Brain Tumor Segmentation

# Introduction

Brain tumors are life-threatening conditions that require timely and accurate diagnosis. Manual interpretation of MRI scans is time-consuming and prone to human error. This project aims to automate the classification of brain tumors using deep learning, specifically CNNs, to aid radiologists and medical professionals.

# Dataset

The dataset contains MRI images categorized into the following classes:  
- glioma  
- meningioma  
- pituitary  
- notumor  
  
Ensure your dataset is located at:  
/content/drive/My Drive/Brain Tumor Segmentation/Training/  
  
Each class should be in its own folder.

# Model Architecture

The CNN model used has the following layers:  
- Conv2D layers with ReLU activation  
- MaxPooling to downsample feature maps  
- BatchNormalization for stable learning  
- GlobalAveragePooling to reduce dimensionality  
- Dense layers with dropout to reduce overfitting  
- Softmax output for multi-class classification  
  
Compiled with:  
- Loss function: categorical\_crossentropy  
- Optimizer: Adam  
- Metrics: accuracy

# Installation

Install required dependencies using pip:  
pip install numpy opencv-python-headless scikit-learn tensorflow gradio  
  
If using Google Colab, the project is already optimized for cloud execution.

# Usage

1. Mount Google Drive:  
from google.colab import drive  
drive.mount('/content/drive')  
  
2. Train the model: Run the notebook/script. Training runs for 10 epochs with image size 128x128.  
  
3. Launch Gradio interface:  
interface.launch()  
  
This opens a web interface for real-time classification.

# Results

The model achieved promising accuracy on the test set. It is capable of correctly classifying most MRI images across all four tumor categories.

# Limitations

- Does not localize the tumor—only classifies the image.  
- Limited by dataset quality and size.  
- Lacks explainability (e.g., heatmaps).  
- Not ready for clinical deployment without validation.

# Future Scope

- Integrate tumor segmentation models (e.g., U-Net).  
- Use pretrained models like ResNet/EfficientNet.  
- Add explainability using Grad-CAM.  
- Support mobile/edge deployment with TensorFlow Lite.  
- Perform clinical testing for real-world use.

# Technologies Used

- Python  
- TensorFlow/Keras  
- OpenCV  
- NumPy  
- Scikit-learn  
- Gradio  
- Google Colab

# Author

Developed as part of an academic project on deep learning in medical imaging.  
For educational purposes only — not for medical diagnosis.