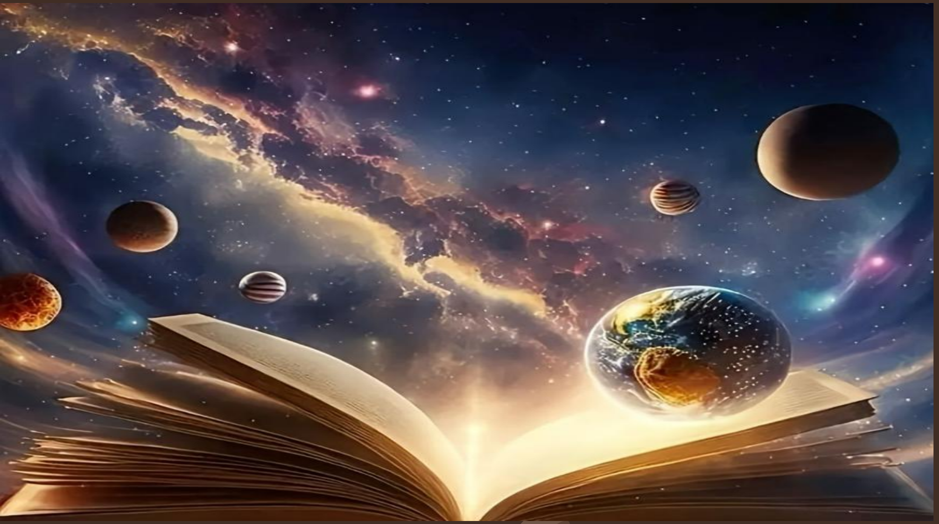
Explainable Transformer based Approach
for Galaxy classification
Astroinformatics
June 2025

Narges Davoudi

P37000156

Pro: Massimo Brescia

Introduction



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 ?

Irregular Galaxy

Spiral Galaxy

Why is it helpful to classify elliptical Galaxies?

❖Understanding galaxy formation and evaluation

❖Study environmental influence

❖Analyze stellar populations and gas content

❖Investigate dark matter distribution

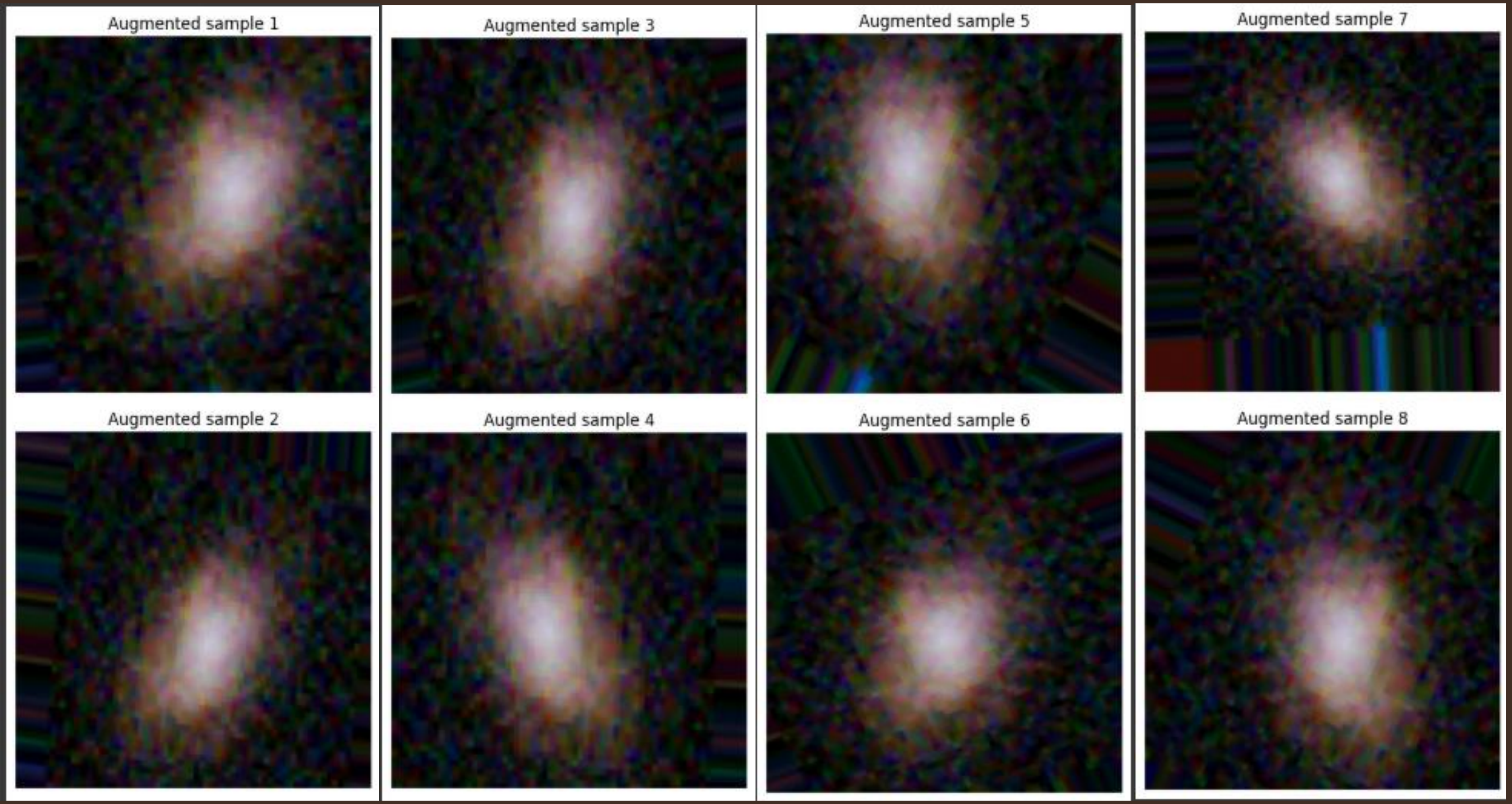
Loading and Preprocessing Data (Using Keras)

- Using tf.keras.preprocessing.image.load_img to load the image
- Convert to NumPy with img_to_array
- Resize to a specific size to 128×128 pixels for uniformity
- 📌 Keras explanation:
- Keras is a high-level API for building neural networks that runs on TensorFlow and makes writing models much easier.

Data Augmentation

Data Augmentation

Adds variations like rotations, flips, zoom to make the model more generalizable.



Improved CNN Architecture with Normalization and Overfitting Control

In this step, an optimized Convolutional Neural Network (CNN) was designed.

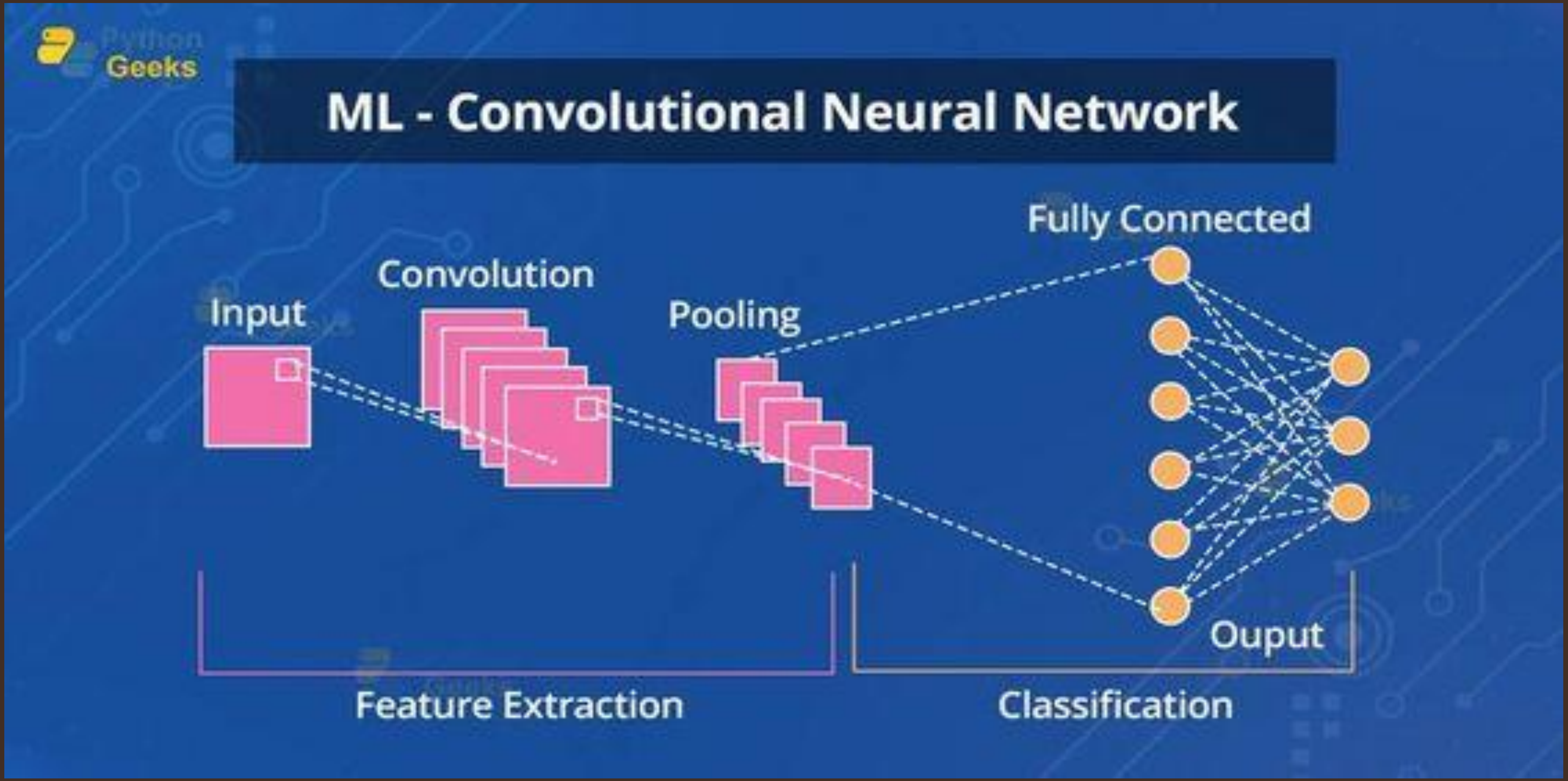
Convolutional layers are used to extract spatial features, and each is followed by Batch Normalization to stabilize gradient flow.

MaxPooling layers reduce dimensionality while preserving essential features.

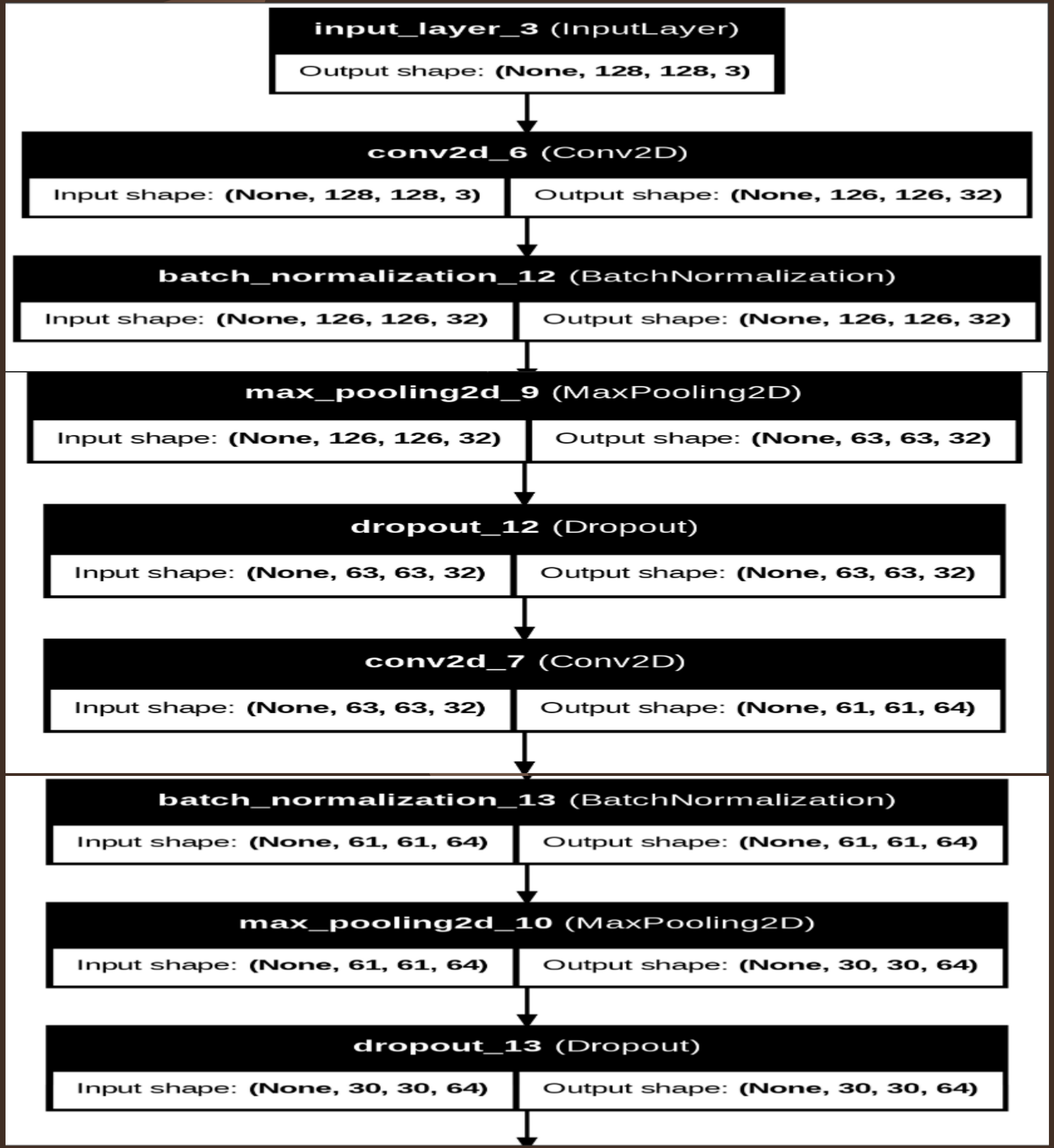
Dropout layers are applied to prevent overfitting.

Finally, a fully connected Dense layer makes the binary classification decision.

Convolutional Neural Network (CNN)



Model Architecture Overview



input_layer_3 (InputLayer)
Output shape: **(None, 128, 128, 3)**

conv2d_6 (Conv2D)
Input shape: **(None, 128, 128, 3)** Output shape: **(None, 126, 126, 32)**

batch_normalization_12 (BatchNormalization)
Input shape: **(None, 126, 126, 32)** Output shape: **(None, 126, 126, 32)**

max_pooling2d_9 (MaxPooling2D)
Input shape: **(None, 126, 126, 32)** Output shape: **(None, 63, 63, 32)**

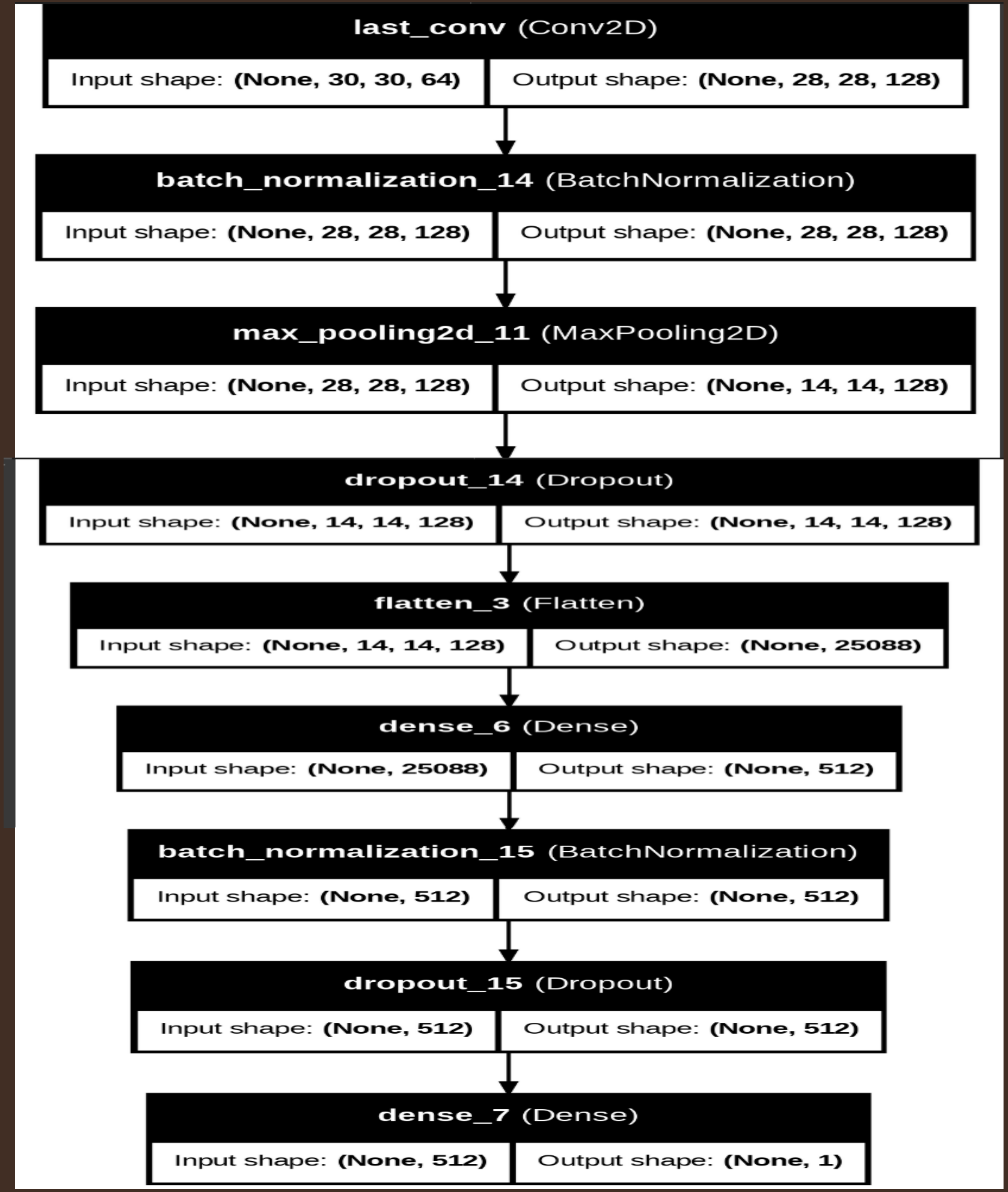
dropout_12 (Dropout)
Input shape: **(None, 63, 63, 32)** Output shape: **(None, 63, 63, 32)**

conv2d_7 (Conv2D)
Input shape: **(None, 63, 63, 32)** Output shape: **(None, 61, 61, 64)**

batch_normalization_13 (BatchNormalization)
Input shape: **(None, 61, 61, 64)** Output shape: **(None, 61, 61, 64)**

max_pooling2d_10 (MaxPooling2D)
Input shape: **(None, 61, 61, 64)** Output shape: **(None, 30, 30, 64)**

dropout_13 (Dropout)
Input shape: **(None, 30, 30, 64)** Output shape: **(None, 30, 30, 64)**



last_conv (Conv2D)
Input shape: **(None, 30, 30, 64)** Output shape: **(None, 28, 28, 128)**

batch_normalization_14 (BatchNormalization)
Input shape: **(None, 28, 28, 128)** Output shape: **(None, 28, 28, 128)**

max_pooling2d_11 (MaxPooling2D)
Input shape: **(None, 28, 28, 128)** Output shape: **(None, 14, 14, 128)**

dropout_14 (Dropout)
Input shape: **(None, 14, 14, 128)** Output shape: **(None, 14, 14, 128)**

flatten_3 (Flatten)
Input shape: **(None, 14, 14, 128)** Output shape: **(None, 25088)**

dense_6 (Dense)
Input shape: **(None, 25088)** Output shape: **(None, 512)**

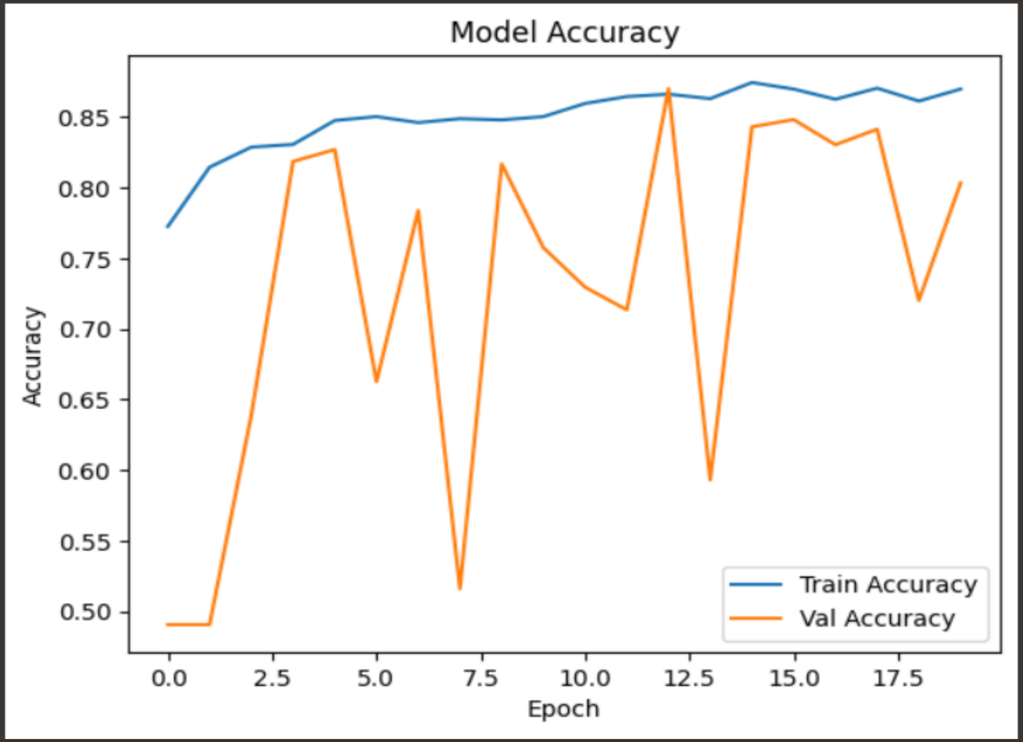
batch_normalization_15 (BatchNormalization)
Input shape: **(None, 512)** Output shape: **(None, 512)**

dropout_15 (Dropout)
Input shape: **(None, 512)** Output shape: **(None, 512)**

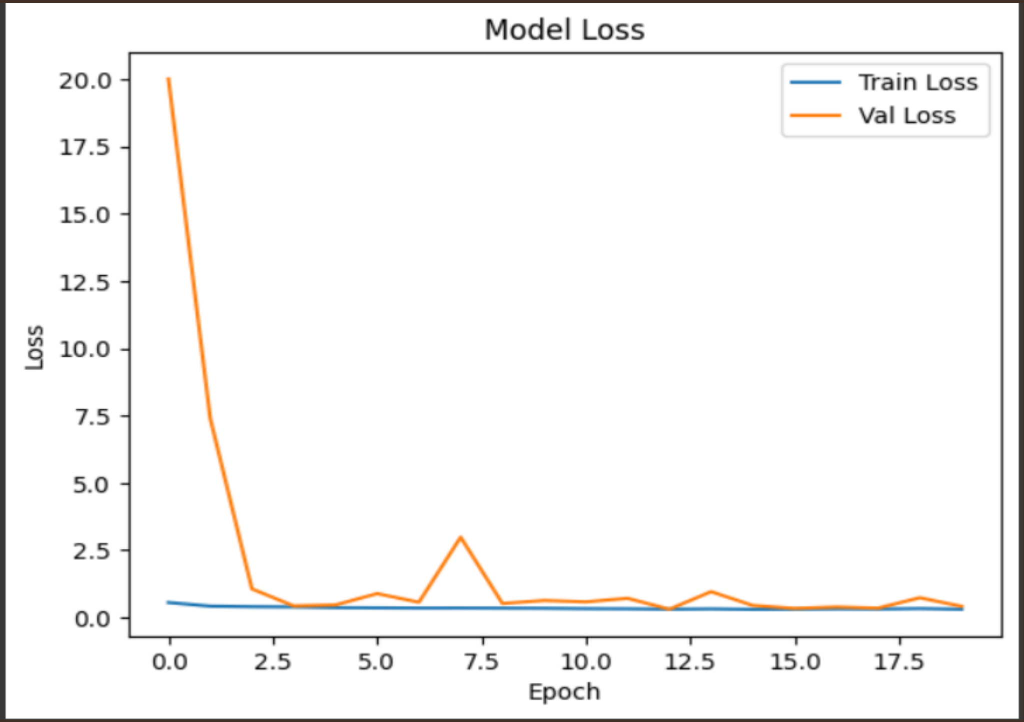
dense_7 (Dense)
Input shape: **(None, 512)** Output shape: **(None, 1)**

The model was trained over 20 epochs, and the accuracy and loss were monitored for both training and validation sets.
In the accuracy graph, the model shows consistent improvement in performance with minimal overfitting, as seen in the proximity of the training and validation curves.
In the loss graph, both training and validation loss steadily decrease, with the model achieving a final validation accuracy of approximately 89%.

Accuracy over Epochs



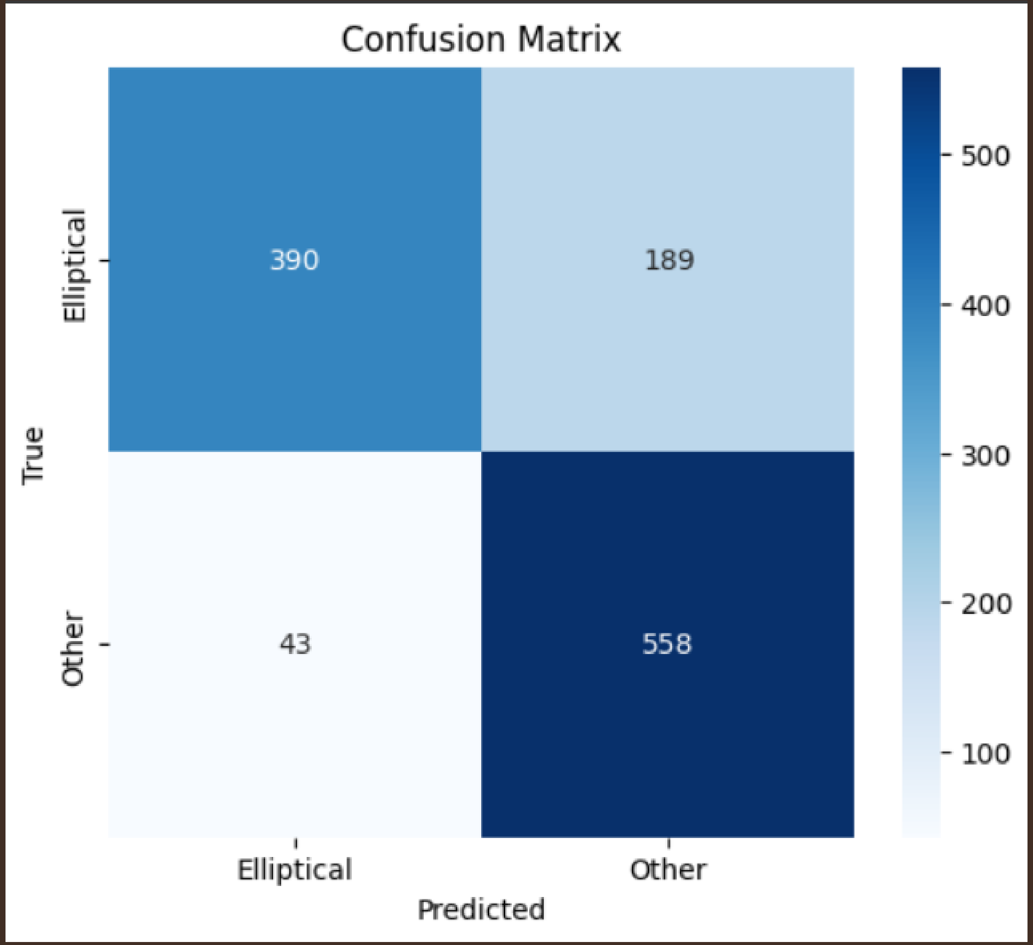
Loss over Epochs



Model Evaluation using Confusion Matrix

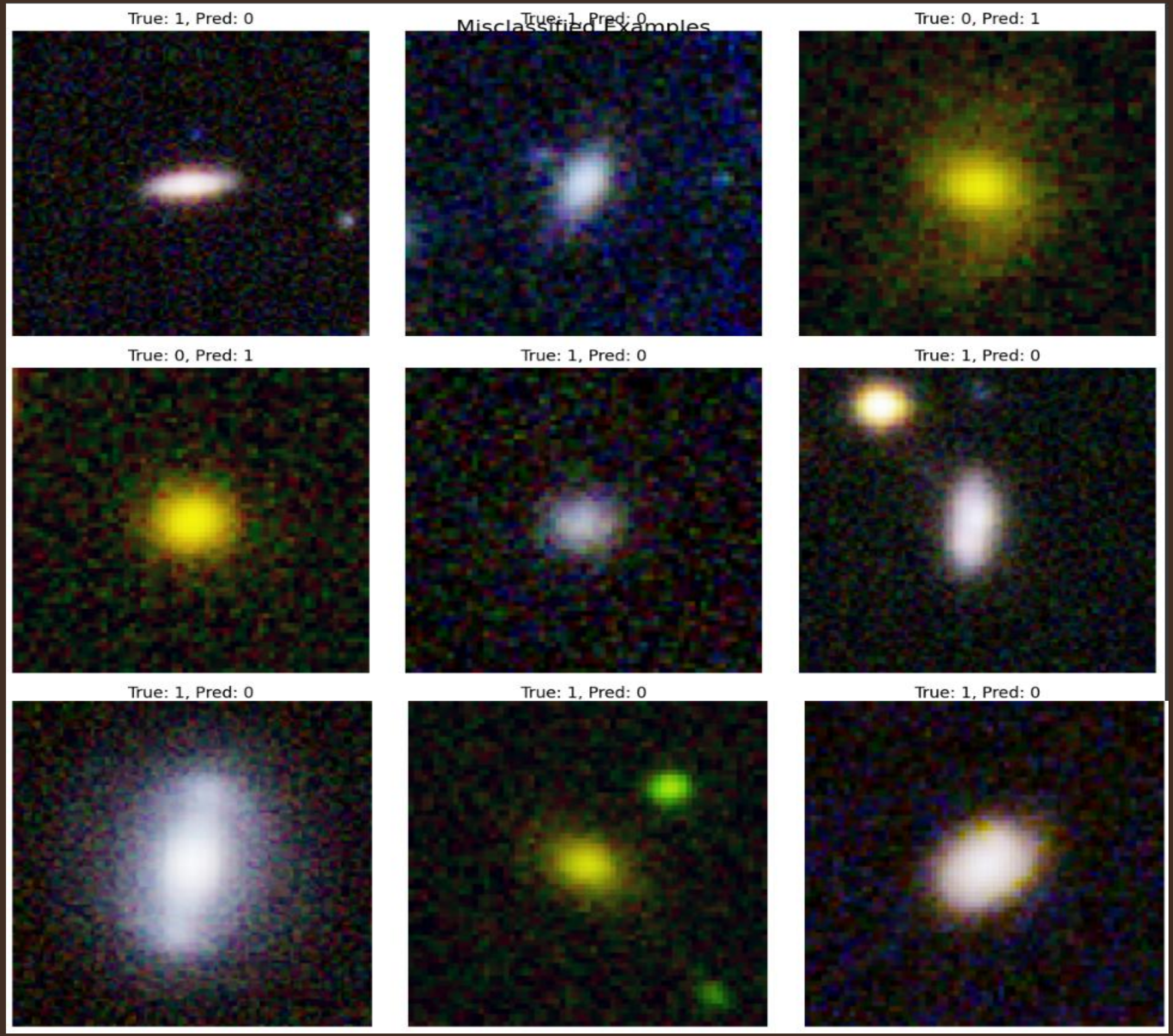
This slide presents the confusion matrix and classification report, showing how well the model performs on both elliptical and non-elliptical classes.

	Precision	Recall	F1-Score	Support
Elliptical	0.90	0.67	0.77	579
Other	0.75	0.93	0.83	601
Accuracy			0.80	1180
Marco Avg	0.82	0.80	0.80	1180
Weighted Avg	0.82	0.80	0.80	1180



Misclassified Galaxy Samples

This slide shows galaxy images that were misclassified by the model. Analyzing these helps identify model weaknesses.



True: 1

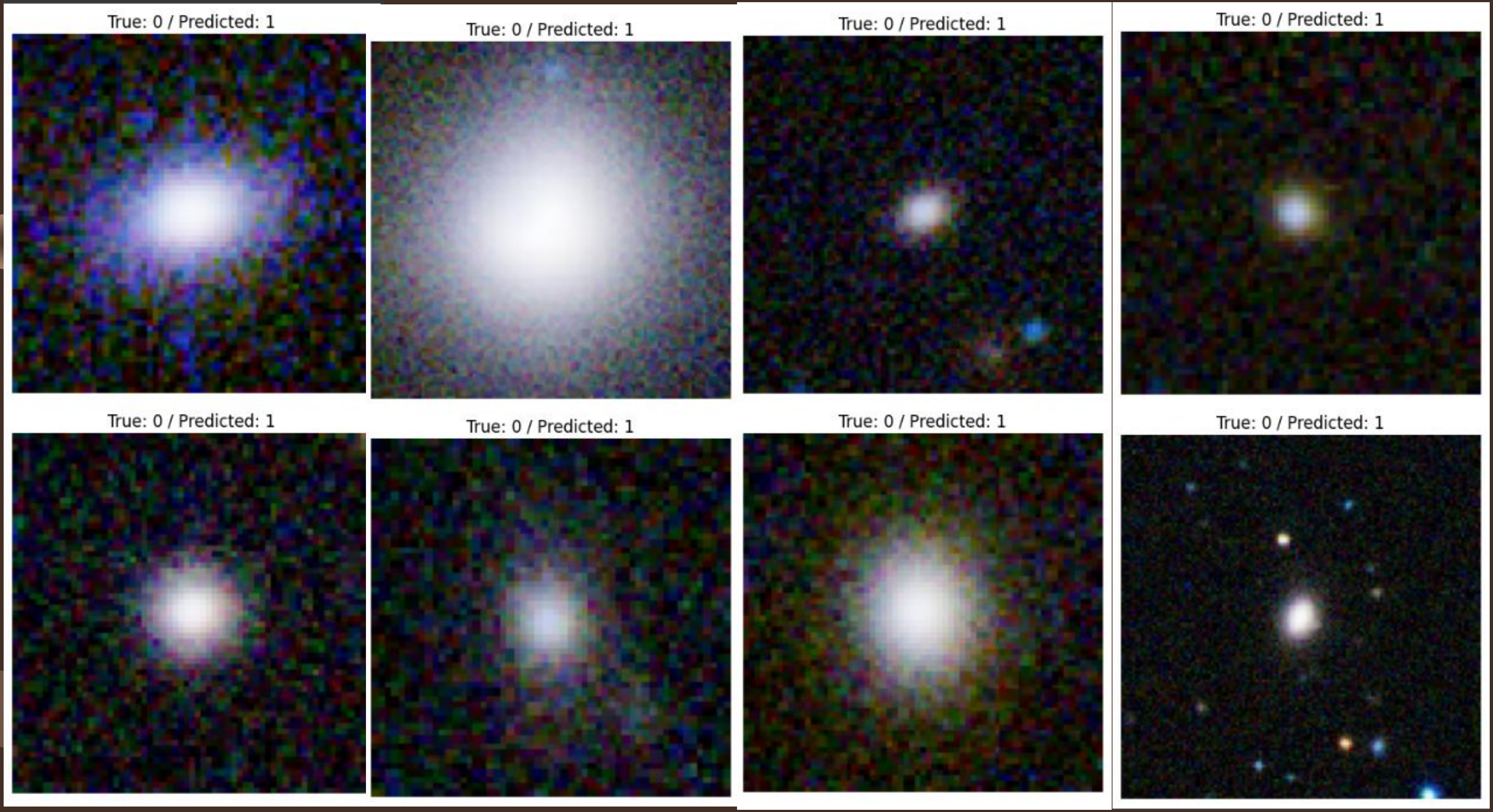
Predicted: 0

Examples of Misclassified Galaxy Images

Next slide shows validation images that were incorrectly classified by the model. Investigating these samples helps identify ambiguous or borderline cases and informs potential model improvements in future iterations.

- These are validation samples misclassified by the model.
- Each image shows the true class and the predicted label.

Examples of Misclassified Galaxy Images



True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

True: 0 / Predicted: 1

Training Summary of Improved CNN Model

The final CNN model consists of three convolutional blocks, each followed by Batch Normalization and Dropout.

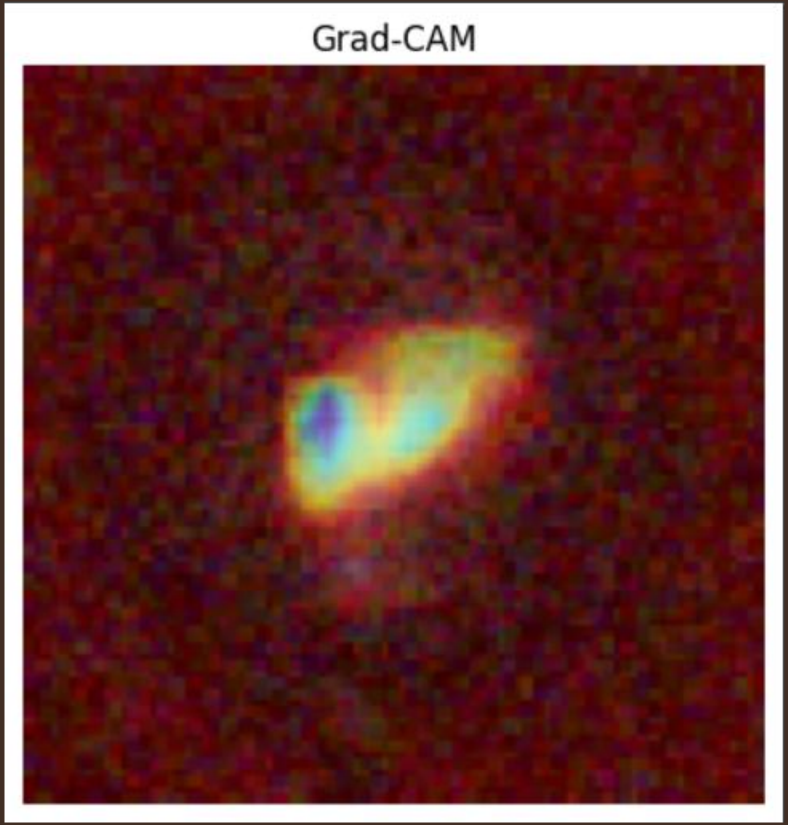
Training was performed on augmented data for 20 epochs with a batch size of 32.

The Adam optimizer was used with Binary Crossentropy as the loss function.

The model achieved ~98% training accuracy and ~89% validation accuracy.

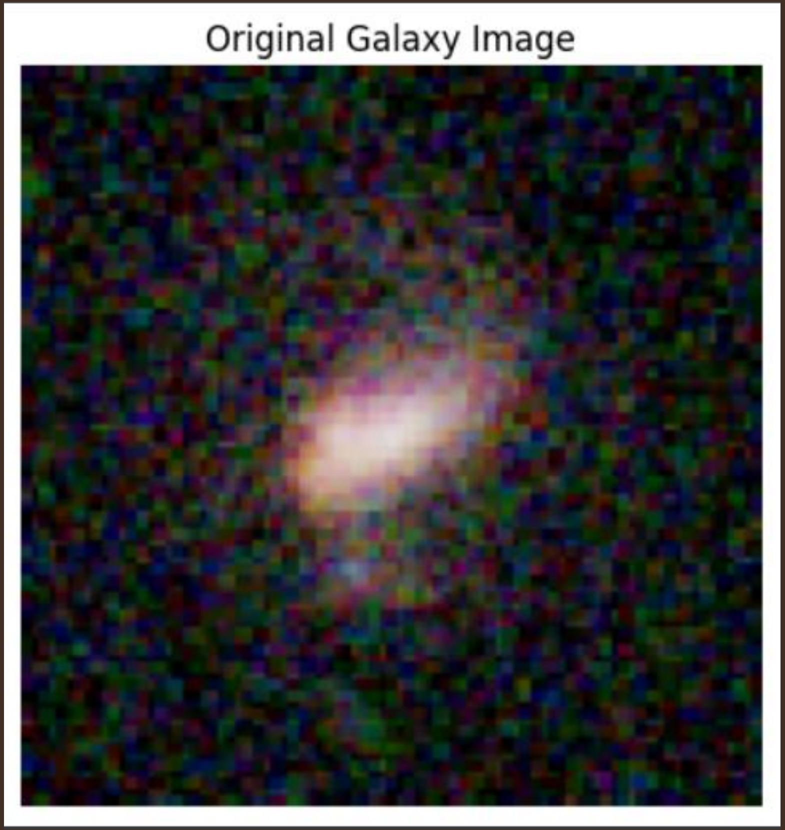
Visual Explanation using Grad-CAM

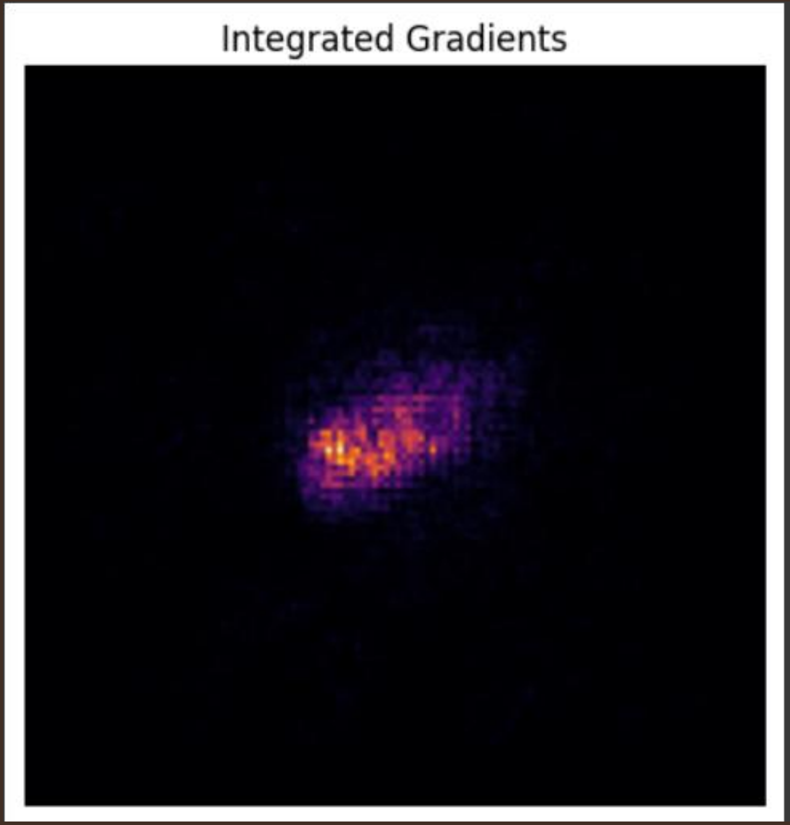
Grad-CAM is a visual explanation method that highlights the most influential regions of the image for a model's prediction.
In this project, a heatmap is generated from the last convolutional layer's gradient and superimposed on the original galaxy image.
As shown, the model focuses on central and bright regions of the galaxy for decision-making.



Model Interpretation via Integrated Gradients

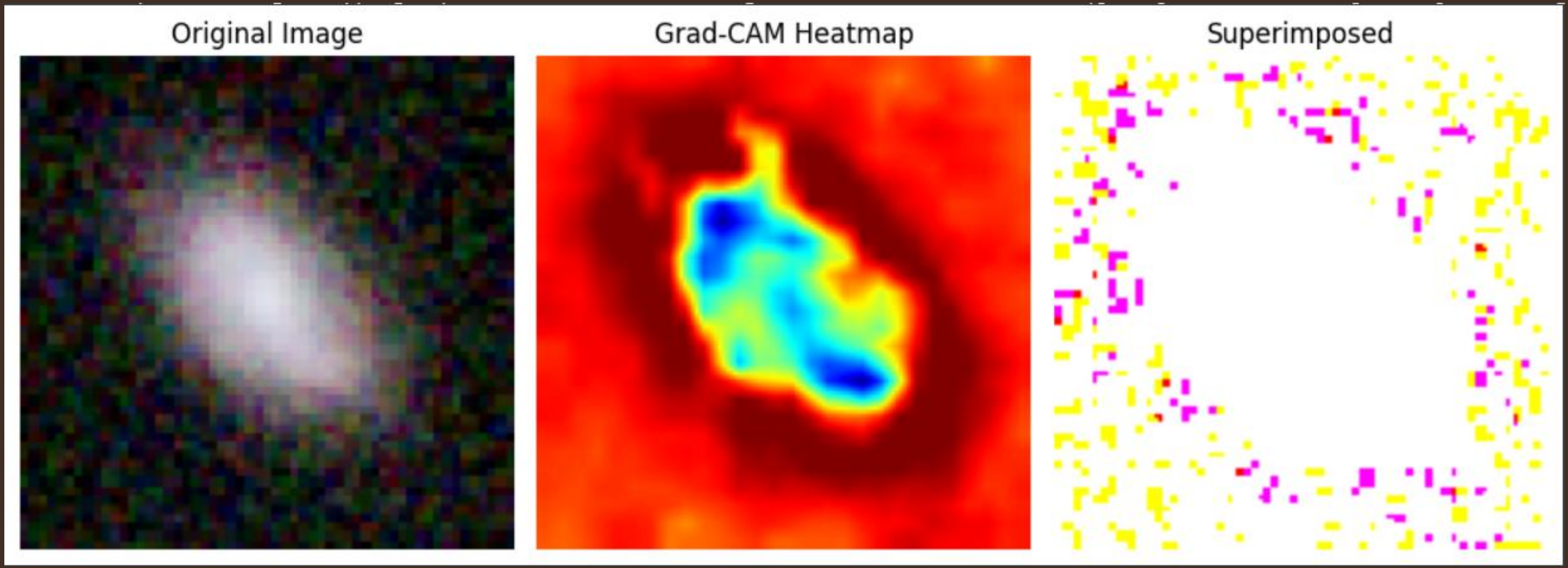
Integrated Gradients is a method for attributing model predictions to input features (pixels) in a stable and reliable way. It integrates gradients along the path from a baseline (black image) to the actual input image. The brighter regions in the visualization indicate which pixels most influenced the model's classification decision.





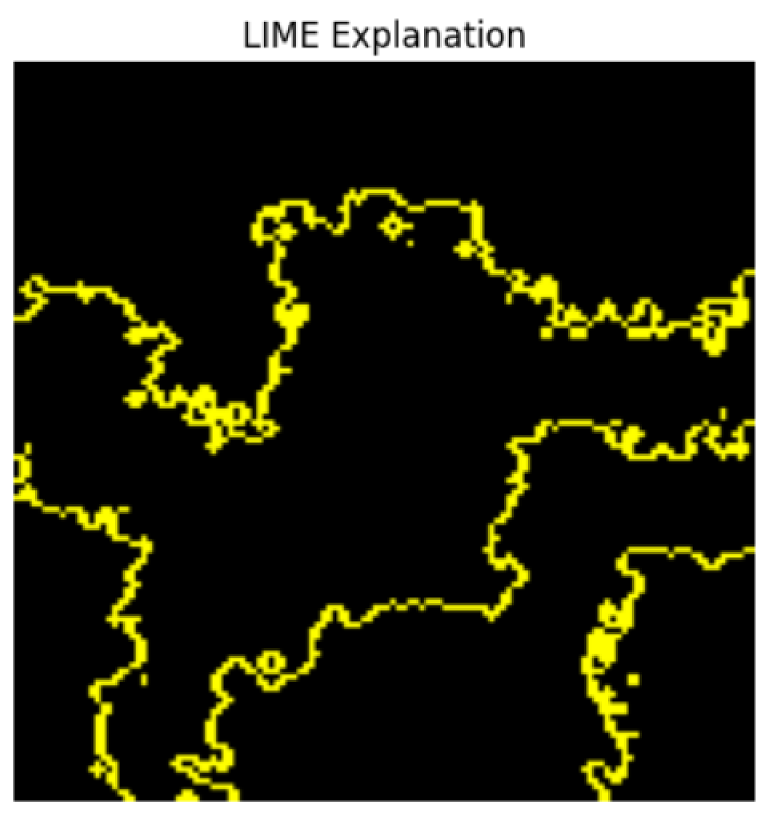
Grad-CAM on Single Sample

Grad-CAM heatmap for a sample galaxy image to show which areas the model focused on during prediction.



Local Explanation of Model Prediction using LIME

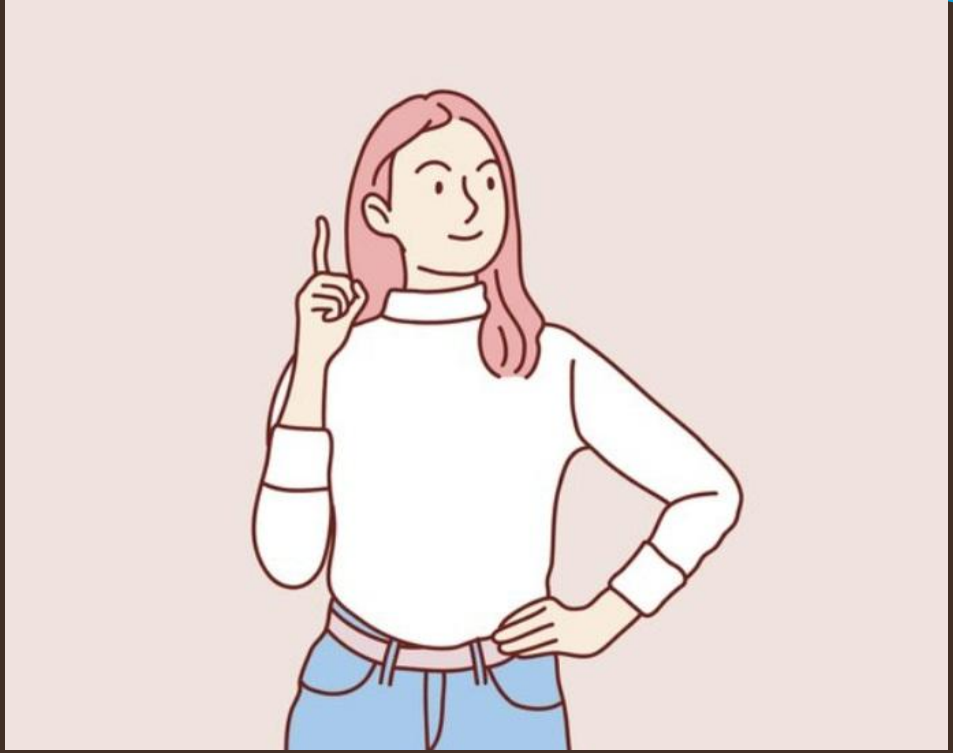
LIME is a model-agnostic explanation method that highlights the most influential regions of an image for a specific prediction.
It builds a local interpretable linear model around the input sample.
In the visualization, the highlighted regions contributed most to the model's decision.



Visual Interpretation of Model Decisions using ExplainableAI

In this project, three explainability methods – Grad-CAM, Integrated Gradients, and LIME – were applied to analyze the decisions made by a custom-trained CNN model. Each technique highlights relevant image regions from a different perspective. Unlike the reference slides which used VGG and ResNet, here the explainability is demonstrated on a custom-built model. The results consistently show focus on central, bright galaxy regions, aligning with expected astronomical structures.

04
Conclusion



The proposed CNN achieved ~89% accuracy in distinguishing elliptical galaxies. Grad-CAM visualizations confirmed the model’s focus on relevant regions. Future improvements may include using pre-trained models and analyzing difficult samples.

Thank You For Your Attention