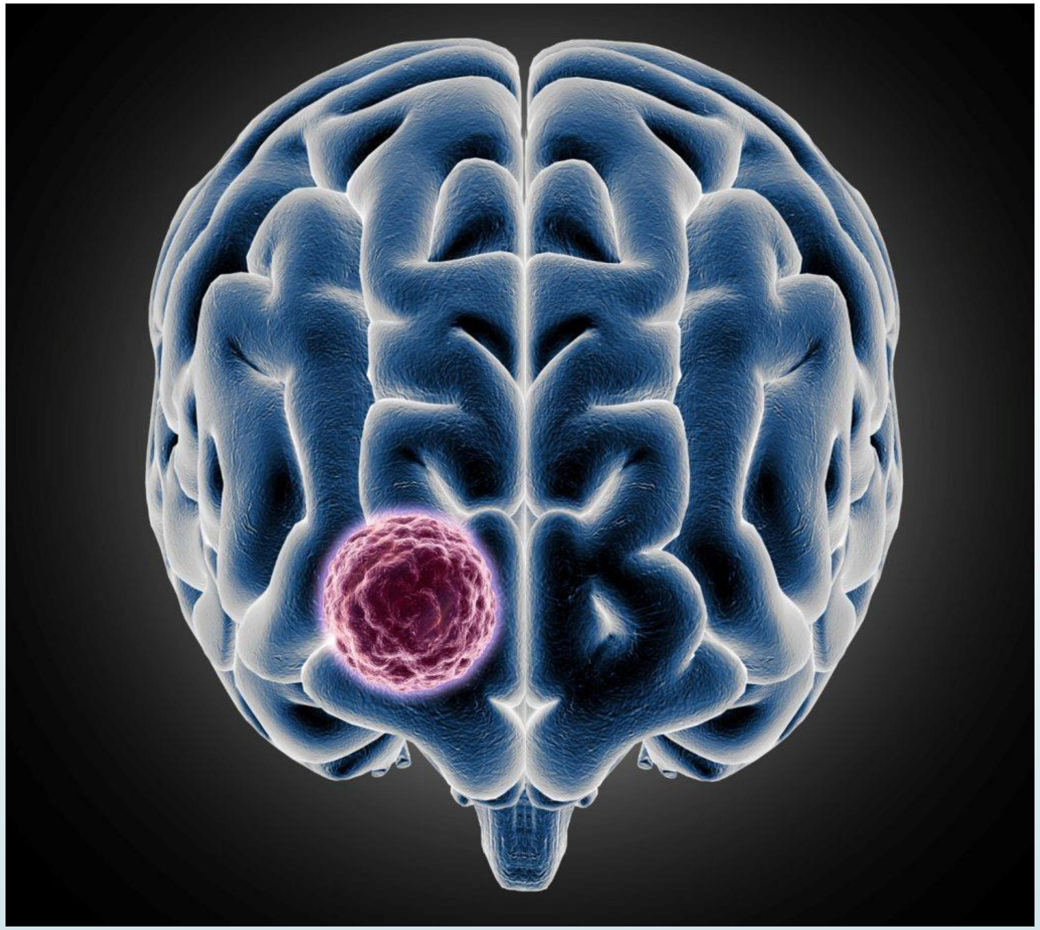
“Reliable and Explainable AI for Brain Tumor Detection Using Graph Neural Networks”



Presented by: Neha Mahanand

30-10-2025

Contents

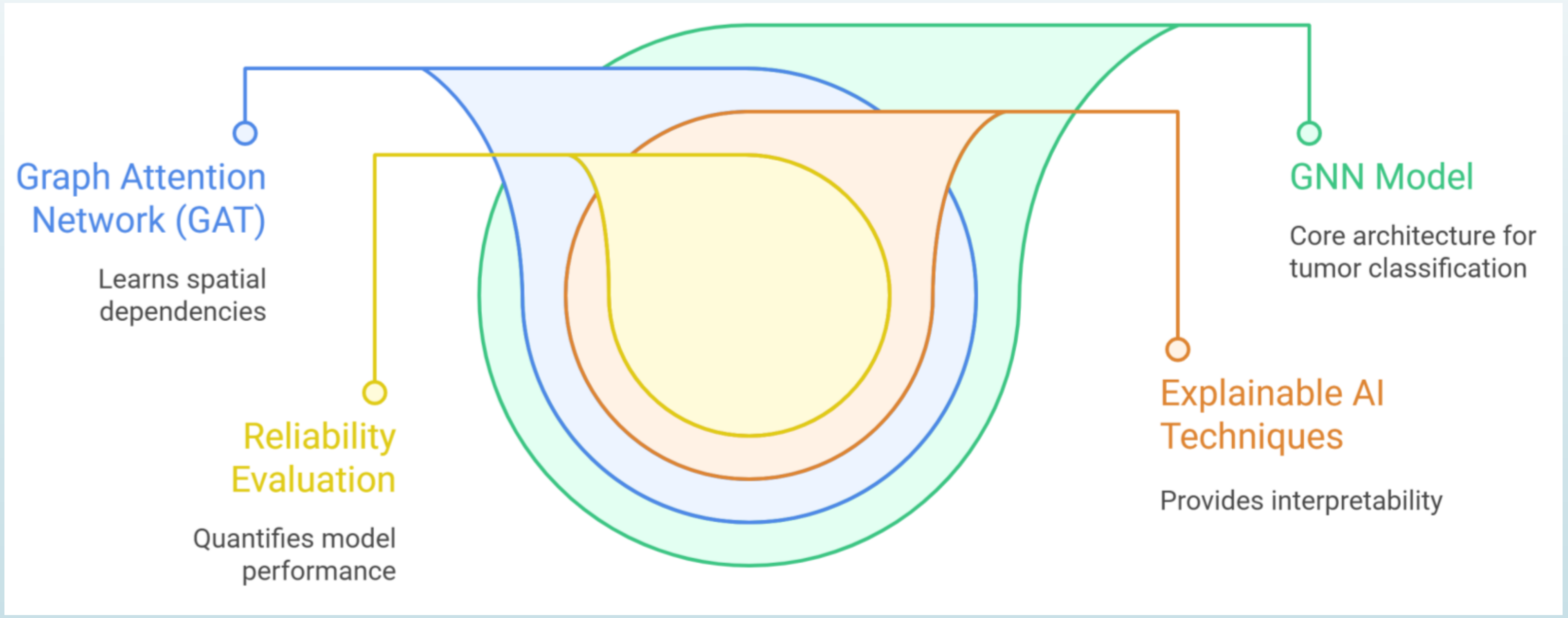
1. Project Background

1.1 Introduction

Brain tumors are a major health concern & early detection is critical. Conventional CNNs give high accuracy but poor interpretability. Explainable AI (XAI) adds transparency and trust to AI-based medical diagnosis.

1.2 Objective

This work aims to develop a GATv2-based Graph Neural Network for accurate brain tumor classification, using attention mechanisms and residual connections for better interpretability, stability, and clinical insight through node-level visualization.


Graph Attention Network (GAT)
Learns spatial dependencies
Reliability Evaluation
Quantifies model performance
GNN Model
Core architecture for tumor classification
Explainable AI Techniques
Provides interpretability

2. Methods and Work Flow

2.1 Dataset

- The dataset is from Kaggle (link: https://www.kaggle.com/datasets/sartajbhuvaji/brain-tumor-classification-mri)

- Number of class: 4
- ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor’]
- These are encoded using label encoding

Training set distribution:
glioma_tumor: 822 images
meningioma_tumor: 822
images no_tumor: 395 images
pituitary_tumor: 826 images

Testing set distribution:
glioma_tumor: 100 images
meningioma_tumor: 115
images no_tumor: 105 images
pituitary_tumor: 74 images

2.2 Image processing & Graph construction

Step 1

Image and Label Collection

Gather brain images and corresponding labels

Step 2

Image Preprocessing

Load, normalize, and skull-strip images

Step 3

Patch Extraction

Divide brain into overlapping patches

Step 4

Feature Representation

Extract features from each patch

Step 5

Graph Connectivity

Define connections between patches

Step 6

PyTorch Geometric Graphs

Create graph data structures for PyTorch

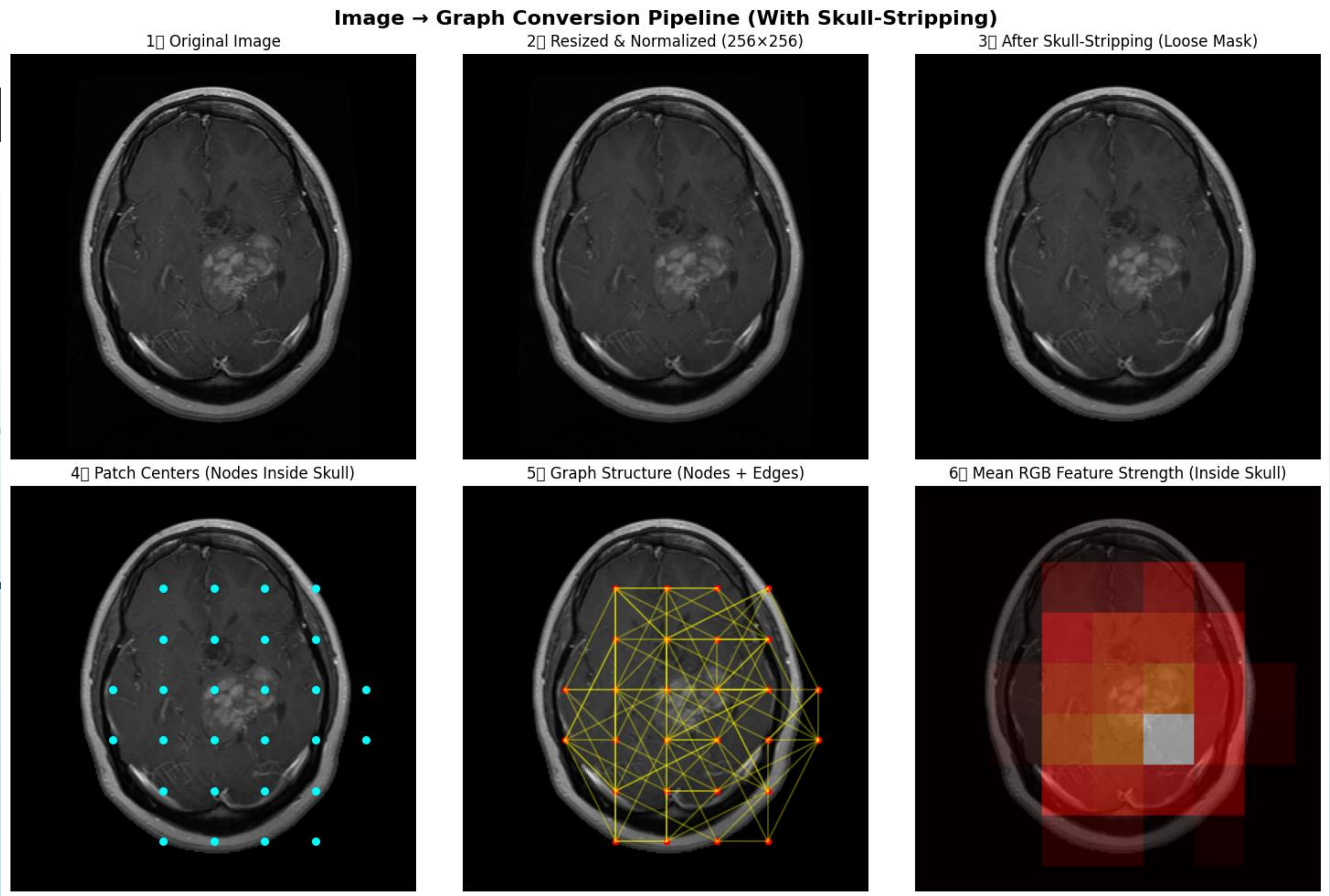
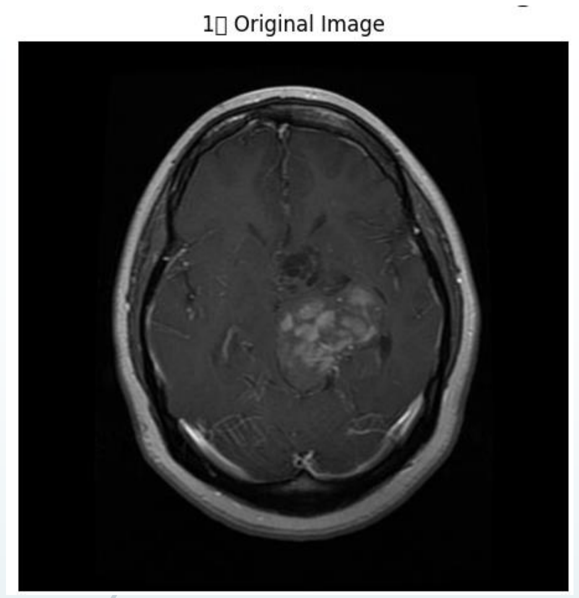


Image → Graph Conversion Pipeline (With Skull-Stripping)
1 Original Image
2 Resized & Normalized (256×256)
3 After Skull-Stripping (Loose Mask)
4 Patch Centers (Nodes Inside Skull)
5 Graph Structure (Nodes + Edges)
6 Mean RGB Feature Strength (Inside Skull)

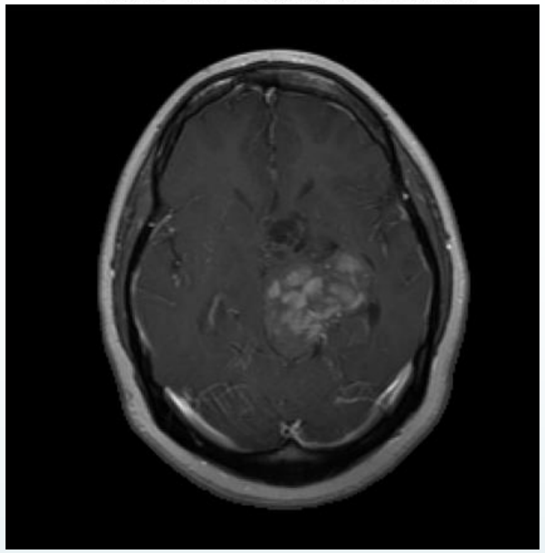


- Step 1: Image and Label Collection
  - Loop through all subfolders in training/testing directories.
  - Each folder name represents a class label (tumor type).
  - Collect file paths → list of all image locations.
  - Collect labels → corresponding tumor type for each image.

- Step 2: Image Loading, Normalization, and Skull-Stripping
  - Loaded each image using cv2.imread(), converted to RGB color format, resized to 256 × 256 pixels, normalize pixel values → divide by 255.0 for consistency
  - Apply loose skull-stripping:
  - Convert image to grayscale.
  - Apply Gaussian blur to reduce noise.
  - Generate a binary mask with Otsu’s thresholding.
  - Use morphological closing to clean small holes.
  - Keep largest connected component → focuses on brain tissue.
  - masked brain image + mask, ensures only brain regions are processed.

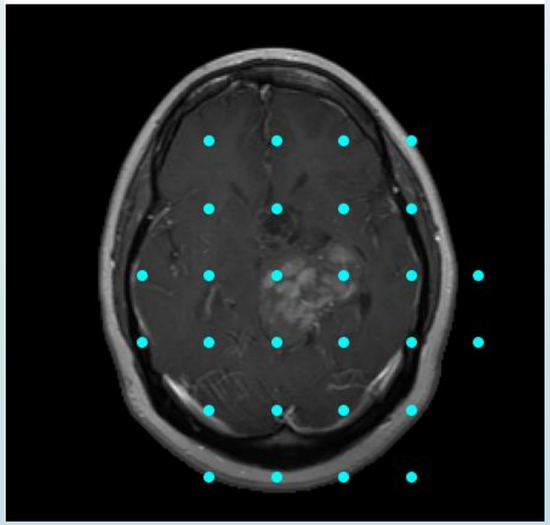
- Step 3: Patch Extraction
  - Split each 256×256 image into small **patches**
  - Each patch → represents a **local region** of the brain
  - Compute patch center coordinates → normalized to 0–1
  - There are 64 patches per image (for 32×32 patch size)
  - Each patch is a future **graph node**

3️ After Skull-Stripping (Loose Mask)



- Step 4: Patch Feature Representation
  - For each patch, calculate the mean color values → (R, G, B)
  - Each patch → feature vector [R_mean, G_mean, B_mean]
  - Each patch becomes a feature vector summarizing color/texture.
  - Example:x.shape = (64 nodes, 3 features)

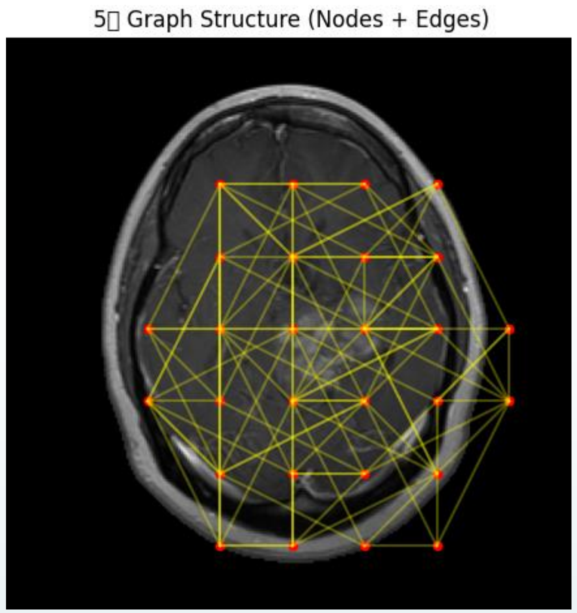
4️ Patch Centers (Nodes Inside Skull)

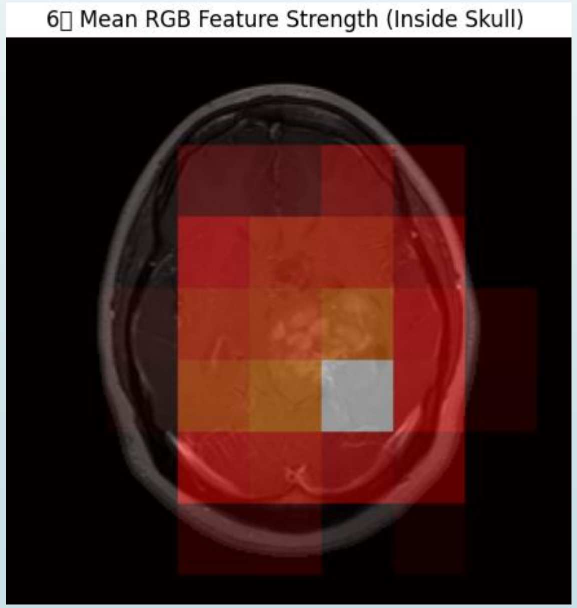


- Step 5: Building Graph Connectivity
  Compute Euclidean distance between patch centers.
  - If distance < 0.3 → connect nodes with an edge
  - This creates a graph structure where nearby patches are connected.

- Step 6: Creating PyTorch Geometric Graphs
  - Convert data to PyTorch Geometric Data objects
  - Each graph contains: x → node , features, edge_index → graph connectivityy, y→ label tensor (class ID)

SUMMARY: Train graphs: 2865 Test graphs: 394





3.MODEL OVERVIEW

- **GATv2 (Graph Attention Network v2)** helps the model learn which nodes (patches) are most important by using attention weights.
- Each GATv2Conv layer learns node relationships by focusing more on important connected neighbors.
- Some tumor classes have fewer images than others, so the model might get biased**.** So, we assigned class weights inversely proportional to the number of samples.

➨ **Model Hyperparameters**

- Node feature size: in_channels (patch features from skull-masked image).
- Hidden channels: 96
- Batch size: 8
- Learning rate: 0.0005
- Epochs: 30
- Dropout: 25%
- Weight decay: 0.0001

4. MODEL ARCHITECTURE

- Each GATv2 layer learns relationships between patches of the brain.
- GATv2Conv Layers: Learn node relationships using attention mechanism.
- stacked 3 GATv2Conv layers with BatchNorm and ELU activation for non-linearity.
- Conv1 → hidden 96, Conv2 → hidden 96, Conv3 → hidden 48, (with residual projection)
- BatchNorm: Normalizes node features.

- **Residual Connections**: Adds previous layer info to improve gradient flow. This preserve information from earlier layers and prevent gradient vanishing. Residuals help stabilize deep graph learning.
- After GNN layers, we combine all node information to make a final tumor prediction.

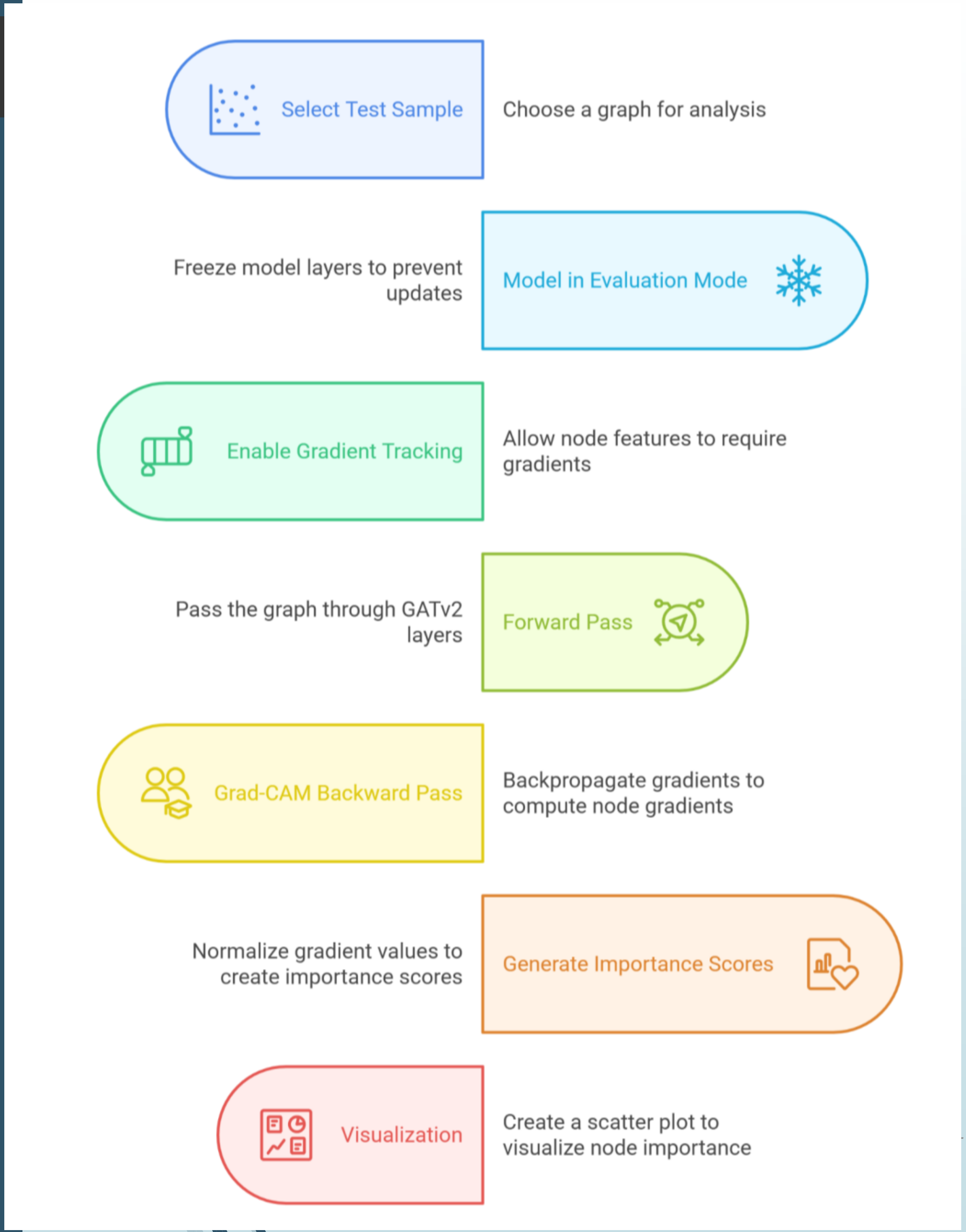
FC1: 128 units + ReLU

FC2: Output layer → predicts tumor class (4 classes)

- **Optimizer & SchedulerOptimizer:**
- AdamW → handles weight decay properly.
- Loss Function: CrossEntropyLoss (weighted by class imbalance).
- Scheduler: CosineAnnealingLR → gradually reduces learning rate for better convergence.

- **Training Loop:**
- Model learns node importance through forward pass. Computes loss and backpropagates gradients. Updates weights. Evaluates accuracy after each epoch
- **Evaluation:**
- Predictions are compared with true labels.
- Metrics: Validation accuracy used to monitor performance.

5. Grad-CAM Style Node Importance Visualization



Select Test Sample — Choose a graph for analysis

Model in Evaluation Mode — Freeze model layers to prevent updates

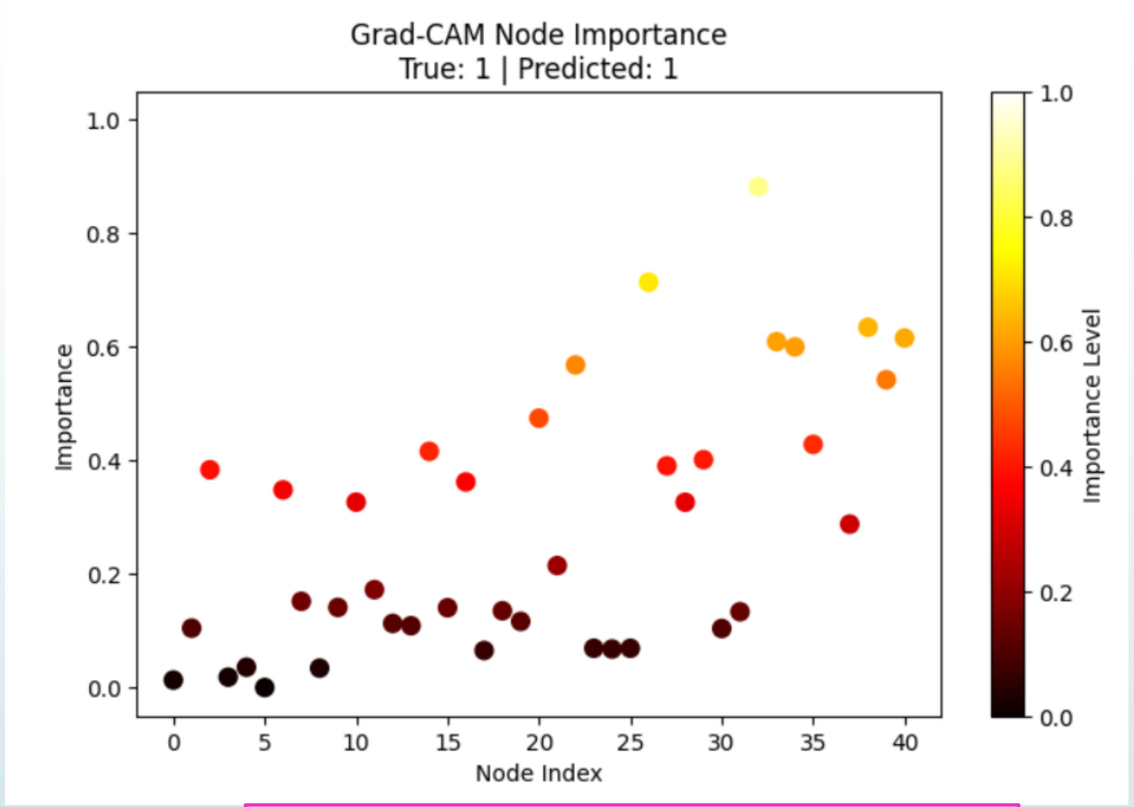
Enable Gradient Tracking — Allow node features to require gradients

Forward Pass — Pass the graph through GATv2 layers

Grad-CAM Backward Pass — Backpropagate gradients to compute node gradients

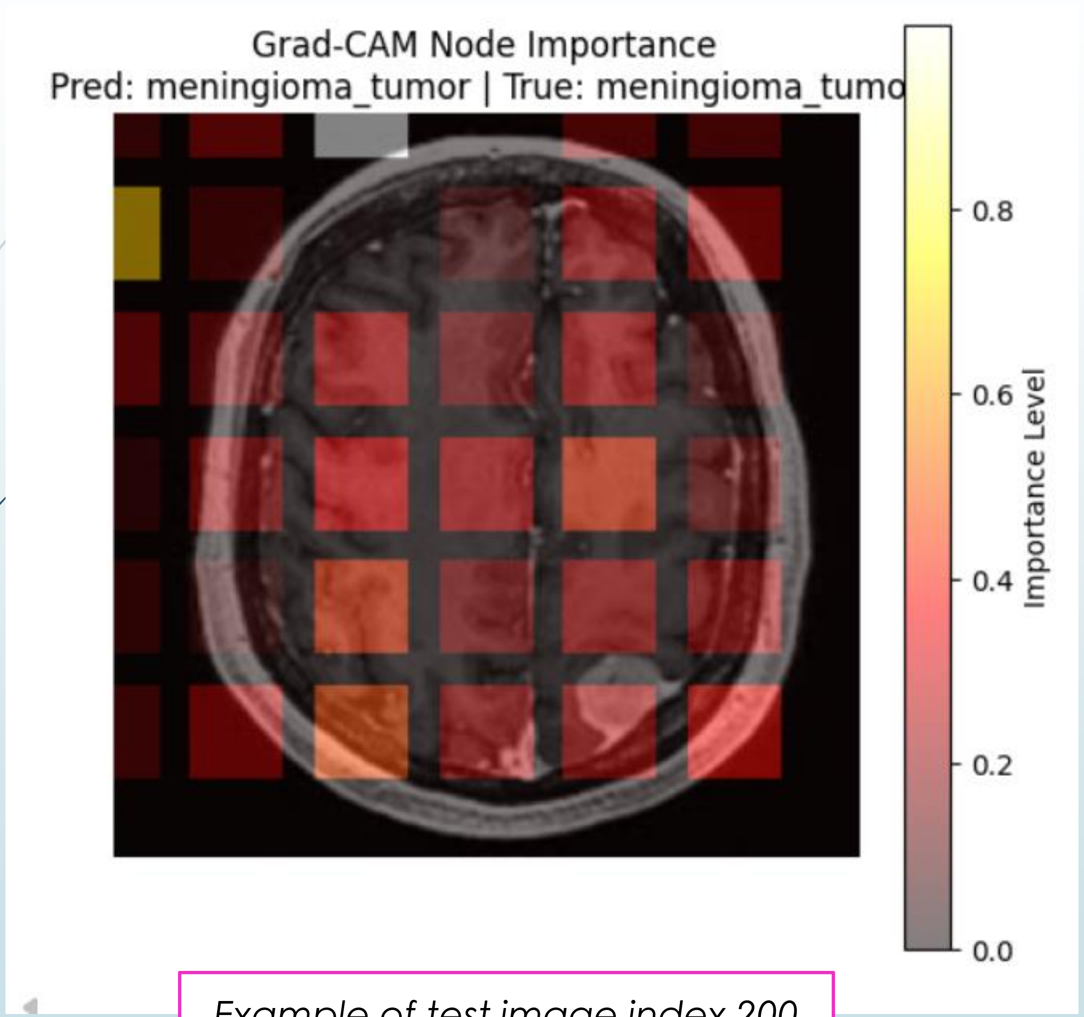
Generate Importance Scores — Normalize gradient values to create importance scores

Visualization — Create a scatter plot to visualize node importance



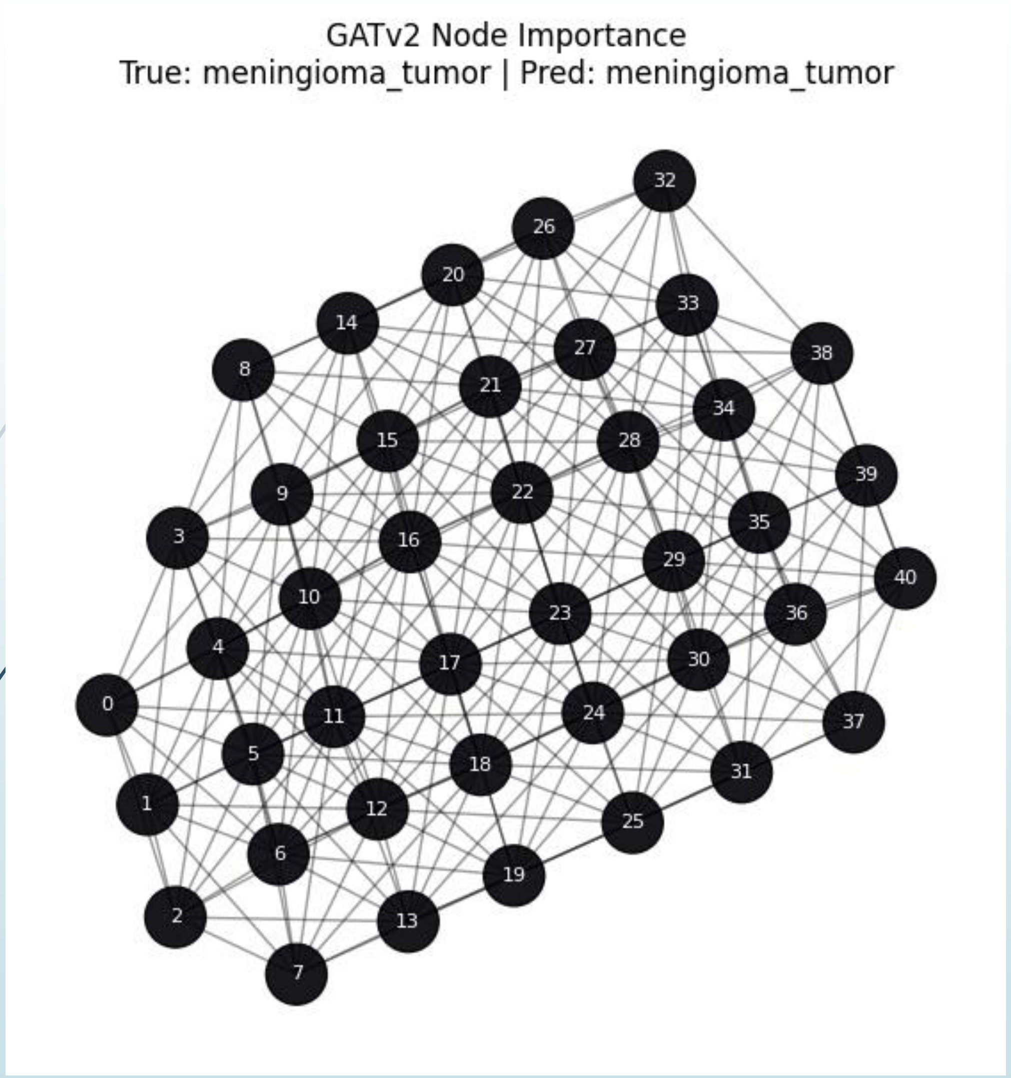
Example of test image index 200

6. Grad-CAM Heatmap for Graph Node Importance



Example of test image index 200

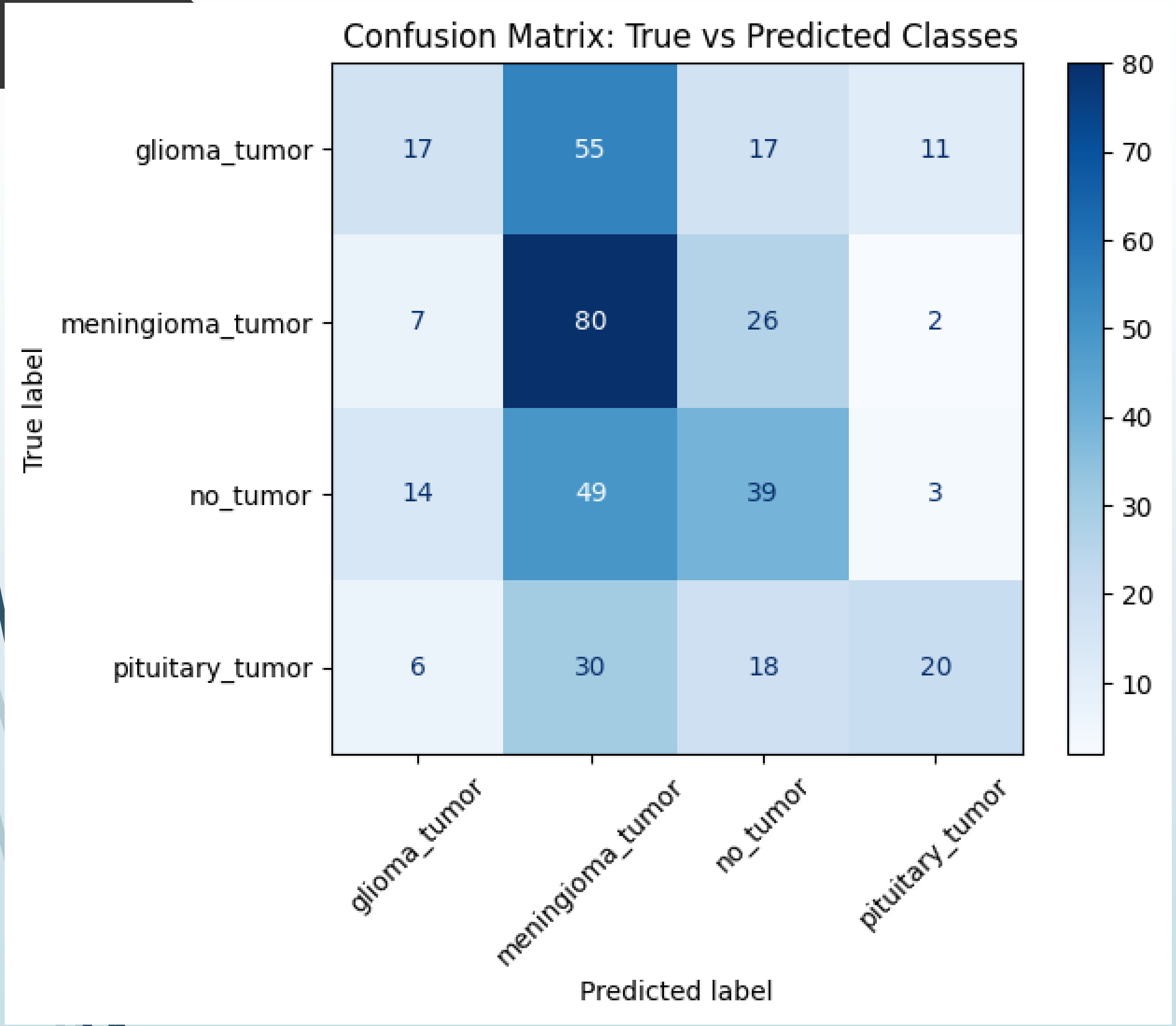
7. Graph Node Importance



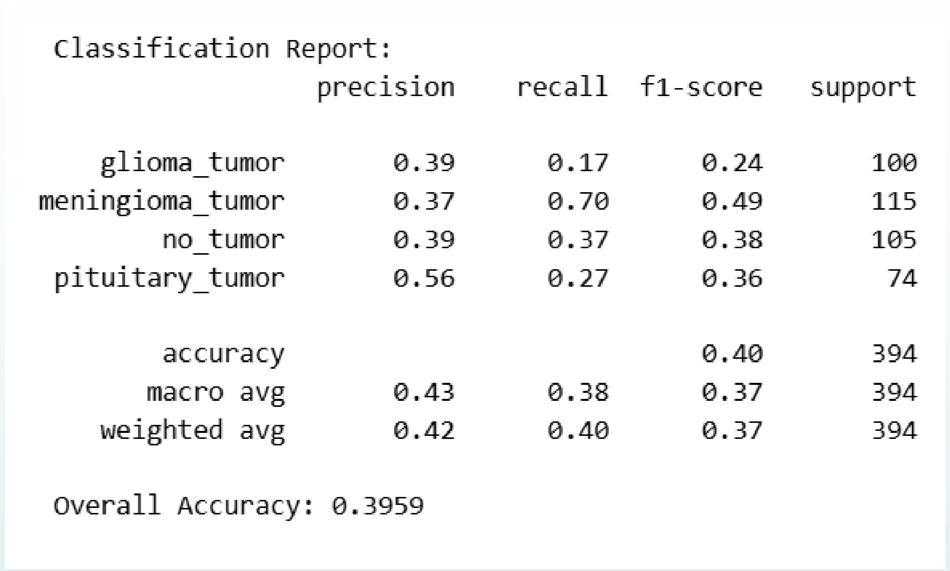
Example of test image index 200

- Visualizes how much attention each brain region (node) received during classification.
- Node numbers correspond to patches of the MRI image that were converted into a graph structure.
- Each node represents a brain region; edges show feature or spatial relationships.
- Thicker edges indicate stronger learned relationships or higher attention between connected regions.
- Brighter or more highlighted nodes represent areas with greater feature importance in prediction.

8.EVALUATION



True \ Predicted	glioma_tumor	meningioma_tumor	no_tumor	pituitary_tumor
glioma_tumor	17	55	17	11
meningioma_tumor	7	80	26	2
no_tumor	14	49	39	3
pituitary_tumor	6	30	18	20



```
Classification Report:
                  precision    recall  f1-score   support

    glioma_tumor       0.39      0.17      0.24       100
meningioma_tumor       0.37      0.70      0.49       115
        no_tumor       0.39      0.37      0.38       105
 pituitary_tumor       0.56      0.27      0.36        74

        accuracy                           0.40       394
       macro avg       0.43      0.38      0.37       394
    weighted avg       0.42      0.40      0.37       394

Overall Accuracy: 0.3959
```

9.RESULTS

- The model performs best for meningioma detection,
- The struggles to differentiate gliomas and pituitary tumors.
- There is a need for more balanced data.
- Moderate performance as some confusion with tumor classes.
- Good precision for pituitary tumors, when predicted, it’s often correct.

10.CONCLUSION

- GATv2-based GNNs can effectively model spatial and relational information in brain MRI data by treating image patches or regions as graph nodes.
- The attention mechanism provides an interpretable framework, identifying key brain regions influencing classification.
- Residual connections ensure smooth gradient flow, faster convergence, and stable multi-layer graph learning.
- The method demonstrates potential as an explainable AI (XAI) tool for clinical diagnosis and decision support.

11.APPLICATIONS

- Assisting radiologists with interpretable tumor detection.
- Understanding tumor progression and spatial dependencies.
- Identifying region-specific biomarkers from MRI scans.
- Deployable model for real-time, graph-based MRI classification.
- Visualizing node importance to ensure model transparency and trustworthiness.

Thank you