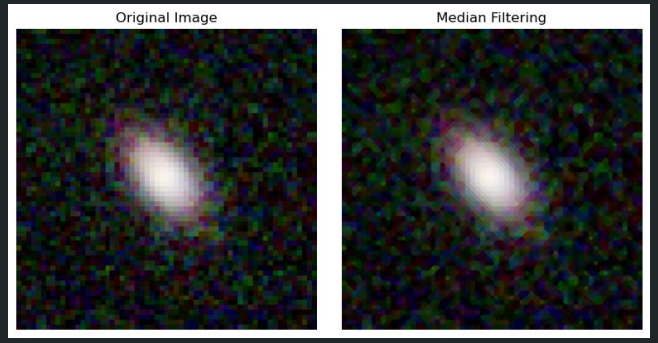
Integrated Gradients for ResNet50

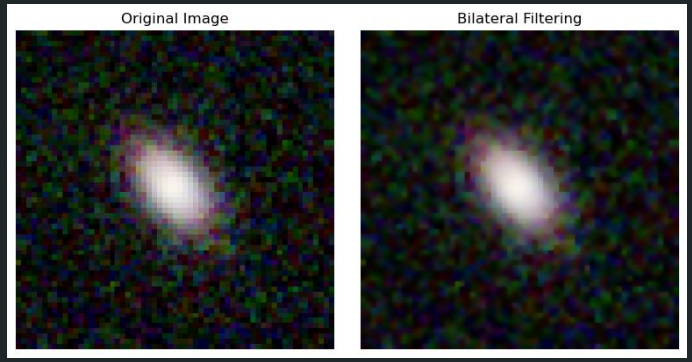
The overlay heatmap is created by applying a color map (often using the 'jet' colormap) to the processed IG images (`ig_explanations_processed`). The color map enhances the visibility of the gradients.

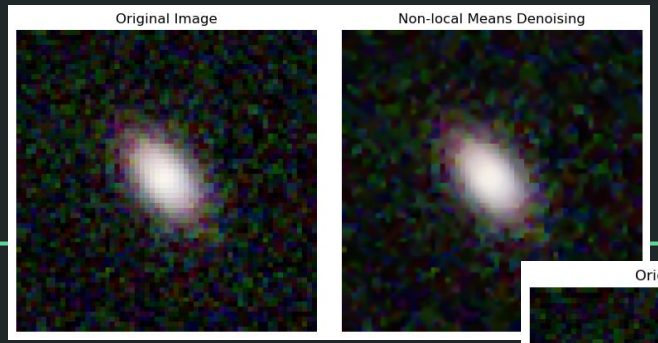


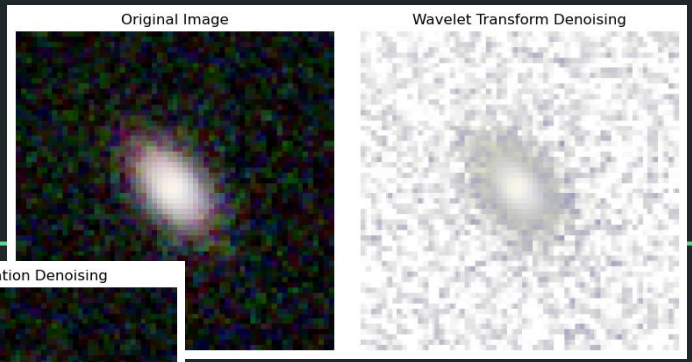
Denoising Dataset and trying to Apply RESNET50

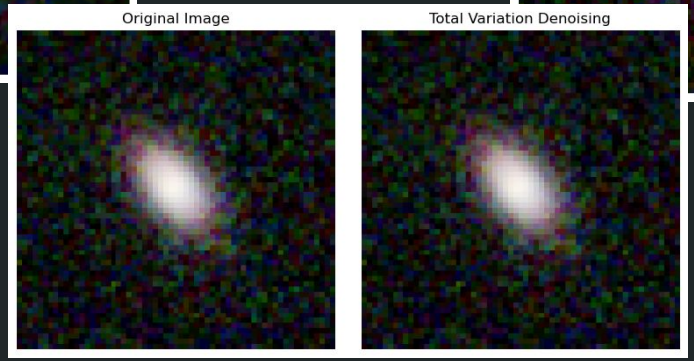
Task 1


Original Image
Median Filtering


Original Image
Bilateral Filtering


Original Image
Non-local Means Denoising


Original Image
Wavelet Transform Denoising


Original Image
Total Variation Denoising

Task 1

Bilateral_Filtering - ResNet50 Fine-tuned Validation Loss: 0.1947125345468521
Bilateral_Filtering - ResNet50 Fine-tuned Validation Accuracy: 0.9254237413406372
37/37 [==============================] - 47s 1s/step
Classification Report for Bilateral_Filtering:

	precision	recall	f1-score	support
Elliptical	0.91	0.93	0.92	567
Other	0.94	0.92	0.93	613
accuracy			0.93	1180
macro avg	0.93	0.93	0.93	1180
weighted avg	0.93	0.93	0.93	1180

Median_Filtering - ResNet50 Fine-tuned Validation Loss: 0.2047974169254303
Median_Filtering - ResNet50 Fine-tuned Validation Accuracy: 0.9245762825012207
37/37 [==============================] - 52s 1s/step
Classification Report for Median_Filtering:

	precision	recall	f1-score	support
Elliptical	0.93	0.92	0.92	567
Other	0.92	0.93	0.93	613
accuracy			0.92	1180
macro avg	0.92	0.92	0.92	1180
weighted avg	0.92	0.92	0.92	1180

Non-local Means Denoising - ResNet50 Fine-tuned Validation Loss: 0.216127589344978 33
Non-local Means Denoising - ResNet50 Fine-tuned Validation Accuracy: 0.9211864471435547
37/37 [==============================] - 48s 1s/step
Classification Report for Non-local Means Denoising:

	precision	recall	f1-score	support
Elliptical	0.89	0.95	0.92	567
Other	0.95	0.90	0.92	613
accuracy			0.92	1180
macro avg	0.92	0.92	0.92	1180
weighted avg	0.92	0.92	0.92	1180

Total_Variation_Denoising - ResNet50 Fine-tuned Validation Loss:
0.22398154437541962
Total_Variation_Denoising - ResNet50 Fine-tuned Validation Accuracy:
0.9101694822311401
37/37 [==============================] - 47s 1s/step
Classification Report for Total_Variation_Denoising:

	precision	recall	f1-score	support
Elliptical	0.94	0.87	0.90	567
Other	0.89	0.95	0.92	613
accuracy			0.91	1180
macro avg	0.91	0.91	0.91	1180
weighted avg	0.91	0.91	0.91	1180

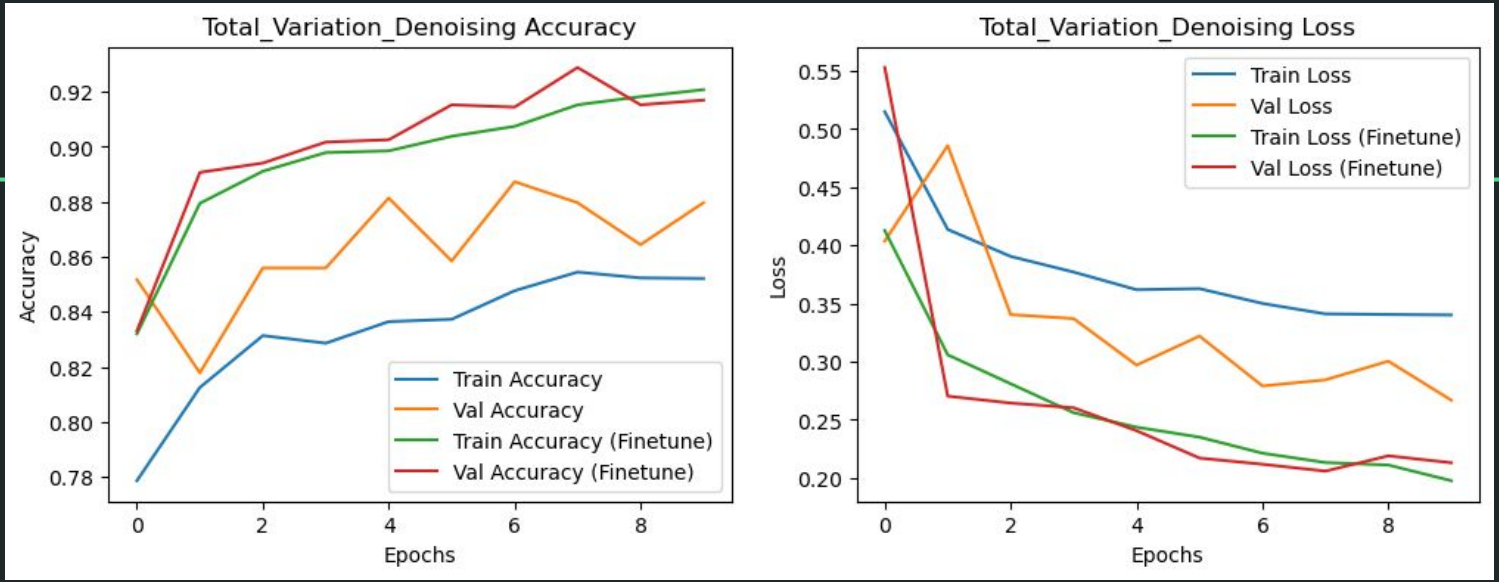
Wavelet_Transform_Denoising - ResNet50 Fine-tuned Validation Loss: 0.20959553122520447
Wavelet_Transform_Denoising - ResNet50 Fine-tuned Validation Accuracy: 0.9067796468734741
37/37 [==============================] - 57s 2s/step
Classification Report for Wavelet_Transform_Denoising:

	precision	recall	f1-score	support
Elliptical	0.93	0.87	0.90	567
Other	0.89	0.94	0.91	613
accuracy			0.91	1180
macro avg	0.91	0.91	0.91	1180
weighted avg	0.91	0.91	0.91	1180

Precision: The ratio of correctly predicted positive observations to the total predicted positives. It tells us how many of the predicted positive cases were actually positive.

Support: The number of actual occurrences of the class in the dataset.

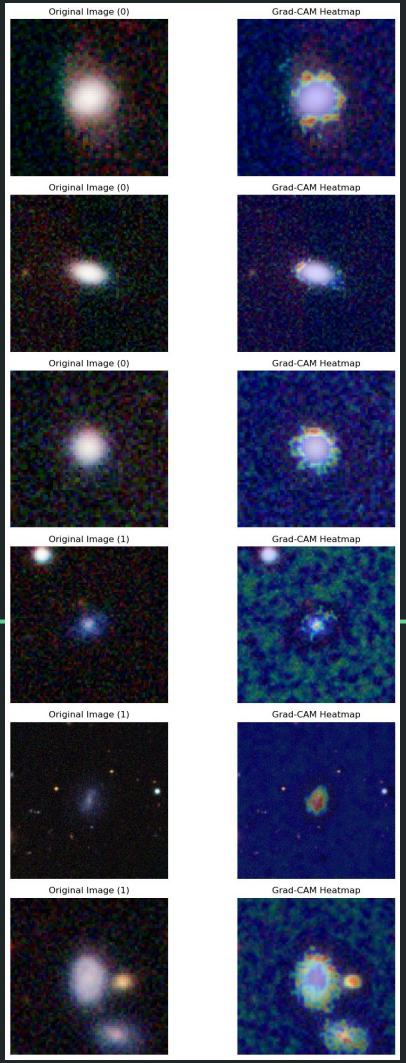
Recall (or Sensitivity): The ratio of correctly predicted positive observations to all the observations in the actual class. It tells us how many of the actual positive cases were correctly predicted.



Different XAI techniques for same Sample

Task 2

Grad CAM for CNN

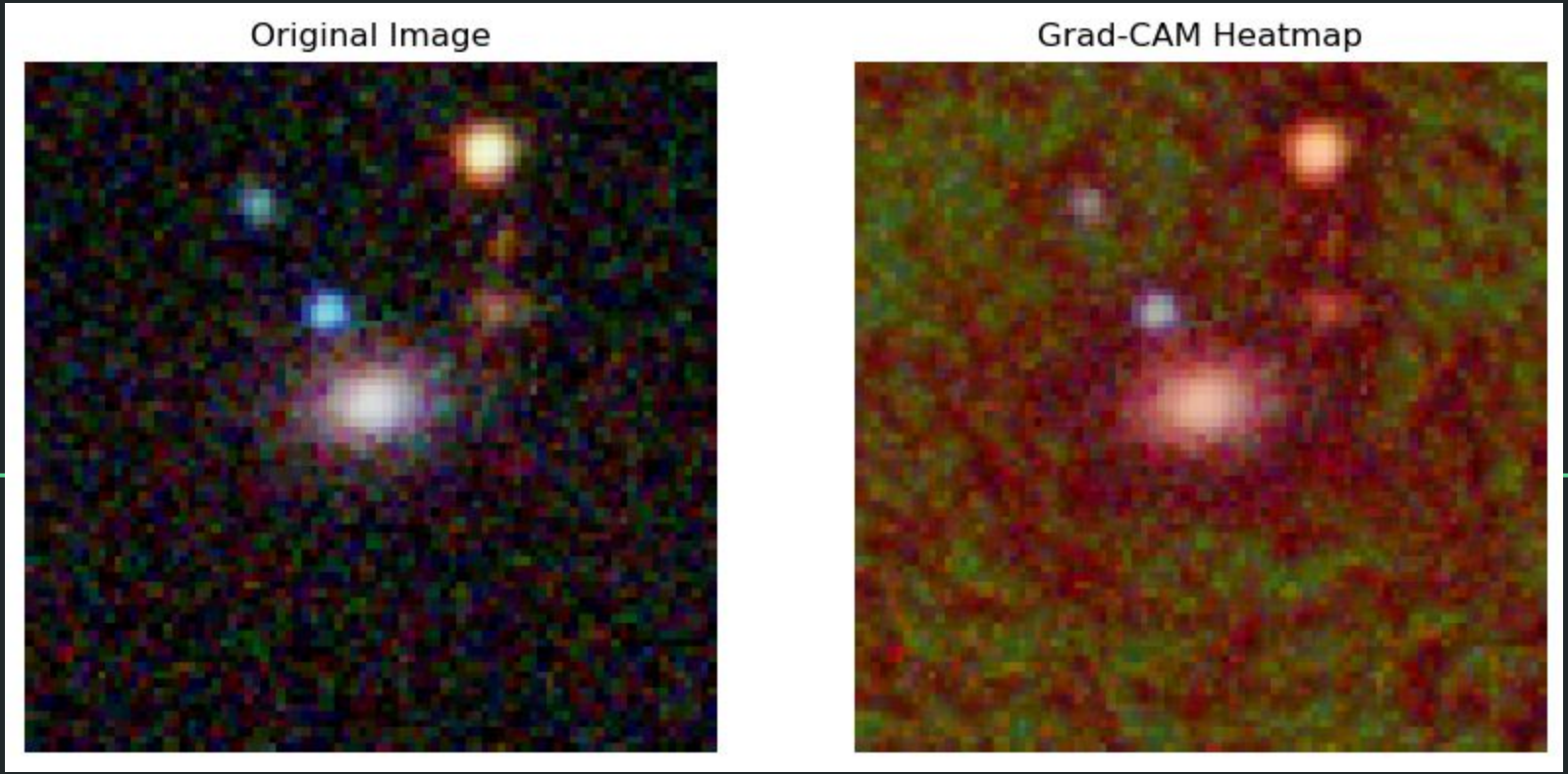


Grad-CAM Heatmap:

- Grad-CAM stands for Gradient-weighted Class Activation Mapping.
- It highlights the regions of the image that the model considered important for making its prediction.
- The heatmap is generated by computing the gradients of the target class output with respect to the last convolutional layer's output.
- The importance values are visualized using a color map (jet colormap in this case).

- **Darker regions represent lower importance, while brighter regions represent higher importance.**
- **This heatmap helps in understanding which parts of the image contributed most to the model's decision.**

sample_image_index = 310

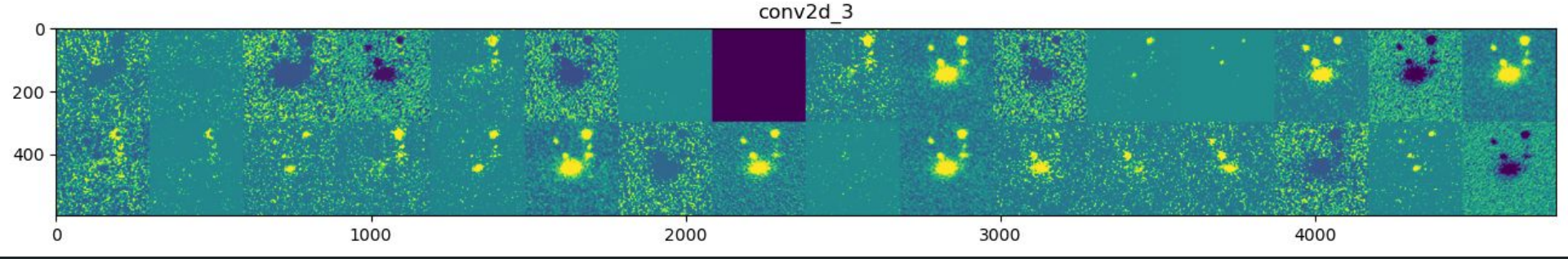


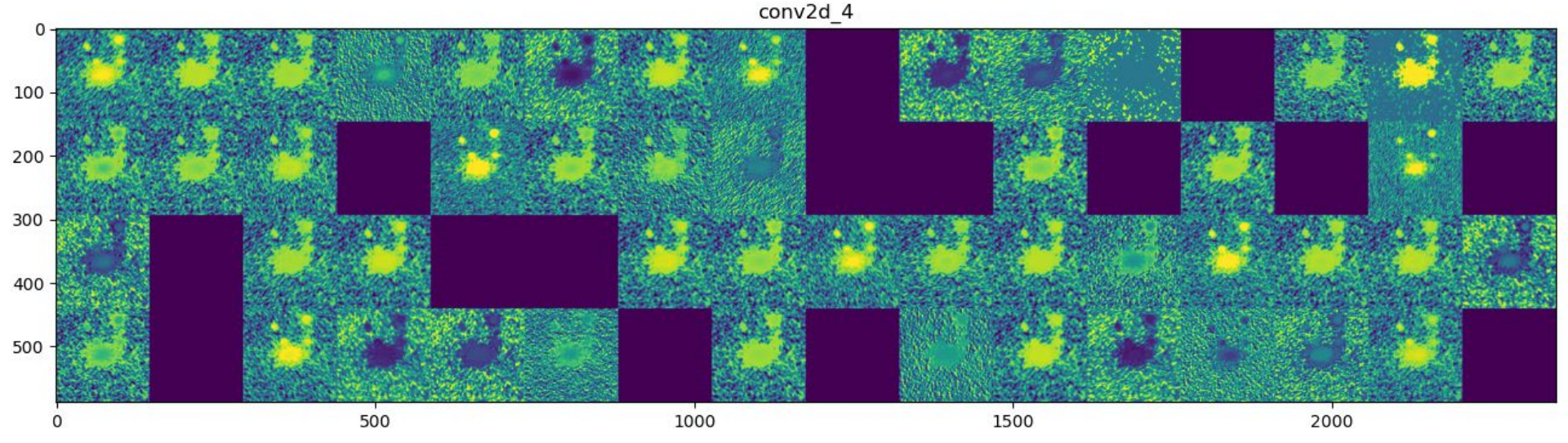
Visualize Activations:

- **This plot visualizes the activations of convolutional layers in the model.**
- Each subplot represents the activations of a different convolutional layer.
- The activations are visualized as a grid of images, where each image represents the response of a single filter in the convolutional layer.

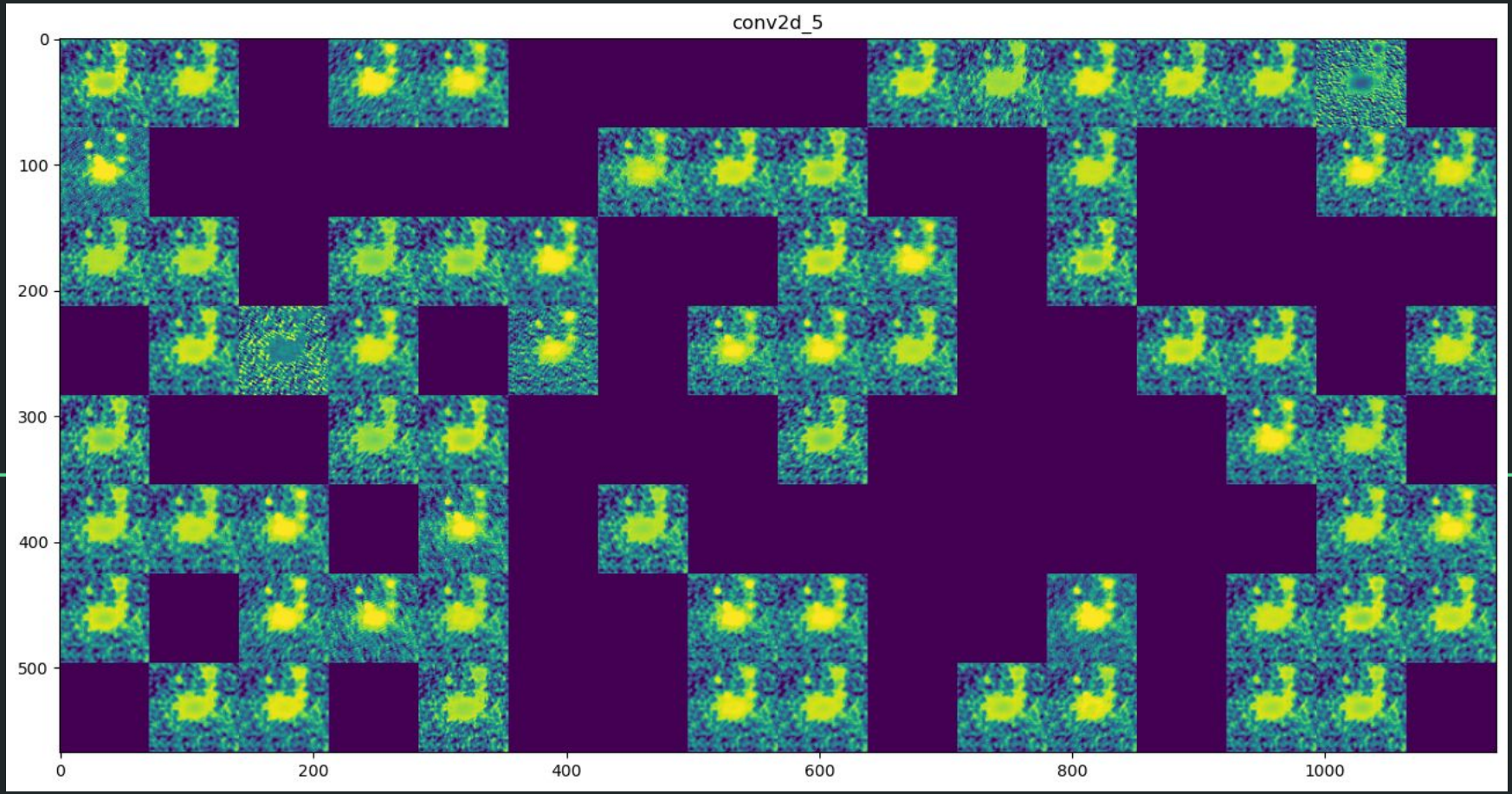
- **This visualization helps in understanding how the input image is transformed and processed at different stages of the model.**

sample_image_index = 310





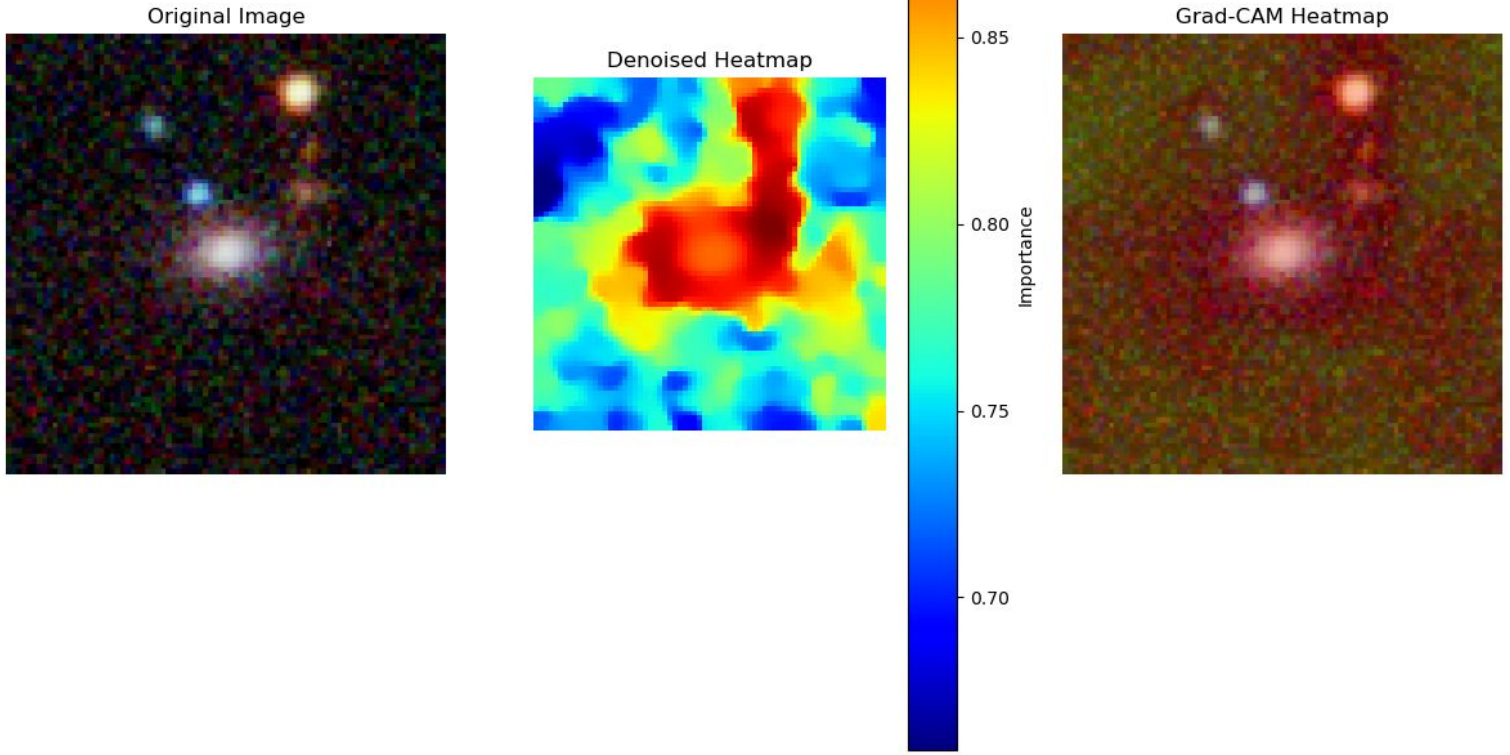
sample_image_index = 310



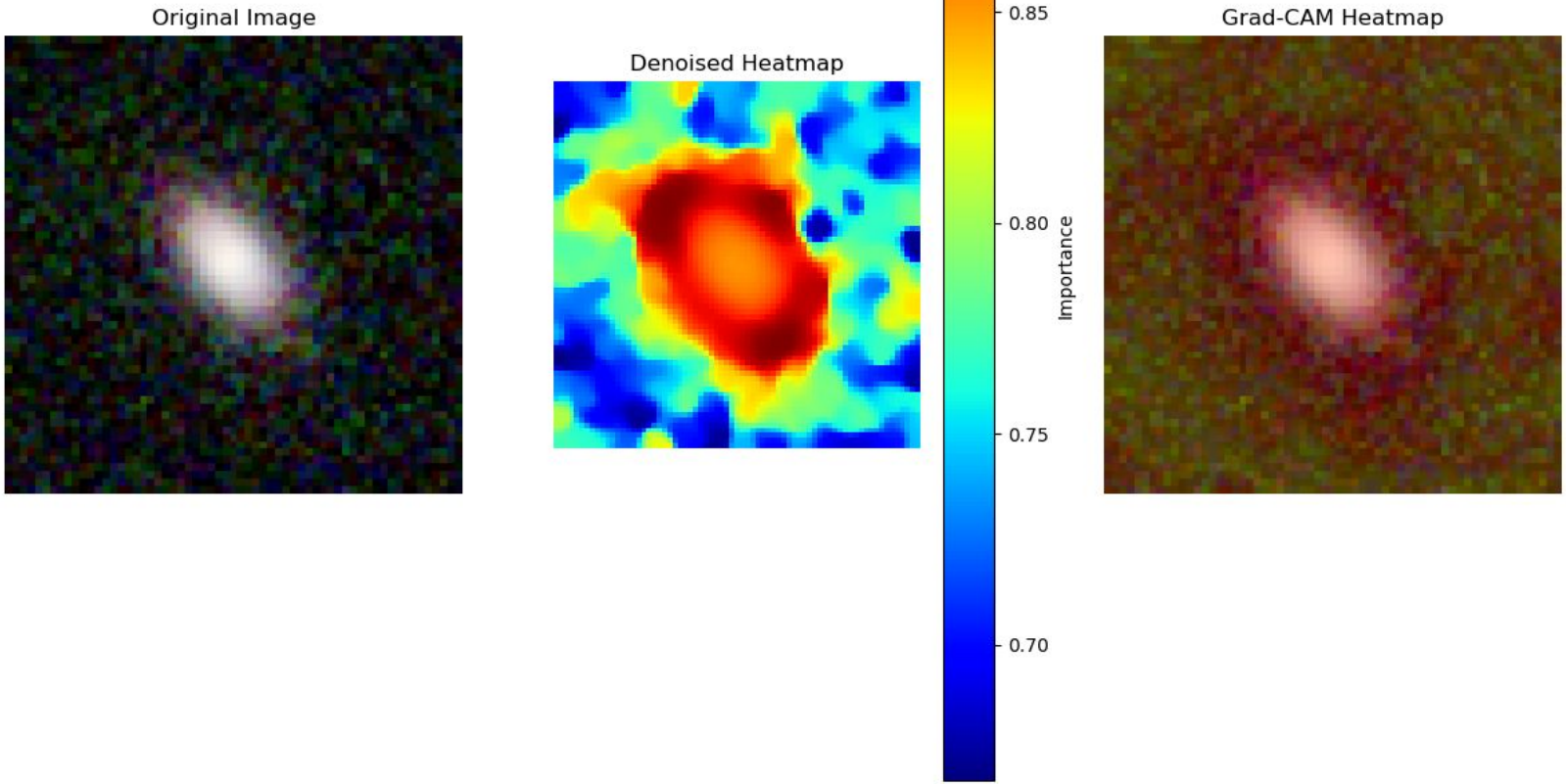
Denoised Heatmap:

- **This plot shows the Grad-CAM heatmap after applying Total Variation Denoising.**
- Total Variation Denoising is a technique used to smooth out the heatmap and remove noise, making it easier to interpret.
- It helps in obtaining cleaner and more visually appealing heatmaps

sample_image_index = 310



sample_image_index = 0

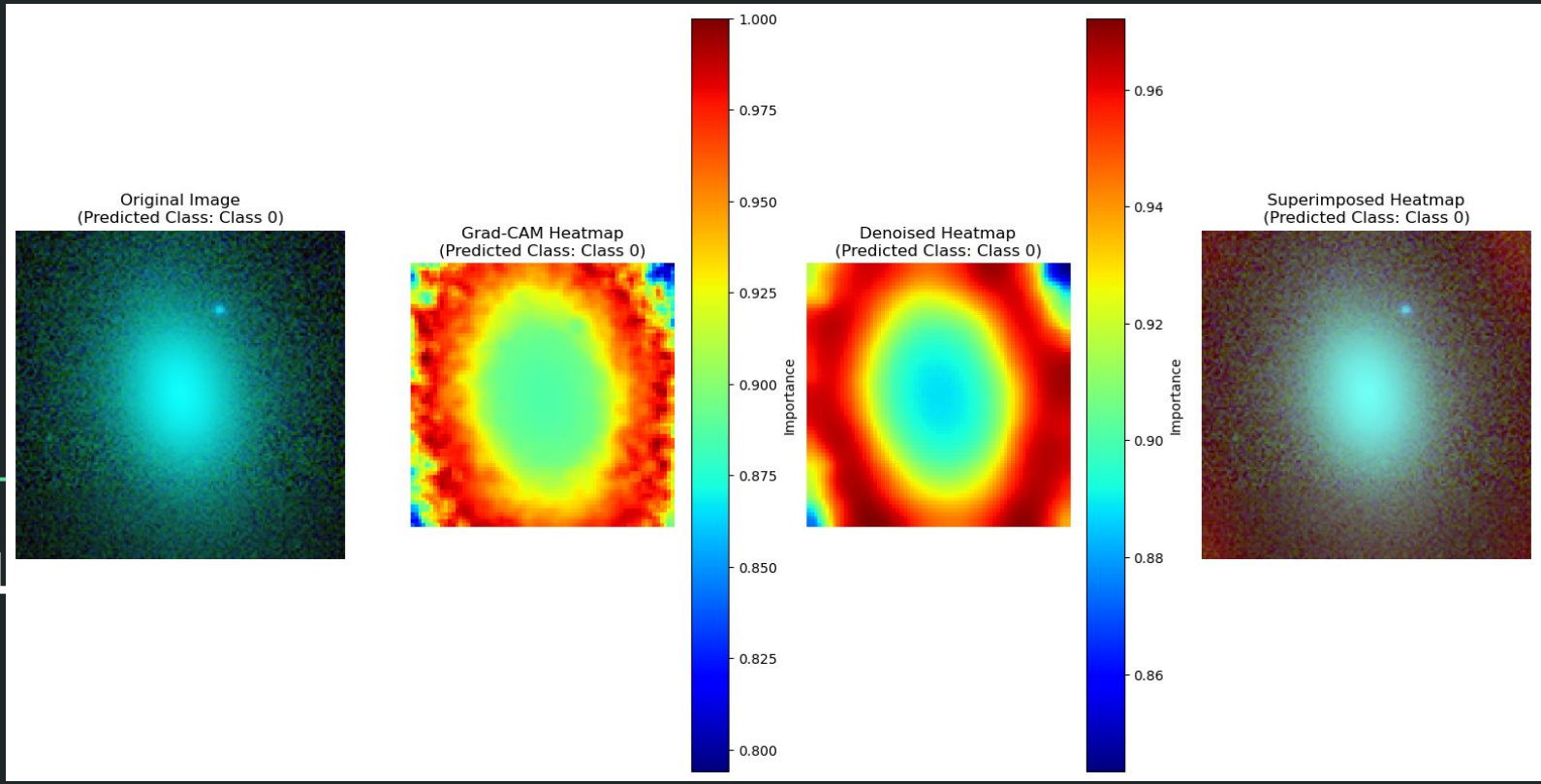


1. **Grad-CAM Heatmap**:
    - Grad-CAM stands for Gradient-weighted Class Activation Mapping.
    - It highlights the regions of the image that the model considered important for making its prediction.
    - The heatmap is generated by computing the gradients of the target class output with respect to the last convolutional layer's output.
    - The importance values are visualized using a color map (jet colormap in this case).
    - Darker regions represent lower importance, while brighter regions represent higher importance.
    - This heatmap helps in understanding which parts of the image contributed most to the model's decision.

2. **Denoised Heatmap**:
    - **This plot shows the Grad-CAM heatmap after applying Total Variation Denoising.**
    - Total Variation Denoising is a technique used to smooth out the heatmap and remove noise, making it easier to interpret.
    - It helps in obtaining cleaner and more visually appealing heatmaps.

---

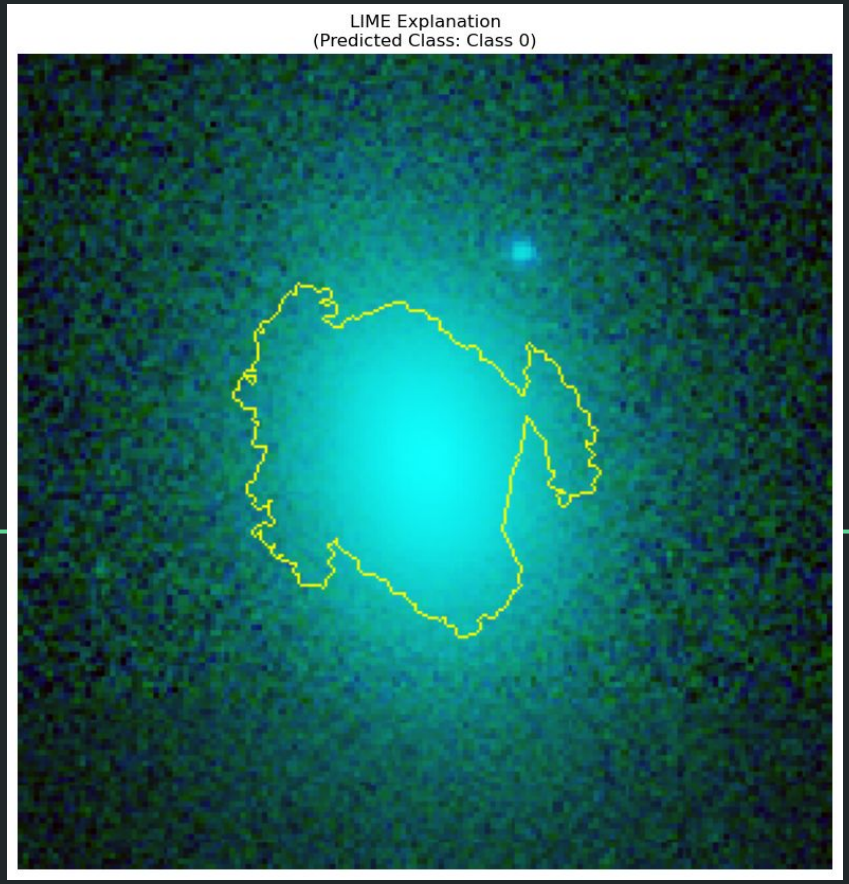
3. **Superimposed Heatmap**:
    - **This plot overlays the denoised heatmap onto the original image.**
    - The transparency of the heatmap is adjusted to make it **semi-transparent,** allowing visualization of both the heatmap and the original image simultaneously.
    - This visualization helps in understanding how the important regions identified by the model correspond to the actual features in the image.

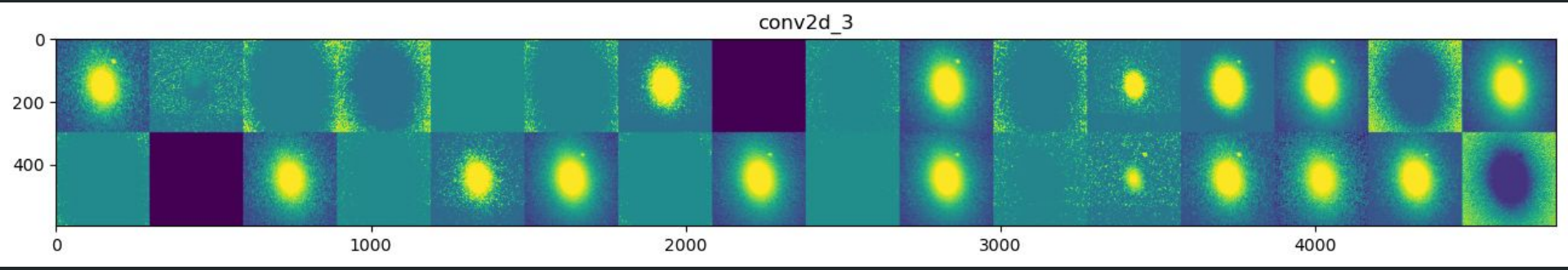

Original Image
(Predicted Class: Class 0)
Grad-CAM Heatmap
(Predicted Class: Class 0)
1.000
0.975
0.950
0.925
0.900
0.875
0.850
0.825
0.800
Importance
Denoised Heatmap
(Predicted Class: Class 0)
0.96
0.94
0.92
0.90
0.88
0.86
Importance
Superimposed Heatmap
(Predicted Class: Class 0)

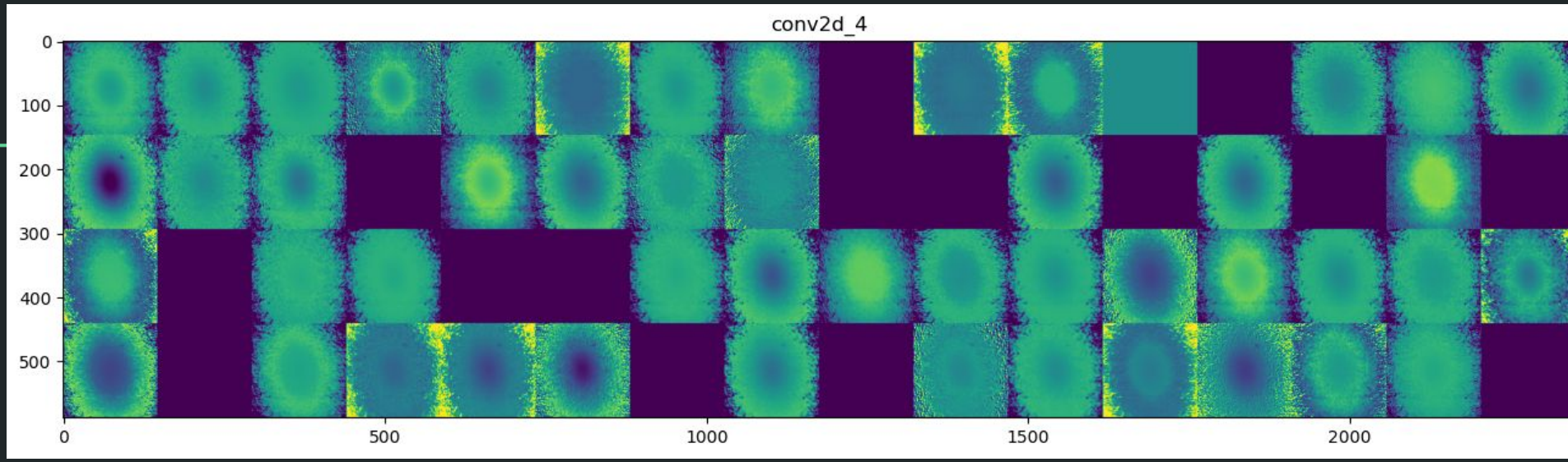
LIME Explanation:

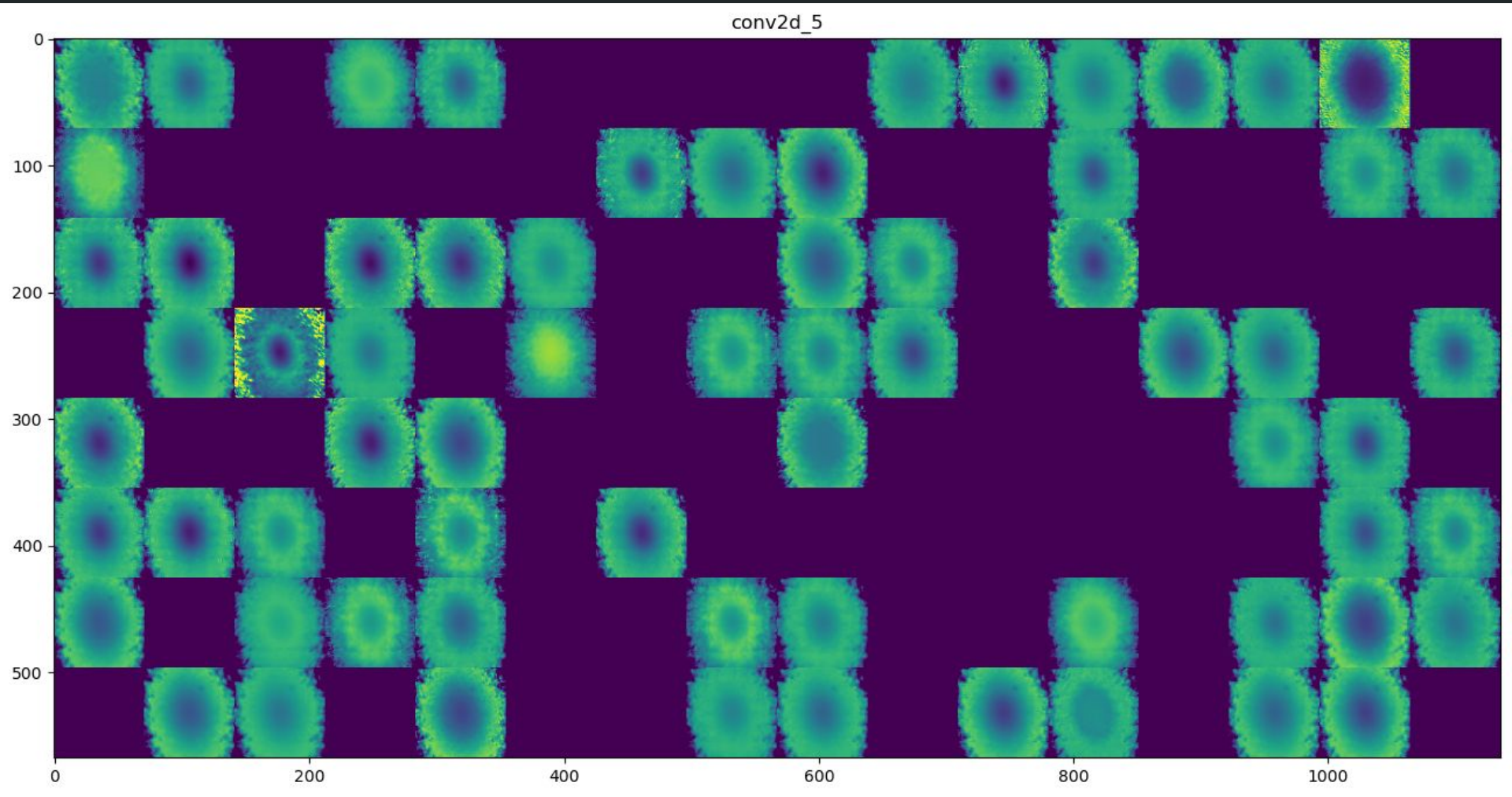
- LIME (**Local Interpretable** Model-agnostic Explanations) provides explanations for **individual predictions made by the model**.
- It generates a heatmap that highlights the regions of the image most responsible for the model's prediction.
- The explanation is based on perturbing the input image and observing the changes in the model's prediction.
- LIME provides local interpretability by approximating the model's behavior around the specific instance being analyzed

Lime


LIME Explanation
(Predicted Class: Class 0)


conv2d_3
0
200
400
0
1000
2000
3000
4000


conv2d_4
0
100
200
300
400
500
0
500
1000
1500
2000


conv2d_5
0
100
200
300
400
500
0
200
400
600
800
1000

Improvements to Grad-CAM Heatmap and Superimposed Heatmap Visualization

1. Color Map and Alpha¶

- **Color Map**: Changed the color map from `jet` to `viridis` for better visual differentiation and perceptual uniformity.
- **Alpha Blending**: Adjusted the alpha blending factor to 0.6 to improve the visibility and contrast of the superimposed image.

2. Normalization

- **Input Image Normalization**: Ensured that the input image array is normalized by dividing pixel values by 255.0 to have them in the range [0, 1]. This ensures consistency and better compatibility with the model's expected input format.

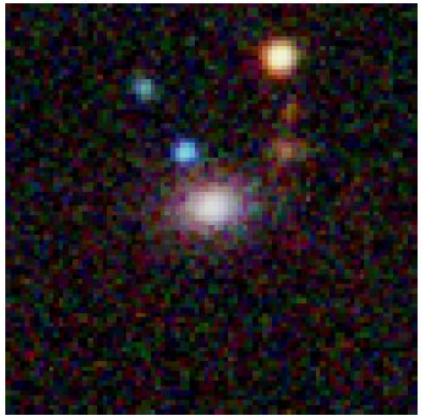
3. Denoising

- **Total Variation Denoising**: Applied Total Variation Denoising using the `denoise_tv_chambolle` method with a weight of 0.1. This step reduces noise in the heatmap, making the highlighted areas more distinct and clear.

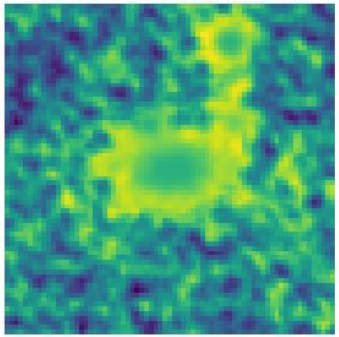
4. Overlay

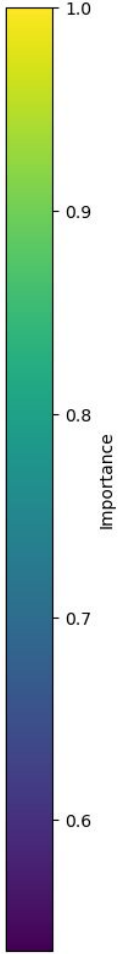
- **Superimposition Technique**: Enhanced the technique for overlaying the heatmap on the original image. This includes:
  - Using the adjusted alpha blending factor and the new color map.
  - Improved the process of resizing and aligning the heatmap with the original image for better visual coherence.

Original Image
(Predicted Class: Class 0)

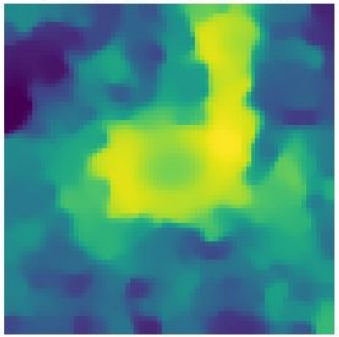


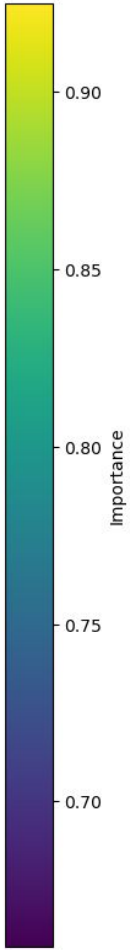
Grad-CAM Heatmap
(Predicted Class: Class 0)



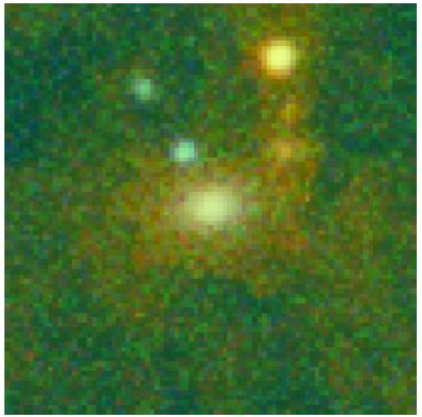


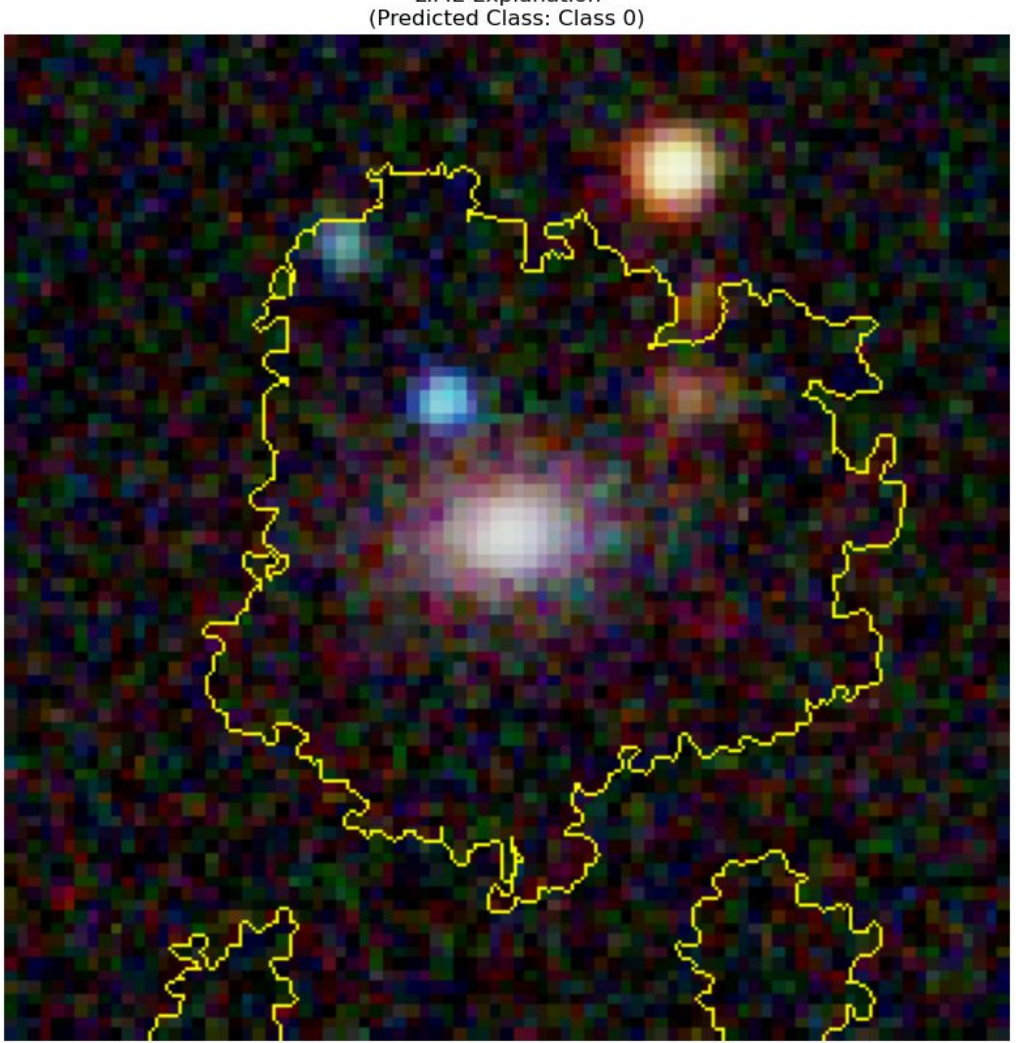
Denoised Heatmap
(Predicted Class: Class 0)

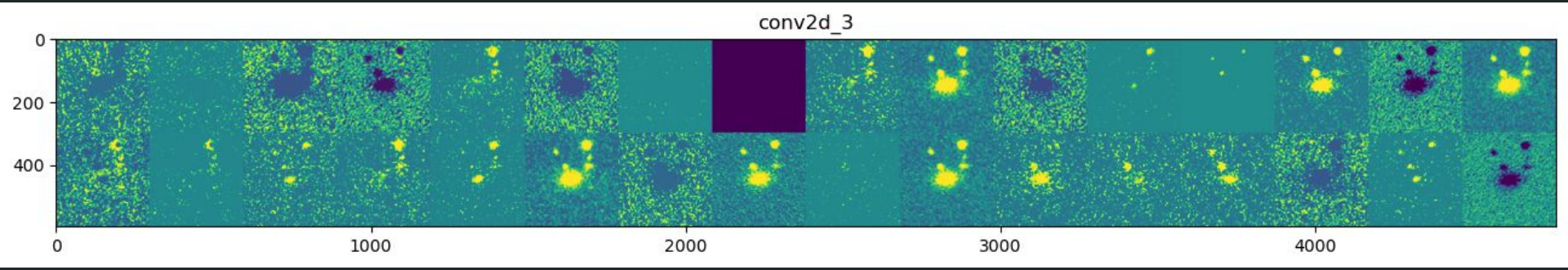


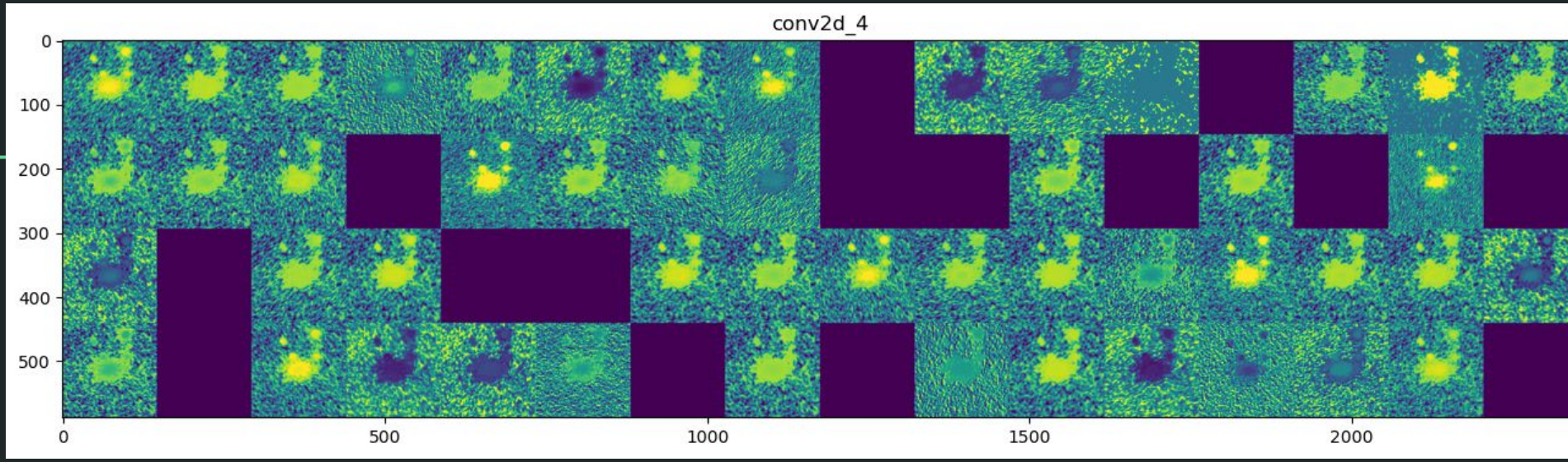


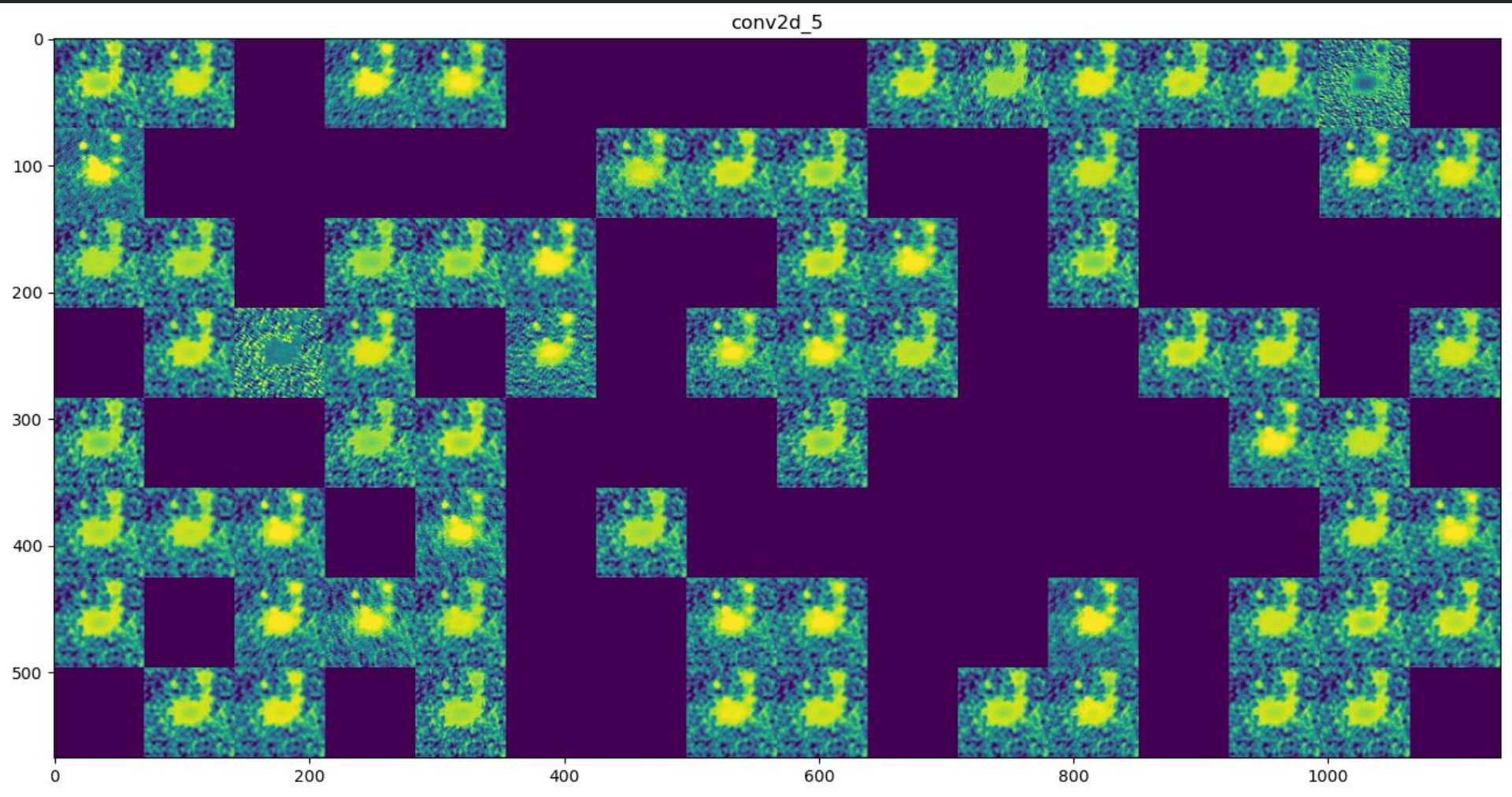
Superimposed Heatmap
(Predicted Class: Class 0)




LIME Explanation
(Predicted Class: Class 0)


conv2d_3
0
200
400
0
1000
2000
3000
4000


conv2d_4
0
100
200
300
400
500
0
500
1000
1500
2000


conv2d_5
0
100
200
300
400
500
0
200
400
600
800
1000

Low or Zero Activations: Some filters might not be activated by the given input image, resulting in near-zero values across the activation map for those filters.

Inappropriate Features: The specific input image might not contain the features that certain filters are designed to detect, leading to minimal activation in those filters.

Inference Phase: During inference (when you are visualizing activations), dropout should be disabled, and all neurons should be active. If dropout is accidentally left active during inference, it could lead to unexpected results, including some activation maps being partially or fully black due to random deactivation of neurons.

Thanks!

Exploring the Black Box: Analyzing Explainable AI Challenges and Best Practices Through Stack Exchange Discussions

Mohammad Mahdi Sayyadnejad [1] · Ali Asgari [2] · Ashkan Sami [3*] · Hooman Tahayori [1]

Received: date / Accepted: date

Abstract Explainable Artificial Intelligence (XAI) is a crucial domain within research and industry, aiming to develop AI models that provide human-understandable explanations for their decisions. While the challenges in AI, deep learning, and big data have been extensively explored, the specific concerns of XAI developers have received limited attention. To address this gap, we analyzed discussions on Stack Exchange websites to delve into these issues. Through a combination of automated and manual analysis, we identified 6 overarching categories, 10 distinct topics, and 40 sub-topics commonly discussed by developers. Our examination revealed a steady rise in discussions on XAI since late 2015, initially focusing on conceptualization and practical applications, with a notable surge in activity across all topic categories since 2019. Notably, Concepts and Applications, Tools Troubleshooting, and Neural Networks Interpretation emerged as the most popular topics. Troubleshooting challenges were commonly encountered with tools like Shap, Eli5, and Aif360, while Visualization issues were prevalent with Yellowbrick and Shap. Further-

✉ Mohammad Mahdi Sayyadnejad
mmehdisayyad@gmail.com

Ali Asgari
a.asgari@studenti.unina.it

*Corresponding author: Ashkan Sami
a.sami@napier.ac.uk

Hooman Tahayori
tahayori@shirazu.ac.ir

[1] Department of Computer Science & Engineering and IT, Shiraz University, Shiraz, Iran
[2] Department of Electrical Engineering and Information Technology, University of Naples Federico II, Naples, Italy
[3] Computer Science Subject Group, Edinburgh Napier University, Edinburgh, United-Kingdom