Frameworks used in detecting tumors using image processing

1-Tensorflow

2-Pytorch

3-Opencv

4-Keras

5-Monai

6-simpleITK

7-scikit-image

Creating a hybrid framework

By merging two existing frameworks and integrating their functionalities to leverage the strengths of both.

Steps to merge

1-Preprocessing with opencv

# Task-Using open cv to preprocess the medical images

## Advancement in image :

## 1-enhancing image: applying different filters like histogram equalization or CLAHE to enhance image.

## 2-Segmentation: dividing image into diffrenet segments  for isolating region of interest.

## 3-Resizing and normalisation: resizing the image to the input expected in tensorflow and normalize the pixel values.

## CODE for preprocessing with open cv

```
import cv2

import numpy as np


# Load the image

image = cv2.imread('tumor_scan.png', cv2.IMREAD_GRAYSCALE)


# Apply Gaussian blur

blurred_image = cv2.GaussianBlur(image, (5, 5), 0)


# Apply thresholding

_, segmented_image = cv2.threshold(blurred_image, 120, 255, cv2.THRESH_BINARY)
```

```
# Resize image
resized_image = cv2.resize(segmented_image, (224, 224))

# Normalize the image
normalized_image = resized_image / 255.0
```

## Modelling with tensorflow

## Task :Using tensorflow to build and train a deep learning model for tumor detection

## Advancements

## 1-Model input: feed the preprocessed  image from opencv into tensorFlow Model.

## 2-Model Architecture: Using CNN to classify or segment tumors.

## 3:training:Train the model on label medical image.

## 4:Evaluation:Evaluating its performance

Code

```
import tensorflow as tf
from tensorflow.keras import layers, models

# Define the CNN model
model = models.Sequential([
    layers.Conv2D(32, (3, 3), activation='relu', input_shape=(224, 224, 1)),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(128, (3, 3), activation='relu'),
```

```
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Assuming `normalized_image` is part of a dataset
model.fit(normalized_image, labels, epochs=10, validation_split=0.2)
```

# 3-Integration

Task: combine the preprocessing pipeline with the tensorflow model in a seamless workflow.

Advancements:

End to end pipeline: create a function that takes raw images, preprocess them using opecv and then forward it to tensorflow

For prediction.

2-Deployment: deploy this hybrid framework as a standalone app.

Code

```
def tumor_detection_pipeline(image_path):
    # Step 1: Preprocess using OpenCV
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    blurred_image = cv2.GaussianBlur(image, (5, 5), 0)
    _, segmented_image = cv2.threshold(blurred_image, 120, 255, cv2.THRESH_BINARY)
    resized_image = cv2.resize(segmented_image, (224, 224))
    normalized_image = resized_image / 255.0
    normalized_image = np.expand_dims(normalized_image, axis=-1)  # Add channel dimension

    # Step 2: Predict using TensorFlow model
    prediction = model.predict(np.expand_dims(normalized_image, axis=0))

    return prediction

# Example usage
result = tumor_detection_pipeline('tumor_scan.png')
print("Tumor detected" if result > 0.5 else "No tumor detected")
```