**Brain Tumor Segmentation**

**1.Introduction**

Brain tumors are life-threatening conditions that require accurate and early diagnosis for effective treatment. Manual segmentation of MRI images is time-consuming and prone to human error. This project aims to automate brain tumor segmentation using a combination of traditional image processing (OpenCV) and deep learning (U-Net) techniques to assist radiologists in accurate tumor detection and visualization.

**2.Problem Statement**

**What the Project Does:**

This project segments brain tumors from MRI images using image processing and deep learning to generate accurate binary masks of tumor regions.

**Objective:**

* Automatically identify and outline tumor regions in MRI brain scans.
* Provide a tool that allows medical practitioners to visualize and download the segmented results.
* Enable easy interaction through a simple web interface using **Streamlit**.

**3. Limitations of the Existing System**

* Manual tumor annotation is labor-intensive and error-prone.
* Traditional segmentation methods lack generalization and robustness.
* Existing solutions often require high-end computational resources and complex installations.
* Lack of a user-friendly deployment platform for end-users.

**4.Proposed System**

**Solutions:**

* Automate segmentation using deep learning (U-Net) trained on labeled MRI images.
* Combine OpenCV-based preprocessing and visualization with deep learning inference.
* Provide a web-based interface via **Streamlit** for real-time usage.
* Ensure modular and lightweight implementation for ease of deployment.

## 5.Pipeline of the System

## Data Collection ➡ Preprocessing ➡ EDA ➡ Segmentation Model (Traditional + Deep Learning) ➡ Evaluation ➡ Streamlit UI ➡ Deployment

## 

## Testing & Implementation

* Model tested on unseen data.
* Evaluated using metrics:
  + Dice Coefficient
  + IoU (Jaccard Index)
  + Precision & Recall
  + Inference Time
  + Model Size

**Loss Evaluation**

* Tracked Dice and IoU scores on validation set.
* Compared results of traditional and deep learning methods.

**Coding**

# Install dependencies

!pip install streamlit pyngrok gradio

# Mount Google Drive

from google.colab import drive

drive.mount('/content/drive')

# Streamlit App Code (EDA)

%%writefile app.py

import streamlit as st

import os

import cv2

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from collections import Counter

st.title("Brain Tumor Segmentation EDA")

dataset\_path = st.text\_input("Enter dataset path", "/content/drive/MyDrive/Brain Tumor Segmentation")

img\_size = st.slider("Image Resize Dimension", min\_value=64, max\_value=256, value=128)

@st.cache\_data

def load\_data(path, img\_size):

categories = ['glioma', 'meningioma', 'notumor', 'pituitary']

data, labels = [], []

for category in categories:

folder\_path = os.path.join(path, category)

if not os.path.exists(folder\_path):

continue

for img\_file in os.listdir(folder\_path):

try:

img\_path = os.path.join(folder\_path, img\_file)

image = cv2.imread(img\_path)

image = cv2.resize(image, (img\_size, img\_size))

image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

data.append(image)

labels.append(category)

except:

continue

return np.array(data) / 255.0, np.array(labels), categories

data, labels, categories = load\_data(dataset\_path, img\_size)

st.success(f"Loaded {len(data)} images successfully!")

st.subheader("Class Distribution")

label\_counts = Counter(labels)

fig, ax = plt.subplots()

sns.barplot(x=list(label\_counts.keys()), y=list(label\_counts.values()), palette="Set2", ax=ax)

ax.set\_xlabel("Tumor Type")

ax.set\_ylabel("Image Count")

st.pyplot(fig)

else:

st.warning("Please provide a valid dataset path.")

# Run Streamlit App

!streamlit run app.py &

# Create Ngrok Tunnel

from pyngrok import ngrok

ngrok.set\_auth\_token("2sL1tv65IJMYCl4RzuFfsELK7n6\_7vUQKdvmz8W7Q3yZfvCoM")

public\_url = ngrok.connect(8501)

print(f"Streamlit app is live at: {public\_url}")

# Gradio + Model Training App

import gradio as gr

import os

import cv2

import numpy as np

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Conv2D, MaxPooling2D, BatchNormalization, Dropout, Dense, GlobalAveragePooling2D

from tensorflow.keras.optimizers import Adam

from sklearn.preprocessing import LabelBinarizer

from sklearn.model\_selection import train\_test\_split

IMG\_SIZE = 128

BATCH\_SIZE = 16

EPOCHS = 10

DATASET\_PATH = '/content/drive/MyDrive/Brain Tumor Segmentation/Training'

CATEGORIES = ['glioma', 'meningioma', 'notumor', 'pituitary']

data = []

labels = []

for category in CATEGORIES:

folder\_path = os.path.join(DATASET\_PATH, category)

print(f"Loading {category}...")

for img in os.listdir(folder\_path):

img\_path = os.path.join(folder\_path, img)

try:

image = cv2.imread(img\_path)

image = cv2.resize(image, (IMG\_SIZE, IMG\_SIZE))

data.append(image)

labels.append(category)

except Exception as e:

pass

data = np.array(data) / 255.0

lb = LabelBinarizer()

labels = lb.fit\_transform(labels)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(data, labels, test\_size=0.2, random\_state=42)

model = Sequential([

Conv2D(32, (3, 3), activation='relu', padding='same', input\_shape=(IMG\_SIZE, IMG\_SIZE, 3)),

MaxPooling2D((2, 2)),

BatchNormalization(),

Conv2D(64, (3, 3), activation='relu', padding='same'),

MaxPooling2D((2, 2)),

BatchNormalization(),

Conv2D(128, (3, 3), activation='relu', padding='same'),

MaxPooling2D((2, 2)),

BatchNormalization(),

GlobalAveragePooling2D(),

Dense(128, activation='relu'),

Dropout(0.5),

Dense(len(CATEGORIES), activation='softmax')

])

model.compile(loss='categorical\_crossentropy', optimizer=Adam(), metrics=['accuracy'])

history = model.fit(X\_train, y\_train, batch\_size=BATCH\_SIZE, validation\_data=(X\_test, y\_test), epochs=EPOCHS, verbose=1)

def predict\_image(image):

image = cv2.cvtColor(np.array(image), cv2.COLOR\_RGB2BGR)

image = cv2.resize(image, (IMG\_SIZE, IMG\_SIZE))

image = np.expand\_dims(image, axis=0) / 255.0

prediction = model.predict(image)

label = lb.inverse\_transform(prediction)

return label[0]

interface = gr.Interface(fn=predict\_image,

inputs=gr.Image(type='pil'),

outputs=gr.Label(num\_top\_classes=4),

title="Brain Tumor Classifier",

description="Upload an image to classify a brain tumor as glioma, meningioma, notumor, or pituitary.")

interface.launch()

**6. Dashboard / User Interface**

* Built using **Streamlit**.
* Allows MRI image uploads.
* Displays:
  + Original MRI image
  + Ground truth mask (if available)
  + Predicted segmentation
* Option to **download the result**.

**7.Libraries & Architecture**

**Libraries Used:**

* OpenCV: Image processing & visualization
* NumPy: Matrix operations
* Pandas: Data analysis
* Matplotlib / Seaborn: Plotting
* TensorFlow/Keras: Deep learning model (U-Net)
* Streamlit: Web interface
* Scikit-learn: Metrics and splitting

**Deep Learning Architecture:**

* **U-Net**
  + Encoder-Decoder structure
  + Skip connections for feature recovery
  + Optimized for biomedical image segmentation

**8. Project Limitations**

* Model may not generalize well to other imaging modalities (e.g., CT).
* Accuracy may decrease on low-quality or noisy images.
* Requires labeled training data (masks), which are difficult to obtain.
* Real-time inference may be slower on CPU-only machines.

**9. Future Enhancements**

* Integrate 3D U-Net for volumetric MRI data.
* Expand dataset to include various types of tumors.
* Use transfer learning from medical segmentation models.
* Deploy as a standalone desktop or mobile app.
* Add explainability module (e.g., Grad-CAM) to show decision rationale.

**10. Conclusion**

This project demonstrates an efficient pipeline for brain tumor segmentation using both traditional and deep learning methods. The solution significantly reduces manual effort, provides accurate segmentation, and includes a user-friendly deployment using Streamlit. It is an essential step toward AI-assisted medical diagnostics.