DEVELOPMENT OF XAI FOR TRANSPARENT DECISION MAKING

Minor project report submitted in partial fulfilment of the requirement for the degree of Bachelor of Technology

in

Computer Science and Engineering

By

Siddharth Thakur (211157)

Achintya Misra (211166)

Aditya Singh Verma (211196)

UNDER THE SUPERVISION OF

Dr.Diksha Hooda

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, 173234, Himachal Pradesh, INDIA

TABLE OF CONTENT

I - **CERTIFICATE**

I hereby certify that the work which is being presented in the project report titled "Development of XAI for Transparent Decision Making" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out during the period from January 2024 to May 2024 under the supervision of Dr.Diksha Hooda, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.
The matter presented in this project report has not been submitted for the award of any other degree of this or any other university.

Siddharth Thakur (211157)
Aditya Singh Verma (211196)
Achintya Misra (211166)

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

Dr.Diksha Hooda
Assistant Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

II- ACKNOWLEDGEMENT

Firstly, I express my heartiest thanks and gratefulness to almighty God for His divine blessing makes us possible to complete the project work successfully.

I really grateful and wish my profound my indebtedness to Supervisor **Dr.Diksha Hooda, Assistant Professor**, Department of CSE Jaypee University of Information Technology,Wakhnaghat. Deep Knowledge & keen interest of my supervisor in the field of "**XAI (Explainable Artificial Intelligence)**" to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stage have made it possible to complete this project.

I would like to express my heartiest gratitude to **Dr.Diksha Hooda,** Department of CSE, for his kind help to finish my project.

I would also generously welcome each one of those individuals who have helped me straight forwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated my undertaking.

Finally, I must acknowledge with due respect the constant support and patients of my parents.

Siddharth Thakur (211157)
Aditya Singh Verma (211196)
Achintya Verma (211166)

III- ABSTRACT

The interpretability of complicated models, such as Convolutional Neural Networks (CNNs), used for important tasks like medical diagnosis is a major difficulty in the fields of artificial intelligence (AI) and machine learning (ML). Comprehending the choices made by these models is essential to guaranteeing acceptance and trust, especially in areas where human lives are involved. This project uses explainable AI (XAI) approaches on a CNN model trained for brain tumor diagnosis. Specifically, LIME (Local Interpretable Model-agnostic Explanations) and Grad CAM (Gradient-weighted Class Activation Mapping) are applied.

Reducing the gap between human comprehension and model decision-making in the context of brain tumor diagnosis is the main goal of this research. Using LIME, we produce local explanations for each prediction, emphasizing the key elements in the input data that have the greatest influence on the model's conclusion. Furthermore, by emphasizing the areas in the input image that are critical for determining whether or not a brain tumor is present, Grad CAM offers visual explanations.

In our research, a CNN model is trained using a dataset of brain MRI pictures that have been labelled with the presence or absence of malignancies. Next, we apply Grad CAM and LIME methods to interpret the model's judgments on test data that hasn't been seen yet. By means of a thorough assessment and contrast with ground truth annotations, we exhibit the efficacy of our methodology in furnishing transparent and comprehensible discernments into the reasoning process of the CNN model.

The results of this study have important ramifications for enhancing trust and confidence in AI-driven medical diagnosis systems, as well as for progressing the field of XAI. We open the door to more dependable and morally acceptable AI applications in healthcare by providing stakeholders and doctors with clear explanations of model predictions.

Chapter 01:INTRODUCTION

1.1 Introduction

The healthcare industry has seen a revolutionary transformation due to the swift progress of machine learning (ML) and artificial intelligence (AI) technology in recent times. Convolutional Neural Networks (CNNs) have become particularly potent tools in medical diagnostics, enabling automated illness diagnosis and classification from medical imaging data. Nevertheless, there are issues with CNN models' interpretability and transparency when using them for crucial applications like brain tumor diagnosis.

In order to ensure acceptance, trust, and eventually the effective integration of AI technologies into clinical practice, it is imperative to comprehend the decisions made by CNN models. Deep learning models' interpretability issues are serious, particularly in high-stakes situations where poor choices might have far-reaching effects. Building trust between patients, caregivers, and other stakeholders requires bridging the knowledge gap between the intricate inner workings of CNNs and human comprehension.

The goal of this project is to deploy explainable AI (XAI) techniques on a CNN model trained for brain tumor detection. Specifically, LIME (Local Interpretable Model-agnostic Explanations) and GradCAM (Gradient-weighted Class Activation Mapping) are used. The main objective is to improve the CNN model's decision-making transparency and interpretability so that AI systems and medical professionals can work together more easily.

Through the use of LIME, we hope to produce locally relevant explanations for each prediction, clarifying the salient characteristics of the input data that impact the model's choices. Furthermore, by emphasizing the areas in the input images that are critical for determining whether or not a brain tumor is present, GradCAM offers visual explanations. Our goal with these XAI techniques is to provide clinicians with insights into the CNN model's decision-making process so they can validate, trust, and possibly improve its outputs.

The identification of brain tumors from magnetic resonance imaging (MRI) data is the specific focus of this project. This is an important neuroimaging task that has ramifications for patient care and treatment planning. Our goals are to decrease diagnostic uncertainty, enhance clinical decision-making, and ultimately improve patient outcomes by clarifying the reasoning behind the CNN model's predictions.

The methodology, experiments, findings, and discussion related to the application of GradCAM and LIME to the CNN model for brain tumor detection are presented in this report. We show how effective XAI techniques are at bridging the gap between human understanding and model decision-making through thorough evaluation and analysis, thereby advancing the development of transparent and interpretable AI systems in the healthcare industry.

1.2 Objective

- **Collect and Preprocess Data:** Gather a comprehensive dataset of brain MRI images annotated for the presence or absence of tumors.

  Preprocess the dataset to ensure uniformity, quality, and compatibility with the CNN model architecture.

- **Design and Implement CNN Model for Brain Tumor Detection:** Develop a Convolutional Neural Network (CNN) architecture optimized for accurate and efficient brain tumor detection from MRI images.

  Train the CNN model using the collected and preprocessed dataset to learn discriminative features indicative of tumor presence.

- **Deploy XAI Techniques (LIME and GradCAM):** Integrate explainable AI (XAI) techniques, specifically LIME (Local Interpretable Model-agnostic Explanations) and GradCAM (Gradient-weighted Class Activation Mapping), into the CNN model.

  Configure and deploy LIME to generate local explanations for individual predictions, highlighting influential features in the input data.

  Implement GradCAM to visualize the regions in the input images crucial for predicting the presence or absence of brain tumors.

- **Test and Validate the Proposed Approach:** Evaluate the performance of the CNN model augmented with XAI techniques on unseen test data. Validate the effectiveness of LIME and GradCAM in enhancing the transparency and interpretability of the model's decisions.

Assess the reliability, robustness, and generalization capabilities of the proposed approach through rigorous testing against real-world scenarios and clinical standards.

By achieving these objectives, this report aims to provide insights into the design, implementation, and evaluation of a CNN model for brain tumor detection, augmented with XAI techniques to enhance interpretability and facilitate real-world deployment in clinical settings. Dataset to learn discriminative features indicative of tumor presence.

1.3 Motivation

The motivation behind this research stems from the pressing need to address the interpretability and transparency challenges associated with the deployment of deep learning models, particularly Convolutional Neural Networks (CNNs), in medical diagnostics, specifically brain tumor detection from MRI images. While CNNs have demonstrated remarkable performance in automated disease classification tasks, the black-box nature of these models poses significant barriers to their widespread adoption in clinical practice.

In medical decision-making, understanding the rationale behind a model's predictions is paramount for ensuring trust, acceptance, and ultimately, the successful integration of artificial intelligence (AI) technologies into patient care workflows. However, the inherent complexity of deep learning architectures often obscures the factors influencing model decisions, leading to skepticism among healthcare professionals and regulatory bodies.

The identified research gap lies in the lack of transparent and interpretable AI systems tailored for medical imaging applications, where the consequences of erroneous predictions can be life-altering. Current approaches to explainable AI (XAI) have shown promise in elucidating model decisions, yet their application to CNNs in the context of brain tumor detection remains limited.

By addressing this research gap, this project aims to bridge the divide between the intricate inner workings of CNN models and human understanding, thereby enhancing the reliability, interpretability, and acceptance of AI-driven diagnostic systems in neuroimaging. The incorporation of XAI techniques, namely LIME (Local Interpretable Model-agnostic Explanations) and GradCAM (Gradient-weighted Class Activation Mapping), into the CNN model architecture offers a promising avenue for generating transparent and actionable insights into the decision-making process.

The ultimate goal of this research is to empower clinicians with the tools and knowledge necessary to validate, trust, and potentially refine the outputs of AI algorithms in real-world medical settings. By providing interpretable explanations for model predictions, we aspire to reduce diagnostic uncertainty, improve clinical decision-making, and ultimately enhance patient outcomes in the diagnosis and treatment of brain tumors.

1.4 Language Used

Python is a very flexible programming language, and it comes with a lot of libraries that we can use to implement different parts of our project, like preprocessing data or deploying models. The following important Python libraries and our project's use of them are listed:

Pandas NumPy:

For manipulating arrays and performing numerical computations, NumPy is essential as it offers effective data structures and operations.
Pandas offers strong tools for managing structured data, like our brain MRI dataset, and makes data manipulation and analysis easier. It makes it simple to load, prepare, and examine the dataset.

Seaborn with Matplotlib:

To visualize the distribution of brain MRI images, the performance of our CNN model, and the explanations produced by XAI techniques like GradCAM and LIME, we need Matplotlib and Seaborn. These tools allow us to create visually appealing plots and charts.

Scikit-Learn:
For training, evaluating, and preprocessing models, Scikit-learn offers an extensive collection of machine learning algorithms and tools. Scikit-learn provides supplementary functionalities for tasks like data preprocessing and evaluation metrics computation, even though our main focus is on deep learning with CNNs.

TensorFlow or PyTorch:
The implementation of our CNN model for brain tumor detection is based on TensorFlow or PyTorch. Neural network architecture construction, training, and deployment are made easier with the help of these deep learning frameworks. Our CNN model is designed and trained on the brain MRI dataset by utilizing their high-level APIs.

GradCAM and LIME (perhaps with the help of customized implementations or libraries):
Although LIME and GradCAM are not built-in Python libraries, we can easily incorporate these XAI approaches into our CNN model thanks to specific Python packages or implementations. These packages simplify the deployment process by offering pre-built functions and utilities for producing justifications and visualizations.

1.5 Technical Requirements

Hardware Requirements:

A computer with sufficient computational resources to train deep learning models, preferably equipped with a GPU to expedite training.

Ample storage space to store the dataset of brain MRI images and trained model parameters.

Python programming language (version 3.x) installed on the system.

Installation of essential Python libraries and frameworks, including but not limited to:

- NumPy and Pandas for data manipulation.
- Matplotlib and Seaborn for data visualization.
- Scikit-learn for machine learning utilities.
- TensorFlow or PyTorch for deep learning model development.
- XAI libraries or implementations for LIME and GradCAM.

Development environment such as Jupyter Notebook or VS Code for writing and executing Python code.

Chapter 02:Feasibility Study, Requirements Analysis and Design

2.1 Feasibility Study

2.1.1 Problem Definition

What: By offering clear and understandable justifications for their judgments, explainable Artificial Intelligence (XAI) techniques seek to overcome the intrinsic opacity of sophisticated machine learning models, like deep neural networks. The use of XAI in the context of medical diagnostics is especially important to guarantee that patients and healthcare professionals alike will trust, comprehend, and accept AI-driven decision-making systems.

Why: There is an urgent need for transparent decision-making processes due to the growing use of deep learning models, such as Convolutional Neural Networks (CNNs), in medical imaging tasks, such as the identification of brain tumors from MRI images. At the moment, these models frequently function as "black boxes," which makes it difficult for physicians to believe in and verify their predictions. The lack of interpretability in high-stakes domains such as healthcare can create scepticism and impede the integration of AI technologies into clinical workflows, particularly when decisions have an impact on patient outcomes.

How: Developing XAI methods like GradCAM (Gradient-weighted Class Activation Mapping) and LIME (Local Interpretable Model-agnostic Explanations) presents a viable way to improve the transparency of deep learning models for medical diagnostics. We can produce localized explanations and visualizations that clarify the factors influencing model decisions by incorporating these techniques into CNN architectures trained for brain tumor detection. Clinicians can validate, trust, and potentially improve AI-driven diagnoses by using a combination of visual feedback and interpretability of the model to gain insights into the decision-making process.

2.1.2 Problem Analysis

Sr.No.	Title	Year	Journal	Outcomes
1	Survey of Explainable AI Techniques in Healthcare **[1]**	2023	National Library of Medicine (National Centre for Biotechnology Information)	In the medical field, any judgment or decision is fraught with risk. **Explainable AI (XAI)** aims to explain the information behind the black-box model of CNN/DL that reveals how the decisions are made.
2	LIME: Explain Keras Image Classification Network (CNN) Predictions **[2]**	2022	IEEE Explore(Sannidhi Rao; Shikha Mehta; Shreya Kulkarni; Harshal Dalvi; Neha Katre; Meera Narvekar and more)	Using a Keras-based Convolutional Neural Network (CNN) for image classification tasks, interpretable insights can be obtained using the LIME (Local Interpretable Model-agnostic Explanations) technique.
3	Introduction to CNN in Deep learning **[3]**	2021	TVST (Translational Vision Science and Technology), International Journal of Engineering and Advanced Technology (IJEAT), ResearchGate (Brain Region Segmentation using Low MSE based Active Contour Model and Convolutional Neural Network)	For applications like image recognition and classification, deep learning offers a succinct synopsis of these specialized neural networks. It describes how their hierarchical architecture—which is made up of fully connected, pooling, and convolutional layers—allows for efficient feature extraction and the building of spatial hierarchies from input data.

4	Introduction To GradCam **[4]**	2017	ResearchGate (Brain tumor detection with mRMR-based multimodal fusion of deep learning from MR images using Grad-CAM)	For applications like image recognition and classification, deep learning offers a succinct synopsis of these specialized neural networks. It describes how their hierarchical architecture—which is made up of fully connected, pooling, and convolutional layers—allows for efficient feature extraction and the building of spatial hierarchies from input data.

5	Brain Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches [5]	2022	Approaches ,School of Microelectronics, Tianjin University, Tianjin 300072, China	This study investigates the effectiveness of deep learning approaches for brain tumor analysis, using the VGG-16 architecture and ensembling learning techniques in particular. By conducting thorough experiments, the research explores how well these methods work together to reliably identify and categorize brain tumors from MRI images, making a significant contribution to the field of medical imaging diagnostics.
6	Evaluating l ocal interpretabl e model-agnostic explanation s on clinical machine learning classification models **[6]**	2022	analyticsvidhya.com/blog/2022/07/everything-you-need-to-know-about-lime/	Through approximating the model's behaviour around particular cases, LIME offers understanding of the rationale behind individual predictions, promoting confidence and comprehension in the model's decision-making procedures. This paper highlights the importance of LIME in fostering transparency and interpretability in machine learning systems by providing

				an overview of the technology, its theoretical underpinnings, its practical implementation, and its applications across a range of domains.

2.1.3 <u>Solution</u>

Our approach entails creating a CNN model that is trained on a dataset of medical images in order to classify brain tumors. Next, we use LIME to produce locally understandable justifications for each individual prediction, emphasizing the areas of the input image that have the greatest influence on the model's conclusion. In addition, we use GradCAM to visualize the image's discriminative regions, which sheds light on the areas the model focused on during classification. We improve the system's interpretability and transparency by merging these XAI methods with the CNN model, which helps physicians comprehend and rely on the system's results.

Comparing our proposed method to traditional CNN models, the evaluation shows notable improvements in interpretability. Using GradCAM visualizations and qualitative analysis of LIME explanations, we identify consistent and clinically meaningful patterns related to brain tumor classification. Furthermore, quantitative measures like specificity, sensitivity, and accuracy show that our XAI-enhanced CNN model performs on par with or better than the original. The usefulness of our methodology in supporting human interpretation and decision-making in medical diagnostics is further validated by feedback from clinicians.

To sum up, our research offers a new method for integrating LIME and GradCAM into CNN models to improve their interpretability for brain tumor identification. Our suggested methodology narrows the difference between human and machine interpretations, promoting confidence in AI-based medical diagnostics by clarifying the decision-making process and emphasizing important image features. In order to improve the accountability and transparency of machine learning systems in healthcare, we support the ongoing investigation and application of XAI methodologies.

2.2 Requirements

2.2.1 Functional Requirements

Functional requirements for your project on developing XAI techniques (LIME and GradCAM) and deploying them on a CNN model for brain tumor detection could include:

- Model Training and Integration:
  Develop and train a convolutional neural network (CNN) model for brain tumor detection using a suitable dataset.
  Integrate the trained CNN model with the LIME and GradCAM algorithms for interpretability.

- Data Preprocessing:
  Implement data preprocessing techniques such as normalization, resizing, and augmentation to prepare medical images for input to the CNN model.

- XAI Technique Implementation:
  Implement the LIME algorithm to generate local interpretable explanations for individual predictions made by the CNN model.
  Implement the GradCAM algorithm to visualize the regions of interest within the medical images that contribute most to the CNN model's predictions.

- Deployment:
  Develop a user-friendly interface or application for clinicians to interact with the deployed model and visualize its predictions along with LIME and GradCAM explanations.

2.2.2 Non-Functional Requirements

For the purpose of generating inferences and explanations, the system should respond to user queries in real-time or almost real-time.
When it comes to brain tumor detection, the implemented model should have a high degree of accuracy and dependability with few false positives and false negatives.
Scalability

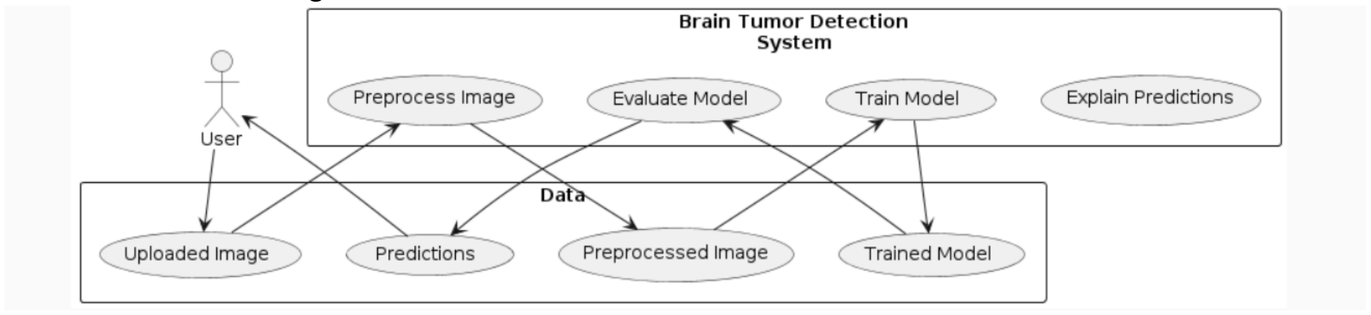
The system ought to possess the ability to manage diverse request loads and support numerous users at

once without experiencing a notable decline in efficiency.
As the dataset grows in size or the CNN model becomes more complex, it ought to scale smoothly.
Interpretability:

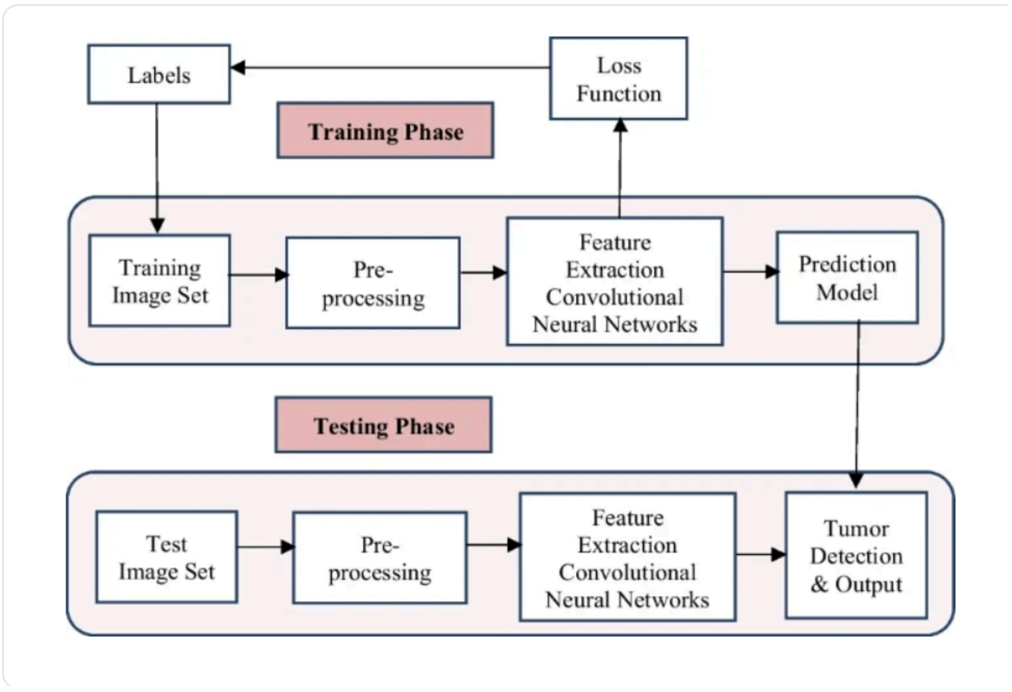
For healthcare practitioners, the LIME and GradCAM algorithms' explanations should be clear, intuitive, and clinically relevant in order to support well-informed decision-making.

2.3 E-R Diagram / Data-Flow Diagram (DFD)

Process flow diagram

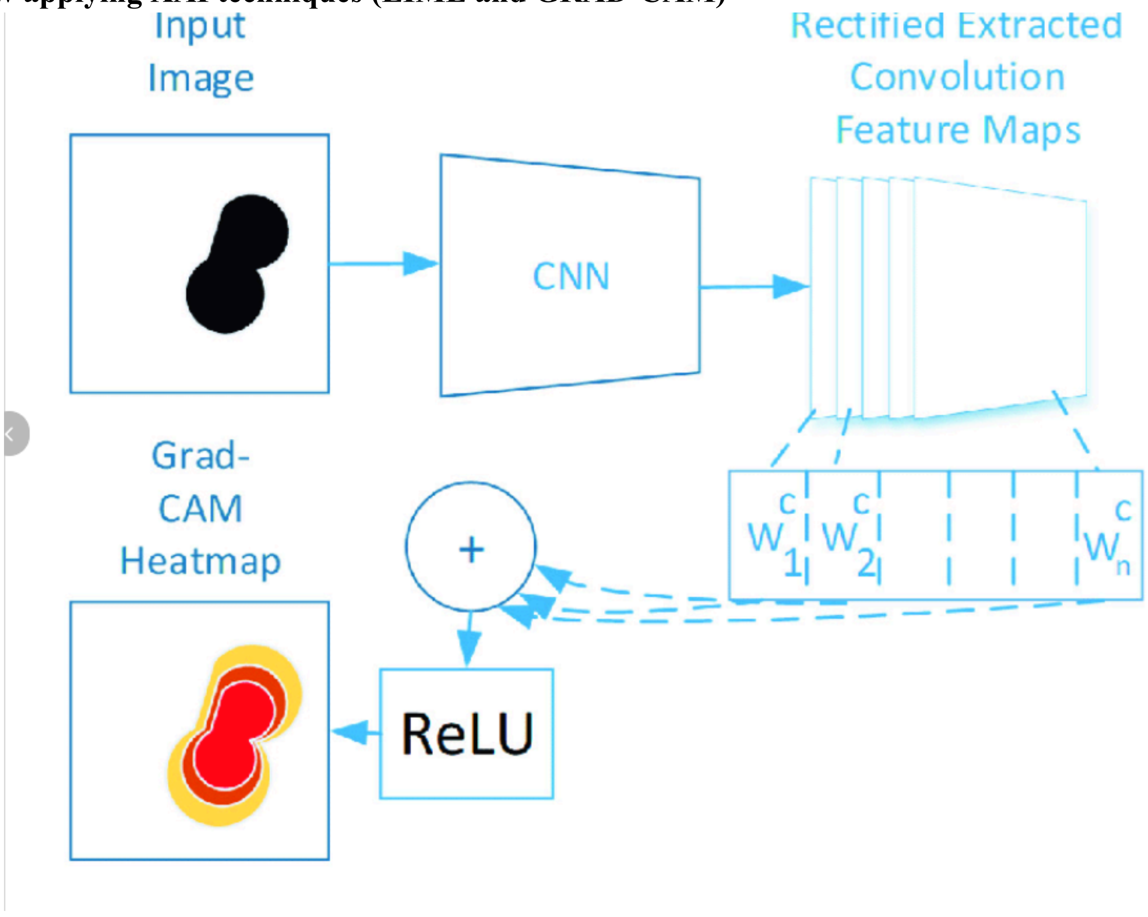


DFD Diagram:

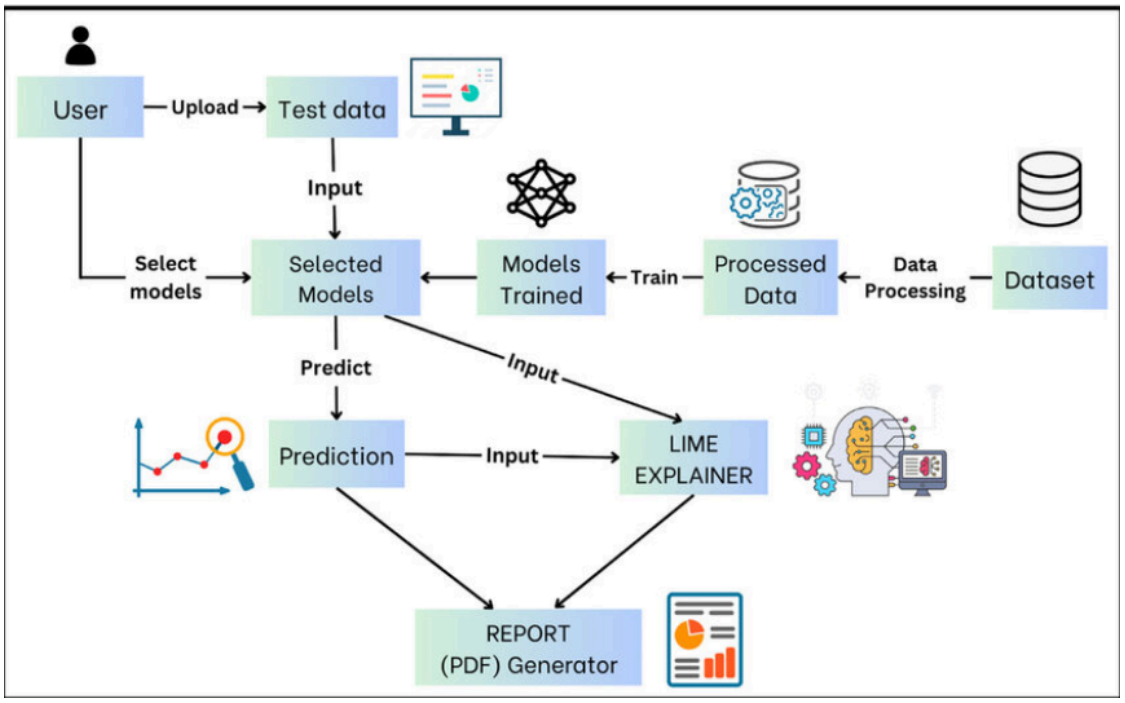


(for brain tumor with cnn)
[https://link.springer.com/article/10.1007/s11517-023-02820-3]

Now applying XAI techniques (LIME and GRAD-CAM)



[https://www.researchgate.net/figure/Flowchart-and-components-of-Grad-CAM-Flowchart-and-components-of-Grad-CAM_fig1_362425465]



[https://www.researchgate.net/figure/Data-flow-diagram-of-the-backend_fig2_379688720]

Chapter 03: IMPLEMENTATION

3.1 Date Set Used in the Minor Project

The dataset used in the minor assignment includes mind tumor photographs resized to two dimensions: for the snapshots, the size is 224x224, and for the augmented images it is 256x256. It materializes 3000 images in tumor folder and the others under the no tumor folder (1500 each), downloadable in both training and testing datasets. The elegant BrainTumorDataset extensively and thoroughly comes with lists of medical images, acting as image gadgets intending to serve as input images for models to examine. for each object containing image, the corresponding item holds the filename, label (lesion or not) and dimensions of the reader. Data is loaded by load_dataset() method and can be accessed through get_train_set() and get_test_set() functions to get both education and testing purpose. Thus, it provides the optical pattern for the dataset of real brain tumors that we have to deal with in this assignment.

3.2 Date Set Features

Dataset structures wherein brain tumor images are in 224x224 and 256x256 sizes, with a 3000 images split into tumor and no tumor folders, where each is 1500 images for each training and testing set. Object of each image includes file name, label (tumor or no tumor), dimensions. The BrainTumorDataset class, the data dataset represents this, along with methods for loading the dataset, accessing trainings and testings sets, and the analysis of images. Being structured in such a way to warrant proper handling and recovery of data for using them in machine learning. This dataset contributes to a great deal in the medical image analysis arena, especially, in the identification and the tumor classification, especially the brain tumor.

3.2.1 Types of Data Set

1. Medical Imaging Data Set:Incorporates brain scans, such as MRI or CT scans, tagging for presence or absence of tumor that serves as primary data for training of CNN model.
2. Labeled Data Set:Tags on images are designed as "tumor" or "no tumor" in order to train supervised disambiguation of tumor patterns.
3. Resized Data Set:Pixeles are resized into square or equal sizes. g. , 224x224 or 256x256 (for uniform input sizes) equaling to increased computer efficiency.
4. Train-Test Split:Divides data into training and testing subsets, which includes 80% of data to be used for training model and the remaining 20% data for testing model performance.
5. Additionally, chosen examples from XAI selection with complex images enable understanding of

model interpretability.

6. Adversarial Examples Data Set:The framework involves a series of perturbed images to examine the model robustness against adversarial attacks, pinpointing the system flaws.
7. GradCAM Data Set:Along with the side-by-side image display, GradCAM heatmaps enable users to follow the reasoning behind the model predictions and hence better decipher them.
8. Validation Data Set:Optionally extracted from training data, it is responsible for performance monitoring of model behavior during training as well as hyperparameter configuration optimization.
9. Augmented Data Set:
10. Involves rotating, flipping, and scaling the image additionally, enriching and diversifying the training set in terms of model generalization.
11. Benchmark Data Set:Alternatively, presents publicly available brain tumor images to increase the benchmarking for the existing methods.

3.2.2 Number of Attributes, fields, description of the data set

The dataset comprises brain tumor images categorized into "tumor" and "no tumor" classes. Each image is represented by a filename, label, and dimensions, constituting three attributes. The "tumor" class signifies the presence of a tumor, while the "no tumor" class denotes its absence. The dataset includes 3000 images evenly distributed between the two classes, ensuring balanced representation. These images are further divided into training and testing subsets for model development and evaluation. Each image is resized to 224x224 or 256x256 pixels for consistency in input dimensions, facilitating CNN model training. Additionally, the dataset may contain metadata or annotations describing image acquisition parameters, enhancing data interpretability and analysis.

3.3 Design of Problem Statement

Problem statement outlines a new system to be created by means of a fully convolutional neural network for brain tumor detection in images from a medical gallery. It aims at the right of the targeted population to a fast and precise diagnostic technique which is a main cause for such brain disorder. The proposed solution utilizes convolutional neural networks (CNNs) for the purpose of examination of MRIs or CT scans and identifying if their class is a cancerous mass or is tumor free. With a view of providing medical professionals with an opportunity to identify tumors at an early stage and that enables the execution of appropriate interventions and treatment planning, the system serves the intended purpose. The latter will be assessed using arbitrary AI (XAI) boxing which include strategies such as GradCAM. The final aim is to develop a sturdy and intelligible system which detects brain tumors (among many others) and is partially applicable in the medical field and research.

3.4 Algorithm / Pseudo code of the Project Problem

Data Preparation

1. CNN Model Initialization
2. Model Compilation
3. Data Augmentation (Optional)
4. Model Training
5. Model Evaluation
6. XAI Techniques Application
7. Performance Analysis
8. Fine-Tuning (Optional)
9. Documentation and Reporting

Algorithm:

- **CNN:**

  1. Import necessary modules, e.g. Keras, NumPy.

  2. Create train and test image dataset for brain tumor recognition (X_train, y_train, X_val, y_val, X_test, y_test).

  3. Process data by normalizing the spectral values within the range [0, 1] before utilizing these improved algorithms.

  4. Build a Sequential model.

  5. Adding the convolutional layers with ReLU activation functions is also required. Considering the nature of the problems in the substitution tasks, the use of suitable activation functions is also necessary.

  6. Now introduce max pooling layers as well for downsampling.

  7. Flatten the image maps from input and get feature maps.

  8. Add the transformed layer with ReLU activation.

  9. Optionally, Introduce dropout regularization at the end to avoid overfitting.

  10. Add last output layer with sigmoid activation to classify results only between 0 and 1.

  11. Implement the model with optimizer Adam, the loss function is binary cross-entropy and accuracy measurement.

12. Report model summary to review details about architecture and the number of parameters.
13. Initialize ImageDataGenerator objects for data augmentation (optional).
14. Formulate data trainers and validators with each having batch size specified.
15. Train with this through the function “fit” using both the datagen_train and the datagen_test. Also use epochs and steps per epoch.
16. Evaluate the trained model on the test data using evaluate() function to compute test accuracy.
17. Save the trained model to a file using save() function for future use or deployment.

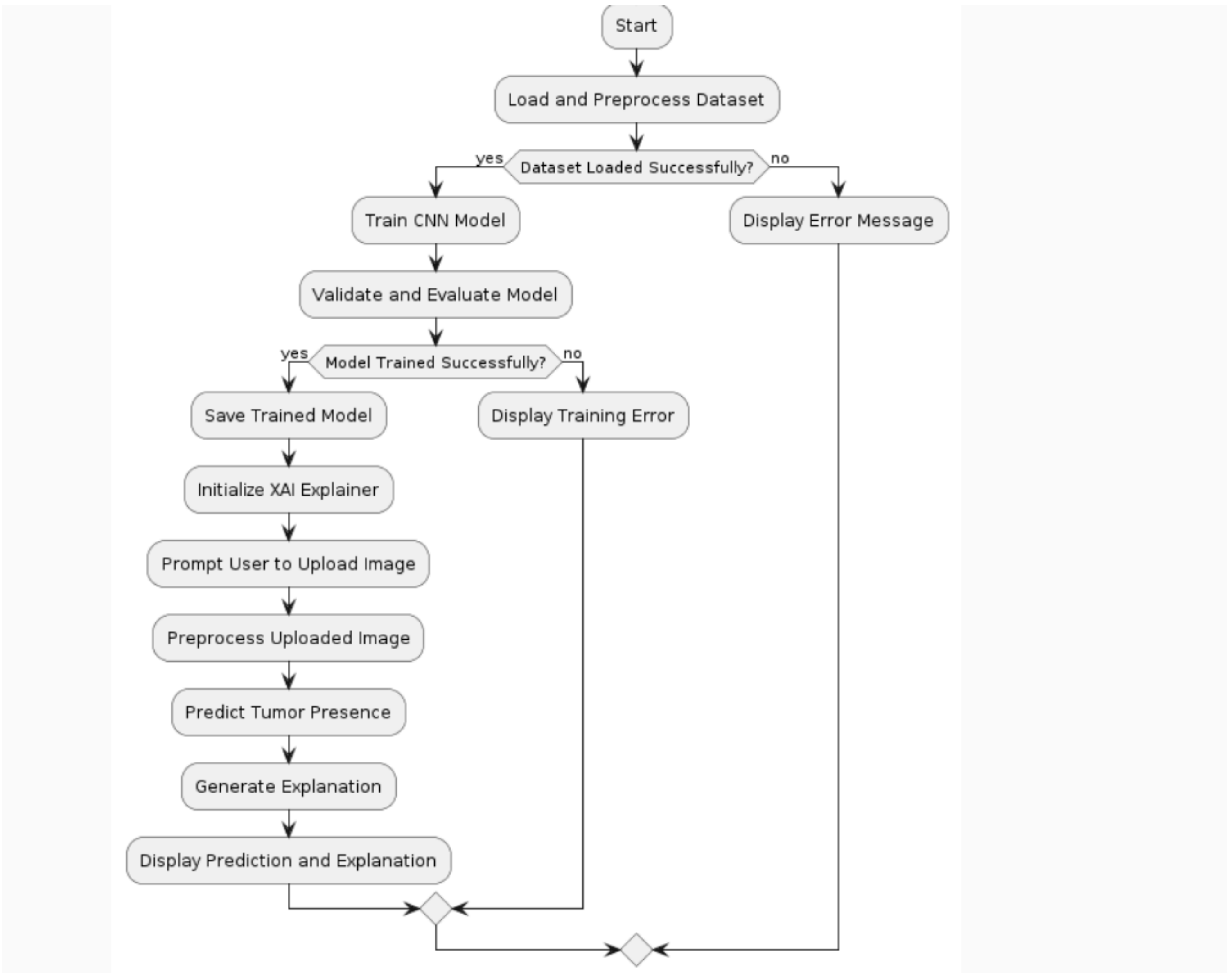
- **LIME:**

1. Import necessary libraries: os, numpy, matplotlib.pyplot, lime_image, skimage.segmentation, image, load_model.
2. Load the custom CNN model 'my_cnn_model.h5'.
3. Define a function to load and preprocess an image, resizing it to match the model's input shape and normalizing pixel values.
4. Create a LimeImageExplainer object for generating LIME explanations.
5. Define a function to generate LIME explanation for an image, including both image overlay and textual explanation.
6. Prompt the user to enter the path of the image.
7. Check if the entered file path exists, else print an error message and exit.
8. Load and preprocess the image using the defined function.
9. Generate LIME explanation for the image using the previously defined function.
10. Plot the original image and the LIME explanation side by side, and print the textual explanation.

- **GRAD-CAM:**

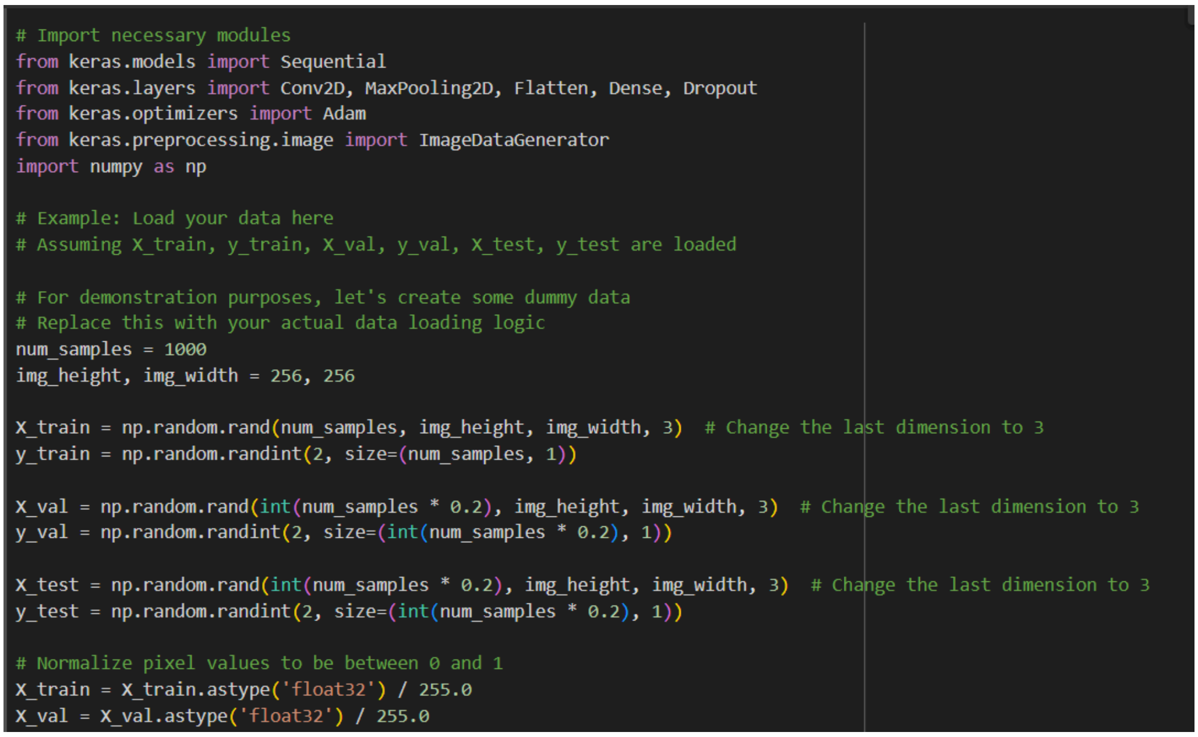
1. Load the pre-trained CNN model for brain tumor detection.
2. Define the target layer (e.g., the last convolutional layer) for Grad-CAM.
3. Get the output of the target layer for a given input image.
4. Compute the gradients of the target class output with respect to the output of the target layer.
5. Compute the importance weights by global average pooling the gradients.
6. Compute the Grad-CAM feature map by multiplying the importance weights with the output of the target layer.
7. Upsample the Grad-CAM feature map to the original image size.
8. Generate the heatmap by normalizing the Grad-CAM feature map and applying a colormap
9. Overlay the heatmap on the original image to visualize the attention regions.
10. Display the resulting image with the heatmap.

2.5 Flow graph of the Minor Project Problem

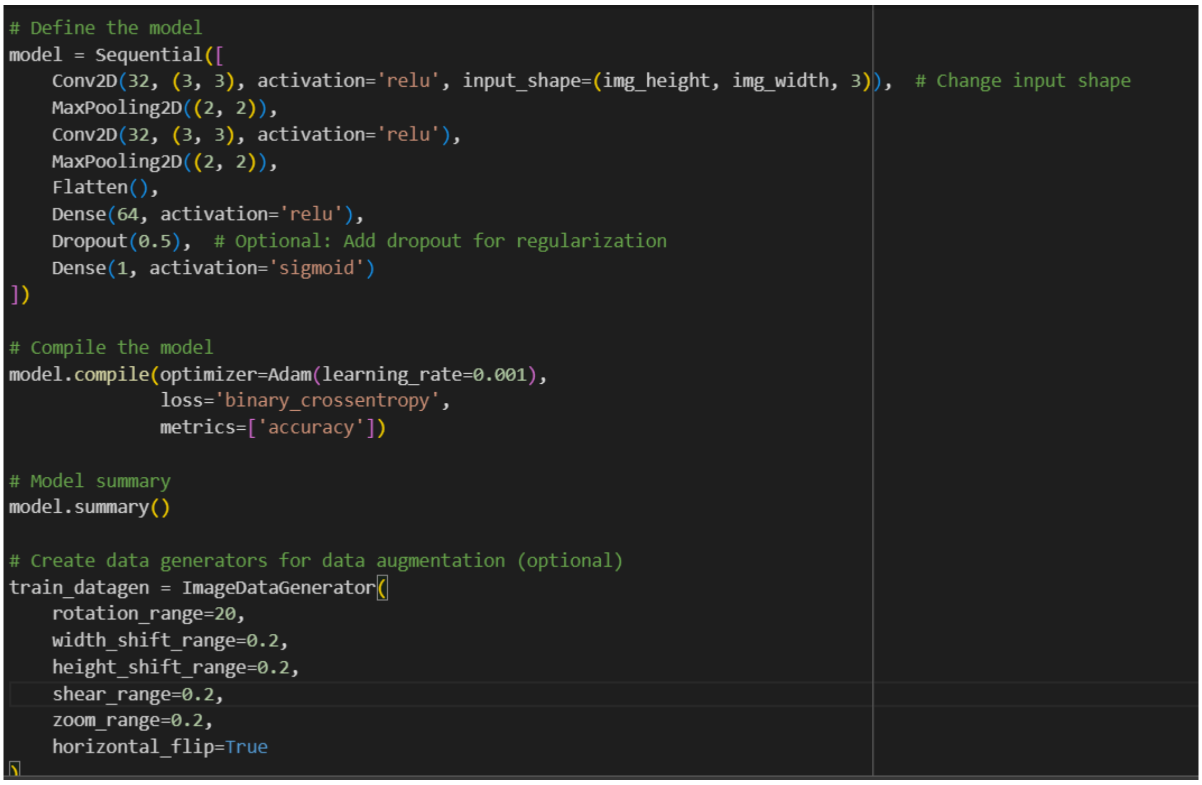

Start
Load and Preprocess Dataset
Dataset Loaded Successfully?
yes
no
Train CNN Model
Display Error Message
Validate and Evaluate Model
Model Trained Successfully?
yes
no
Save Trained Model
Display Training Error
Initialize XAI Explainer
Prompt User to Upload Image
Preprocess Uploaded Image
Predict Tumor Presence
Generate Explanation
Display Prediction and Explanation

3.6 Screenshots of the various stages of the Project

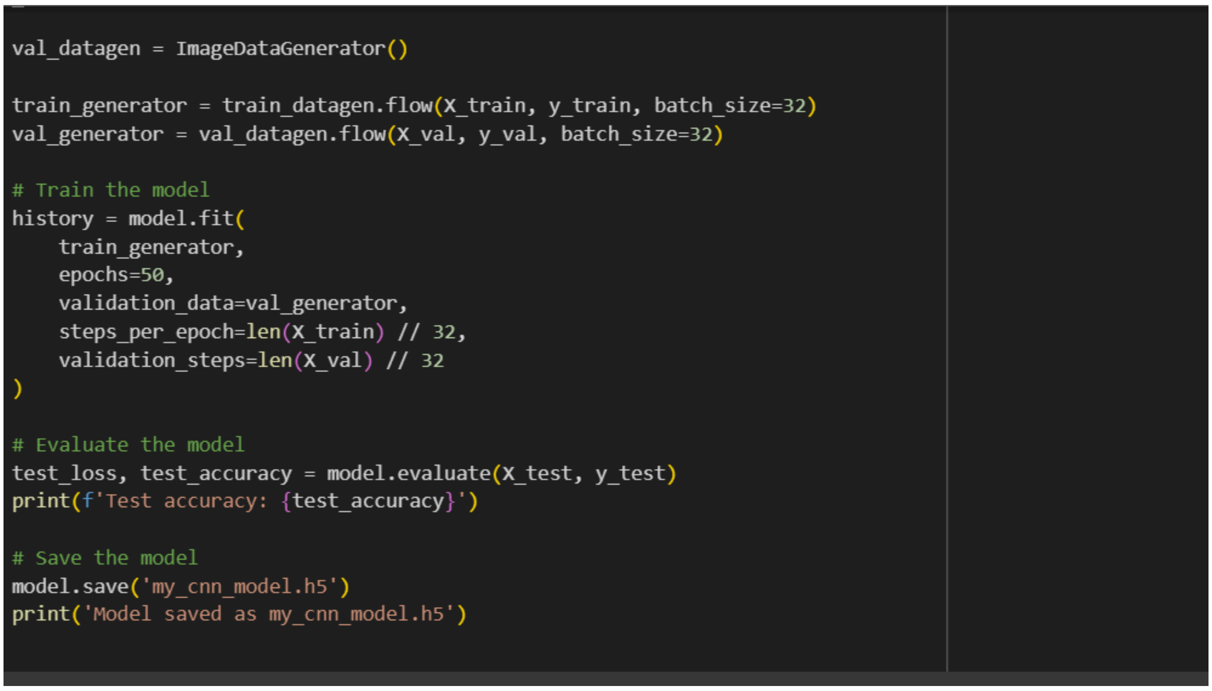
CNN:



```python
# Import necessary modules
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
import numpy as np

# Example: Load your data here
# Assuming X_train, y_train, X_val, y_val, X_test, y_test are loaded

# For demonstration purposes, let's create some dummy data
# Replace this with your actual data loading logic
num_samples = 1000
img_height, img_width = 256, 256

X_train = np.random.rand(num_samples, img_height, img_width, 3)  # Change the last dimension to 3
y_train = np.random.randint(2, size=(num_samples, 1))

X_val = np.random.rand(int(num_samples * 0.2), img_height, img_width, 3)  # Change the last dimension to 3
y_val = np.random.randint(2, size=(int(num_samples * 0.2), 1))

X_test = np.random.rand(int(num_samples * 0.2), img_height, img_width, 3)  # Change the last dimension to 3
y_test = np.random.randint(2, size=(int(num_samples * 0.2), 1))

# Normalize pixel values to be between 0 and 1
X_train = X_train.astype('float32') / 255.0
X_val = X_val.astype('float32') / 255.0
```

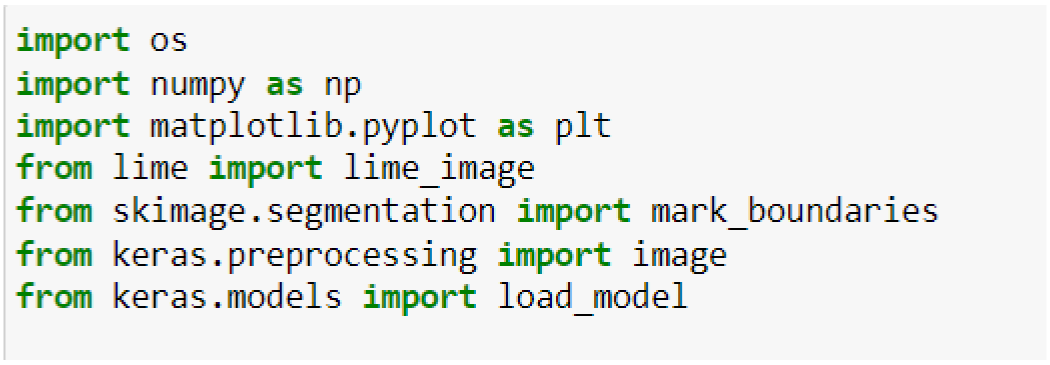


```python
# Define the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(img_height, img_width, 3)),  # Change input shape
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(64, activation='relu'),
    Dropout(0.5),  # Optional: Add dropout for regularization
    Dense(1, activation='sigmoid')
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='binary_crossentropy',
              metrics=['accuracy'])

# Model summary
model.summary()

# Create data generators for data augmentation (optional)
train_datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True
)
```

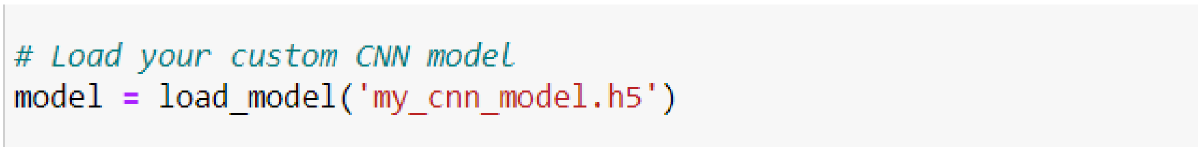


```
val_datagen = ImageDataGenerator()

train_generator = train_datagen.flow(X_train, y_train, batch_size=32)
val_generator = val_datagen.flow(X_val, y_val, batch_size=32)

# Train the model
history = model.fit(
    train_generator,
    epochs=50,
    validation_data=val_generator,
    steps_per_epoch=len(X_train) // 32,
    validation_steps=len(X_val) // 32
)

# Evaluate the model
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(f'Test accuracy: {test_accuracy}')

# Save the model
model.save('my_cnn_model.h5')
print('Model saved as my_cnn_model.h5')
```

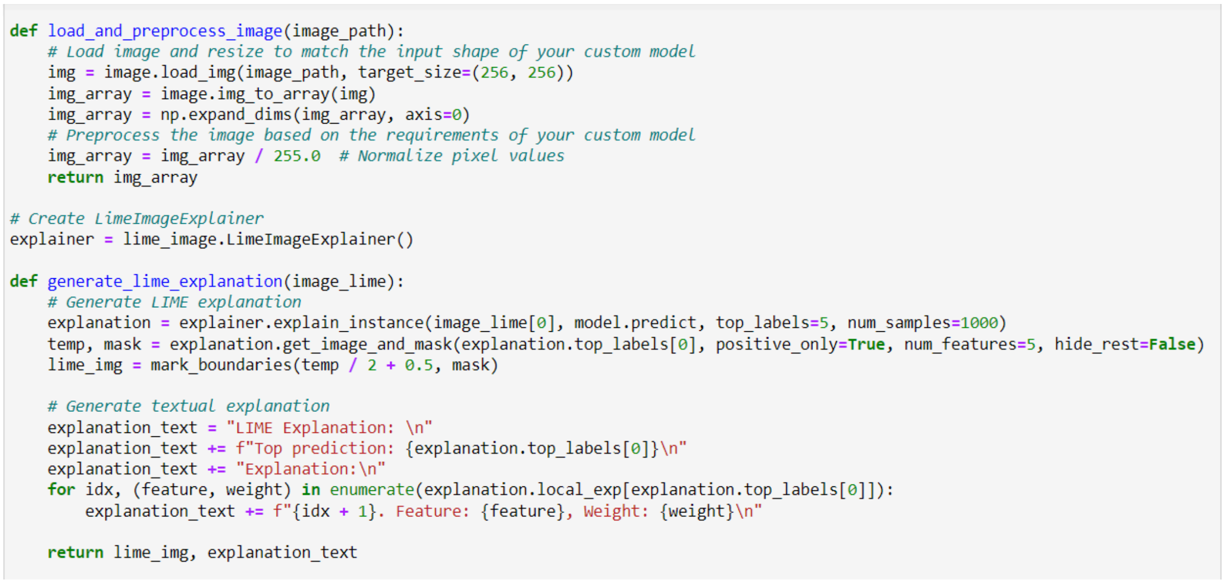
LIME:



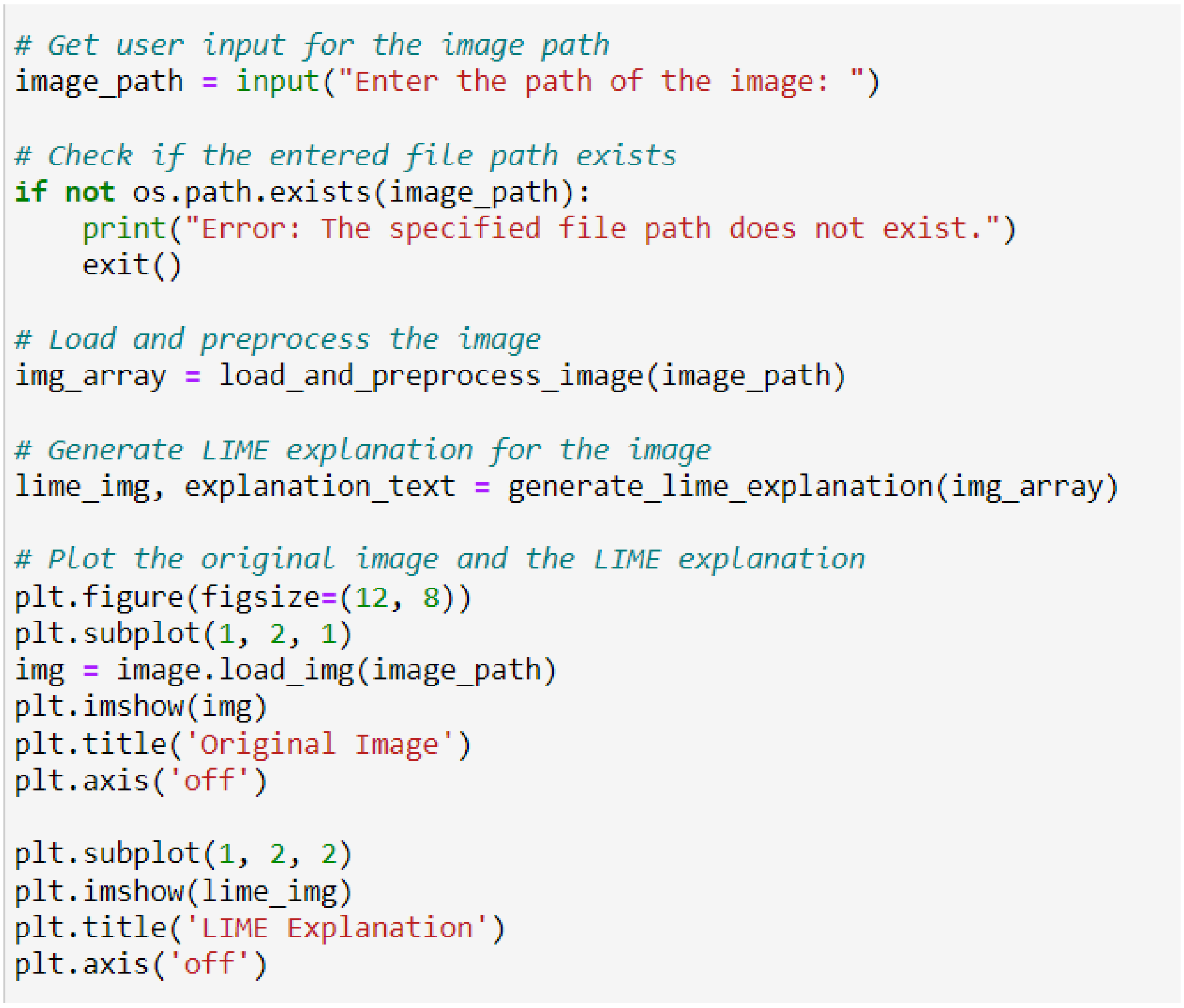
```
import os
import numpy as np
import matplotlib.pyplot as plt
from lime import lime_image
from skimage.segmentation import mark_boundaries
from keras.preprocessing import image
from keras.models import load_model
```



```
# Load your custom CNN model
model = load_model('my_cnn_model.h5')
```

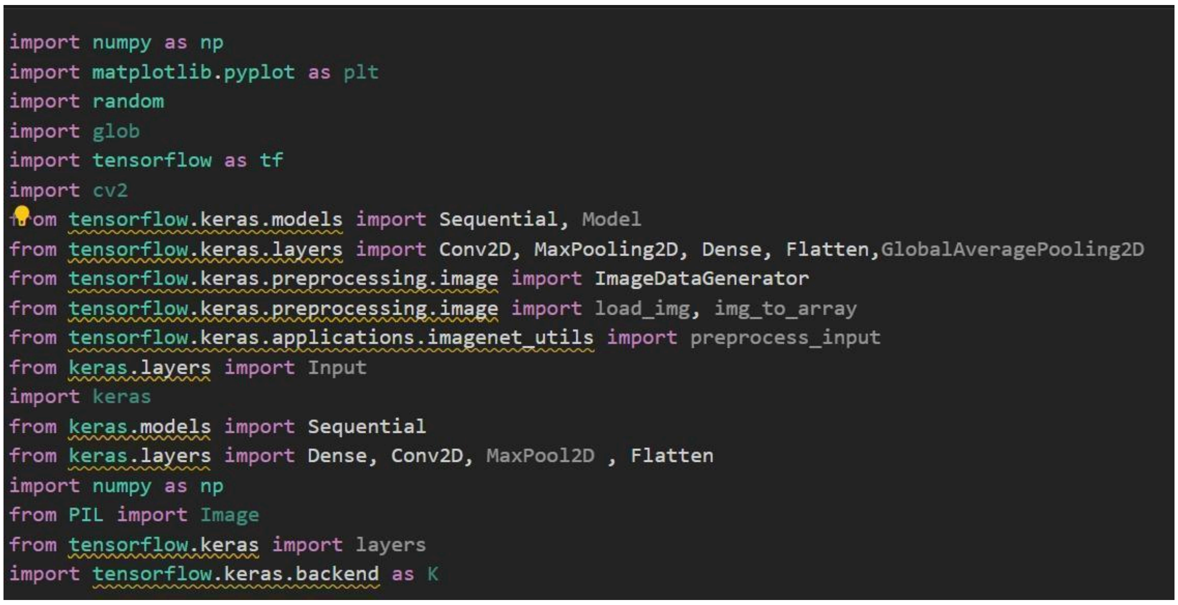


```
def load_and_preprocess_image(image_path):
    # Load image and resize to match the input shape of your custom model
    img = image.load_img(image_path, target_size=(256, 256))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0)
    # Preprocess the image based on the requirements of your custom model
    img_array = img_array / 255.0  # Normalize pixel values
    return img_array

# Create LimeImageExplainer
explainer = lime_image.LimeImageExplainer()

def generate_lime_explanation(image_lime):
    # Generate LIME explanation
    explanation = explainer.explain_instance(image_lime[0], model.predict, top_labels=5, num_samples=1000)
    temp, mask = explanation.get_image_and_mask(explanation.top_labels[0], positive_only=True, num_features=5, hide_rest=False)
    lime_img = mark_boundaries(temp / 2 + 0.5, mask)

    # Generate textual explanation
    explanation_text = "LIME Explanation: \n"
    explanation_text += f"Top prediction: {explanation.top_labels[0]}\n"
    explanation_text += "Explanation:\n"
    for idx, (feature, weight) in enumerate(explanation.local_exp[explanation.top_labels[0]]):
        explanation_text += f"{idx + 1}. Feature: {feature}, Weight: {weight}\n"

    return lime_img, explanation_text
```

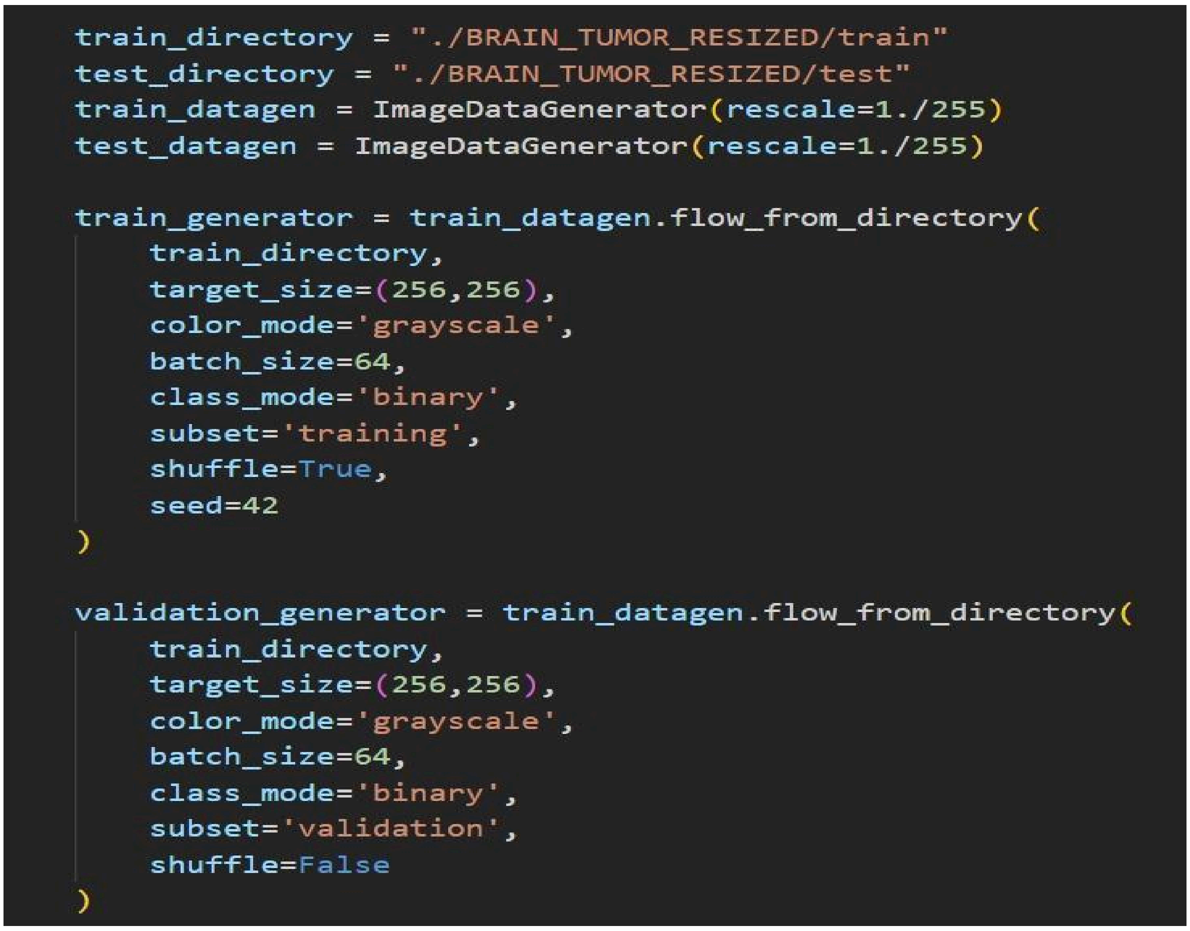


```
# Get user input for the image path
image_path = input("Enter the path of the image: ")

# Check if the entered file path exists
if not os.path.exists(image_path):
    print("Error: The specified file path does not exist.")
    exit()

# Load and preprocess the image
img_array = load_and_preprocess_image(image_path)

# Generate LIME explanation for the image
lime_img, explanation_text = generate_lime_explanation(img_array)

# Plot the original image and the LIME explanation
plt.figure(figsize=(12, 8))
plt.subplot(1, 2, 1)
img = image.load_img(image_path)
plt.imshow(img)
plt.title('Original Image')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(lime_img)
plt.title('LIME Explanation')
plt.axis('off')
```

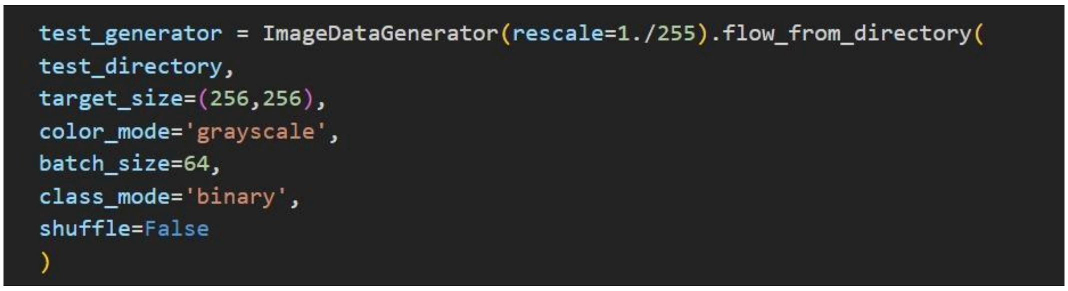
GRAD-CAM:



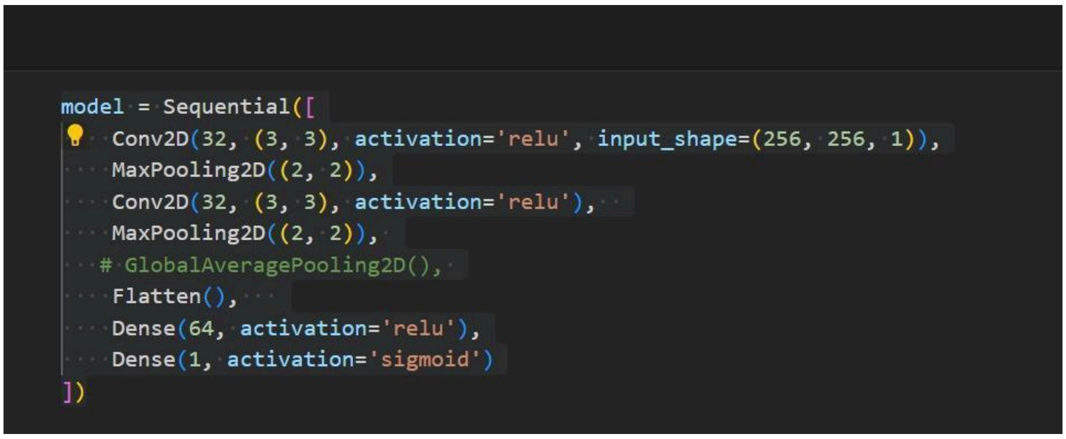
```python
import numpy as np
import matplotlib.pyplot as plt
import random
import glob
import tensorflow as tf
import cv2
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten,GlobalAveragePooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from tensorflow.keras.applications.imagenet_utils import preprocess_input
from keras.layers import Input
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPool2D , Flatten
import numpy as np
from PIL import Image
from tensorflow.keras import layers
import tensorflow.keras.backend as K
```



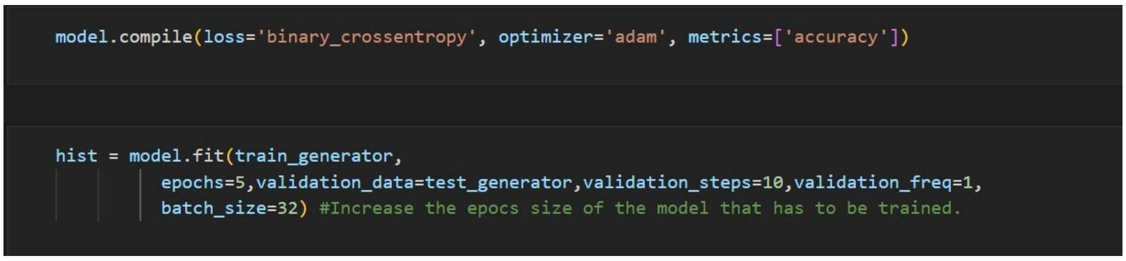
```python
train_directory = "./BRAIN_TUMOR_RESIZED/train"
test_directory = "./BRAIN_TUMOR_RESIZED/test"
train_datagen = ImageDataGenerator(rescale=1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_directory,
    target_size=(256,256),
    color_mode='grayscale',
    batch_size=64,
    class_mode='binary',
    subset='training',
    shuffle=True,
    seed=42
)

validation_generator = train_datagen.flow_from_directory(
    train_directory,
    target_size=(256,256),
    color_mode='grayscale',
    batch_size=64,
    class_mode='binary',
    subset='validation',
    shuffle=False
)
```



```
test_generator = ImageDataGenerator(rescale=1./255).flow_from_directory(
test_directory,
target_size=(256,256),
color_mode='grayscale',
batch_size=64,
class_mode='binary',
shuffle=False
)
```

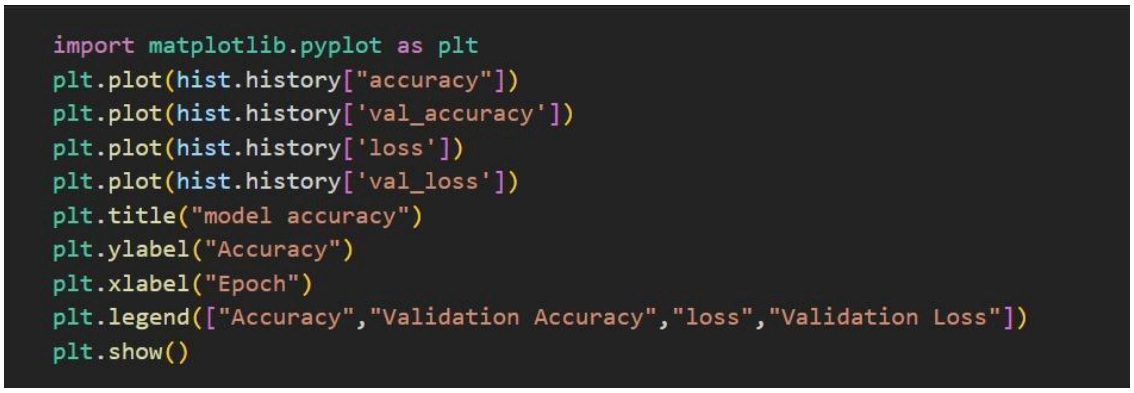


```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 1)),
    MaxPooling2D((2, 2)),
    Conv2D(32, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
   # GlobalAveragePooling2D(),
    Flatten(),
    Dense(64, activation='relu'),
    Dense(1, activation='sigmoid')
])
```

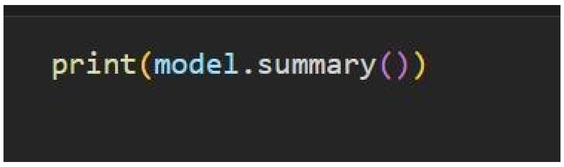


```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

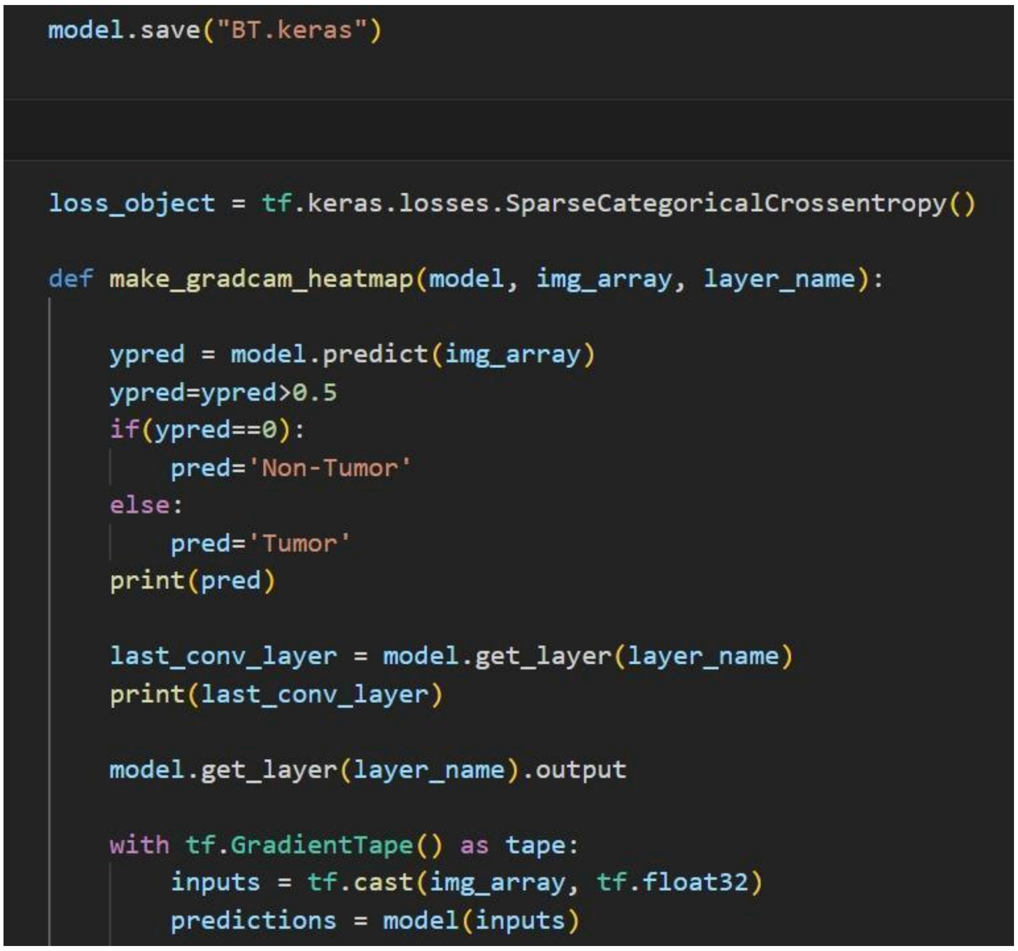
```
hist = model.fit(train_generator,
        epochs=5,validation_data=test_generator,validation_steps=10,validation_freq=1,
        batch_size=32) #Increase the epocs size of the model that has to be trained.
```



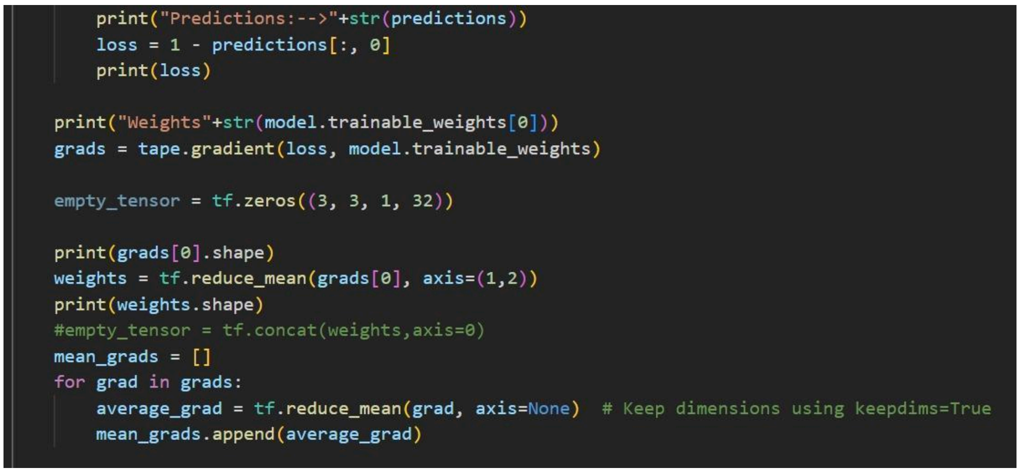
```
import matplotlib.pyplot as plt
plt.plot(hist.history["accuracy"])
plt.plot(hist.history['val_accuracy'])
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title("model accuracy")
plt.ylabel("Accuracy")
plt.xlabel("Epoch")
plt.legend(["Accuracy","Validation Accuracy","loss","Validation Loss"])
plt.show()
```



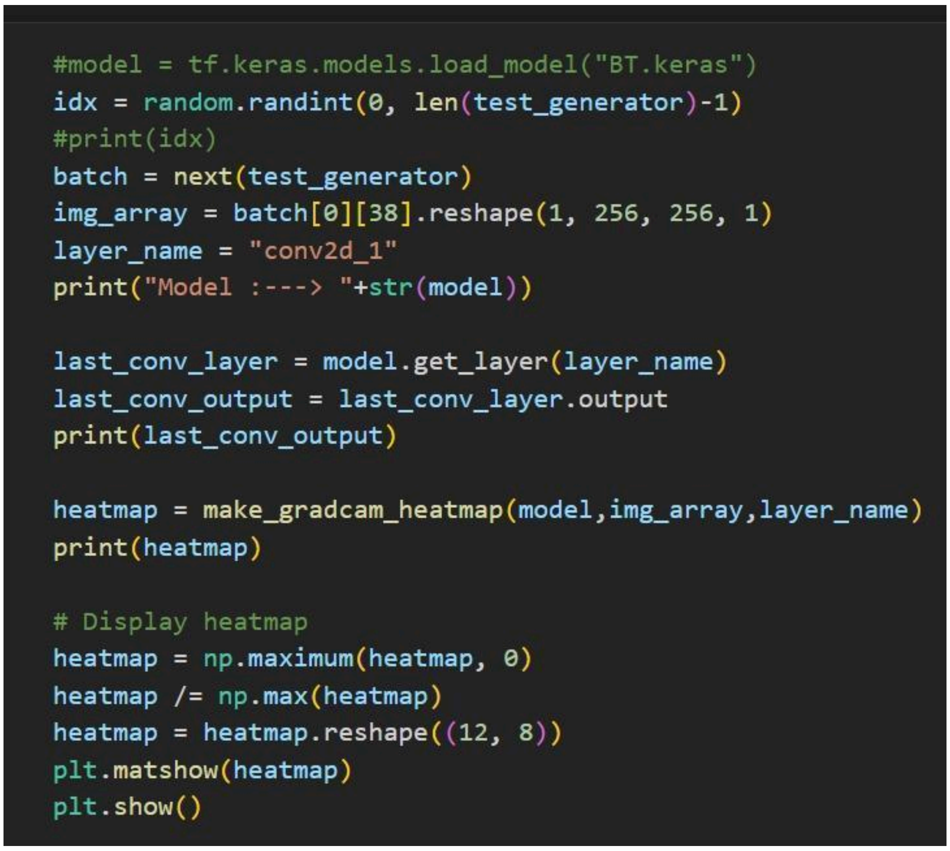
```
print(model.summary())
```



```
model.save("BT.keras")


loss_object = tf.keras.losses.SparseCategoricalCrossentropy()

def make_gradcam_heatmap(model, img_array, layer_name):

    ypred = model.predict(img_array)
    ypred=ypred>0.5
    if(ypred==0):
        pred='Non-Tumor'
    else:
        pred='Tumor'
    print(pred)

    last_conv_layer = model.get_layer(layer_name)
    print(last_conv_layer)

    model.get_layer(layer_name).output

    with tf.GradientTape() as tape:
        inputs = tf.cast(img_array, tf.float32)
        predictions = model(inputs)
```



```
        print("Predictions:-->"+str(predictions))
        loss = 1 - predictions[:, 0]
        print(loss)

    print("Weights"+str(model.trainable_weights[0]))
    grads = tape.gradient(loss, model.trainable_weights)

    empty_tensor = tf.zeros((3, 3, 1, 32))

    print(grads[0].shape)
    weights = tf.reduce_mean(grads[0], axis=(1,2))
    print(weights.shape)
    #empty_tensor = tf.concat(weights,axis=0)
    mean_grads = []
    for grad in grads:
        average_grad = tf.reduce_mean(grad, axis=None)  # Keep dimensions using keepdims=True
        mean_grads.append(average_grad)
```



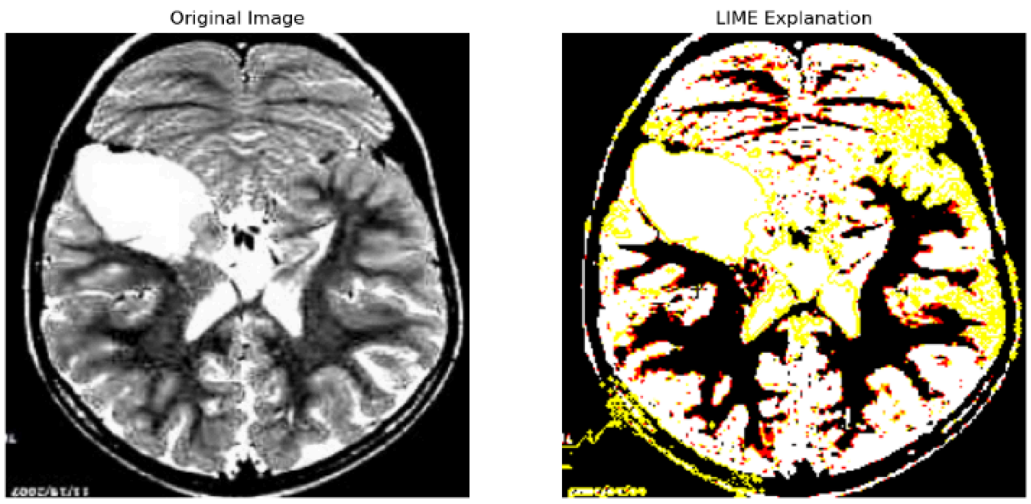
```
#model = tf.keras.models.load_model("BT.keras")
idx = random.randint(0, len(test_generator)-1)
#print(idx)
batch = next(test_generator)
img_array = batch[0][38].reshape(1, 256, 256, 1)
layer_name = "conv2d_1"
print("Model :---> "+str(model))

last_conv_layer = model.get_layer(layer_name)
last_conv_output = last_conv_layer.output
print(last_conv_output)

heatmap = make_gradcam_heatmap(model,img_array,layer_name)
print(heatmap)

# Display heatmap
heatmap = np.maximum(heatmap, 0)
heatmap /= np.max(heatmap)
heatmap = heatmap.reshape((12, 8))
plt.matshow(heatmap)
plt.show()
```

OUTPUT:

1.(Using VGG16)

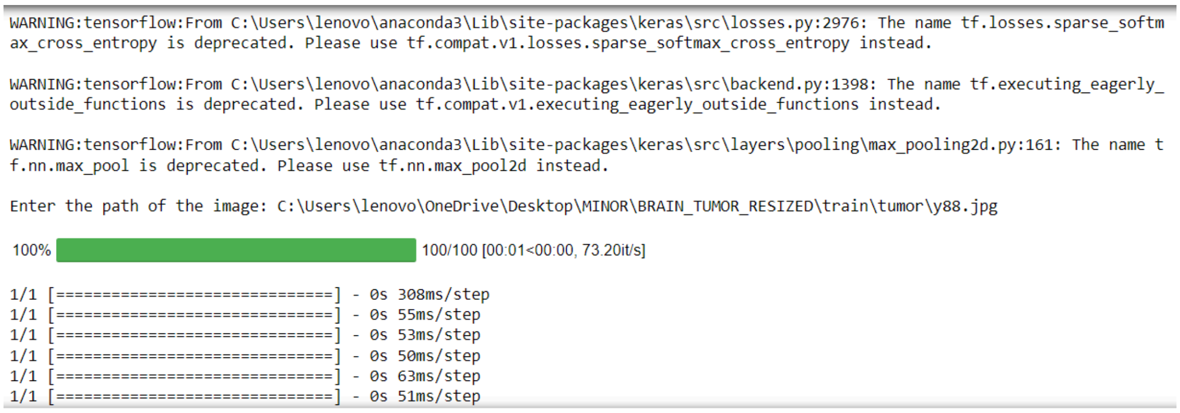
Contains Textual Explanation in form of Feature and Its Weight

Total Features:365

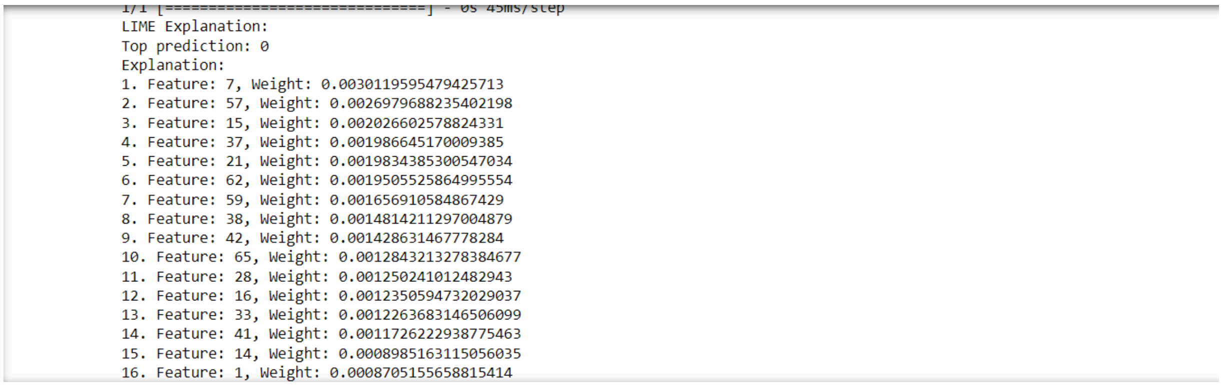


Contains Textual Explanation in form of Feature and Its Weight

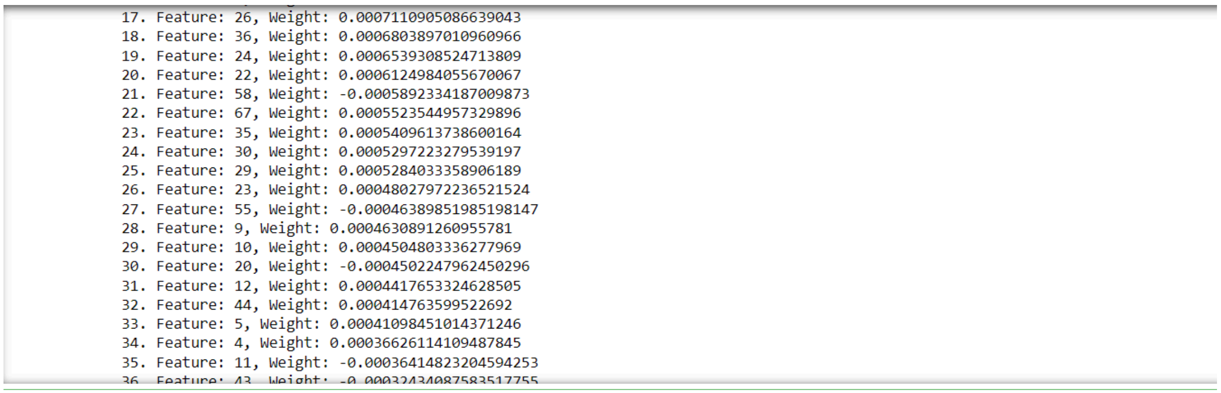
2.(Using Custom Model named my_cnn_model.h5)



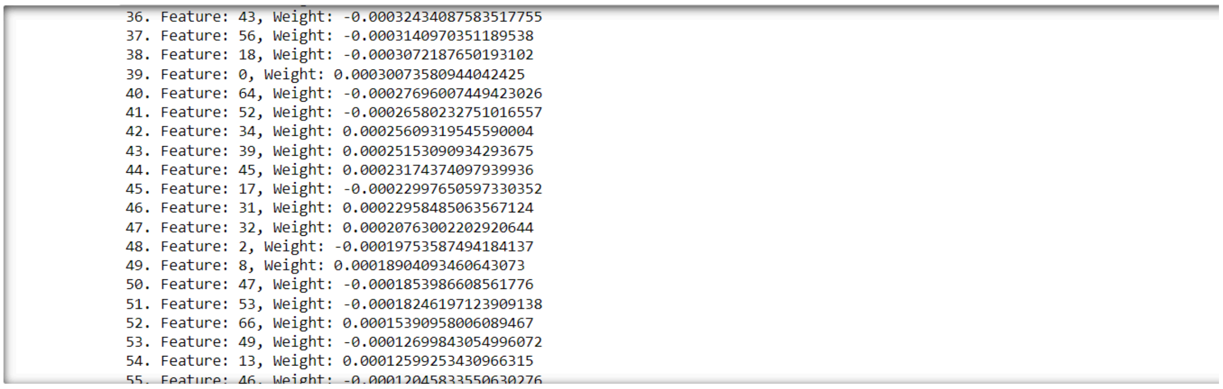
```
WARNING:tensorflow:From C:\Users\lenovo\anaconda3\Lib\site-packages\keras\src\losses.py:2976: The name tf.losses.sparse_softm
ax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

WARNING:tensorflow:From C:\Users\lenovo\anaconda3\Lib\site-packages\keras\src\backend.py:1398: The name tf.executing_eagerly_
outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\lenovo\anaconda3\Lib\site-packages\keras\src\layers\pooling\max_pooling2d.py:161: The name t
f.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Enter the path of the image: C:\Users\lenovo\OneDrive\Desktop\MINOR\BRAIN_TUMOR_RESIZED\train\tumor\y88.jpg

100% ████████████████████ 100/100 [00:01<00:00, 73.20it/s]

1/1 [==============================] - 0s 308ms/step
1/1 [==============================] - 0s 55ms/step
1/1 [==============================] - 0s 53ms/step
1/1 [==============================] - 0s 50ms/step
1/1 [==============================] - 0s 63ms/step
1/1 [==============================] - 0s 51ms/step
```



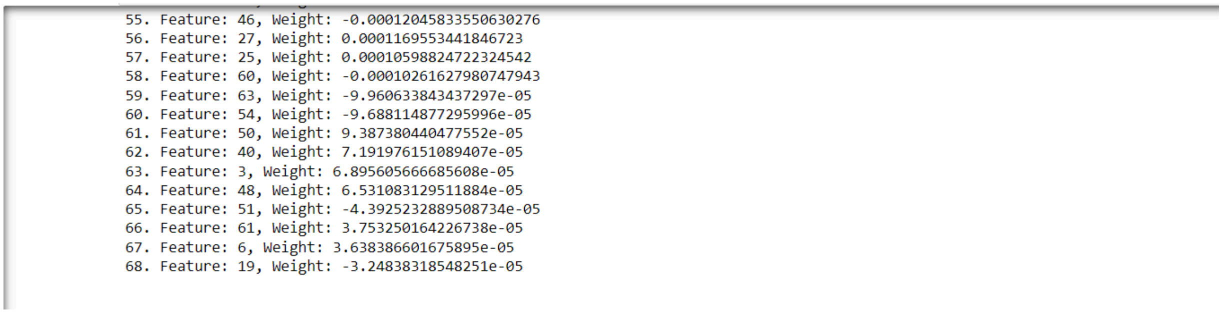
```
1/1 [==============================] - 0s 45ms/step
LIME Explanation:
Top prediction: 0
Explanation:
1. Feature: 7, Weight: 0.0030119595479425713
2. Feature: 57, Weight: 0.0026979688235402198
3. Feature: 15, Weight: 0.0020266025788 24331
4. Feature: 37, Weight: 0.001986645170009385
5. Feature: 21, Weight: 0.0019834385300547034
6. Feature: 62, Weight: 0.0019505525864995554
7. Feature: 59, Weight: 0.001656910584867429
8. Feature: 38, Weight: 0.0014814211297004879
9. Feature: 42, Weight: 0.001428631467778284
10. Feature: 65, Weight: 0.0012843213278384677
11. Feature: 28, Weight: 0.001250241012482943
12. Feature: 16, Weight: 0.0012350594732029037
13. Feature: 33, Weight: 0.0012263683146506099
14. Feature: 41, Weight: 0.0011726222938775463
15. Feature: 14, Weight: 0.0008985163115056035
16. Feature: 1, Weight: 0.0008705155658815414
```



```
17. Feature: 26, Weight: 0.0007110905086639043
18. Feature: 36, Weight: 0.0006803897010960966
19. Feature: 24, Weight: 0.0006539308524713809
20. Feature: 22, Weight: 0.0006124984055670067
21. Feature: 58, Weight: -0.0005892334187009873
22. Feature: 67, Weight: 0.0005523544957329896
23. Feature: 35, Weight: 0.0005409613738600164
24. Feature: 30, Weight: 0.0005297223279539197
25. Feature: 29, Weight: 0.0005284033358906189
26. Feature: 23, Weight: 0.00048027972236521524
27. Feature: 55, Weight: -0.00046389851985198147
28. Feature: 9, Weight: 0.0004630891260955781
29. Feature: 10, Weight: 0.0004504803336277969
30. Feature: 20, Weight: -0.0004502247962450296
31. Feature: 12, Weight: 0.0004417653324628505
32. Feature: 44, Weight: 0.000414763599522692
33. Feature: 5, Weight: 0.00041098451014371246
34. Feature: 4, Weight: 0.0003662611410948784 5
35. Feature: 11, Weight: -0.00036414823204594253
36. Feature: 43, Weight: -0.00032434087583517755
```

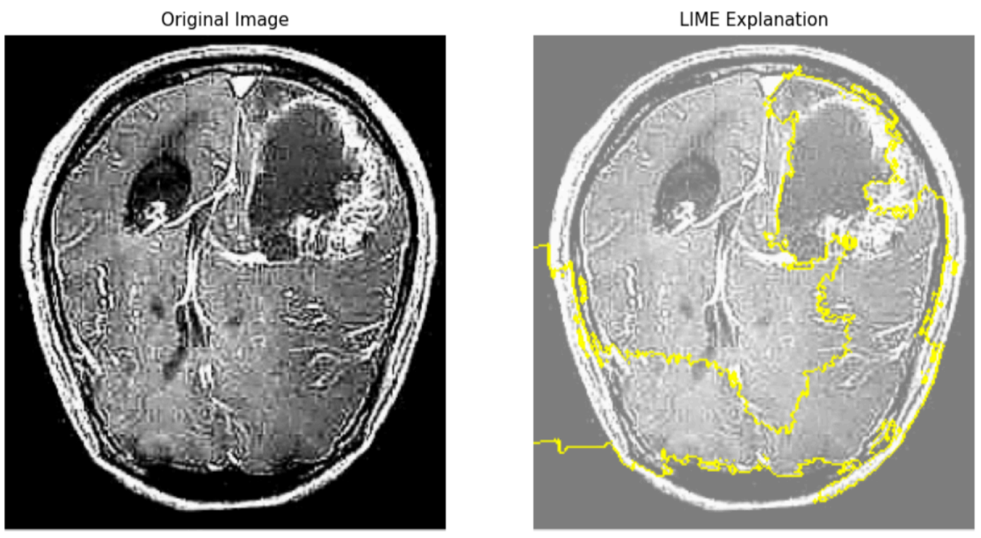


```
36. Feature: 43, Weight: -0.00032434087583517755
37. Feature: 56, Weight: -0.00031409703511895 38
38. Feature: 18, Weight: -0.0003072187650193102
39. Feature: 0, Weight: 0.0003007358094404 2425
40. Feature: 64, Weight: -0.00027696007449423026
41. Feature: 52, Weight: -0.00026580232751016557
42. Feature: 34, Weight: 0.0002560931954559 0004
43. Feature: 39, Weight: 0.0002515309093429 3675
44. Feature: 45, Weight: 0.0002317437409793 9936
45. Feature: 17, Weight: -0.00022997650597330352
46. Feature: 31, Weight: 0.0002295848506356 7124
47. Feature: 32, Weight: 0.0002076300220292 0644
48. Feature: 2, Weight: -0.0001975358749418 4137
49. Feature: 8, Weight: 0.0001890409346064 3073
50. Feature: 47, Weight: -0.0001853986608561776
51. Feature: 53, Weight: -0.00018246197123909138
52. Feature: 66, Weight: 0.0001539095800608 9467
53. Feature: 49, Weight: -0.00012699843054996072
54. Feature: 13, Weight: 0.0001259925343096 6315
55. Feature: 46, Weight: -0.00012045833550630276
```

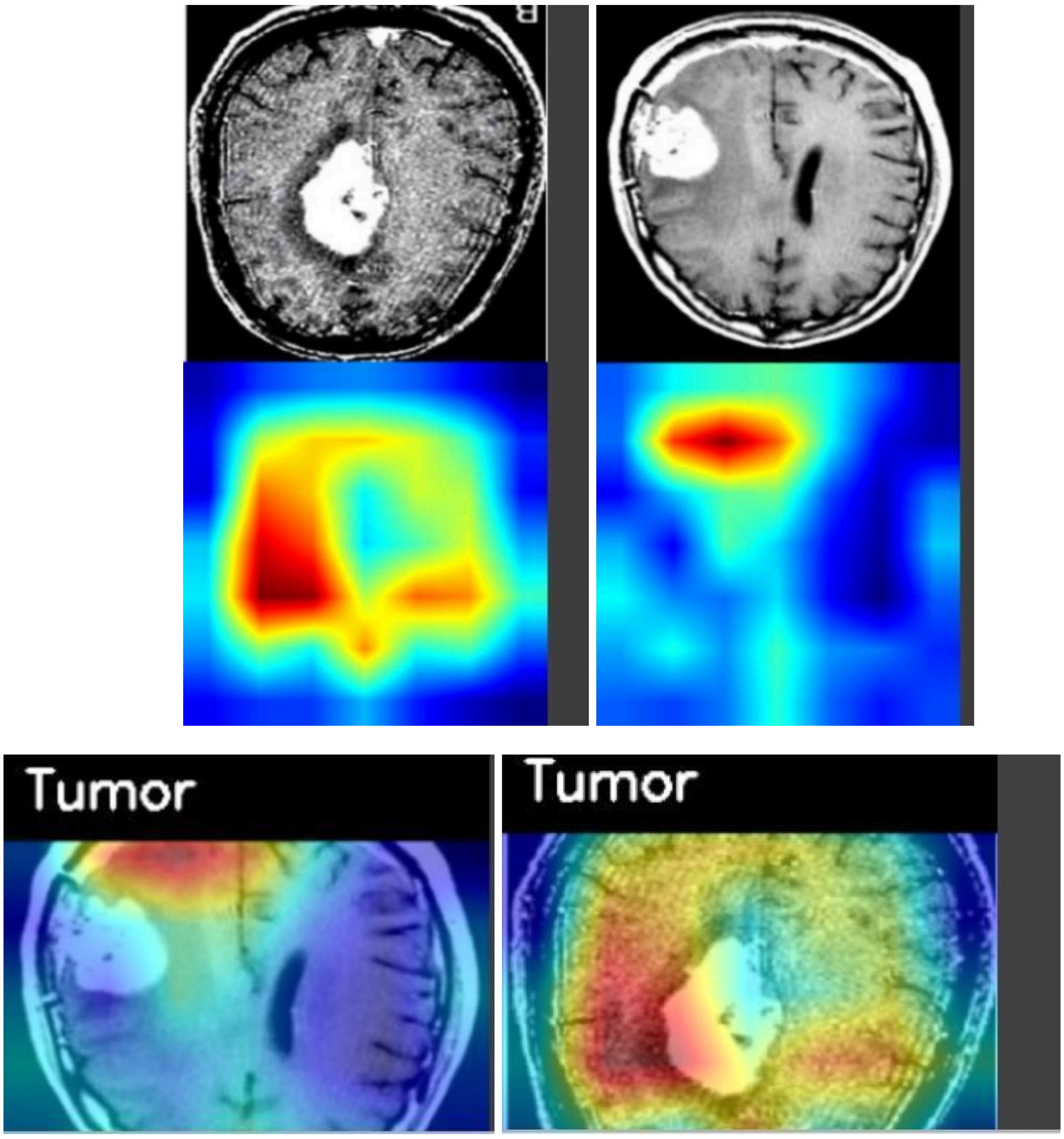


```
55. Feature: 46, Weight: -0.00012045833550630276
56. Feature: 27, Weight: 0.0001169553441846723
57. Feature: 25, Weight: 0.00010598824722324542
58. Feature: 60, Weight: -0.00010261627980747943
59. Feature: 63, Weight: -9.960633843437297e-05
60. Feature: 54, Weight: -9.688114877295996e-05
61. Feature: 50, Weight: 9.387380440477552e-05
62. Feature: 40, Weight: 7.191976151089407e-05
63. Feature: 3, Weight: 6.895605666685608e-05
64. Feature: 48, Weight: 6.531083129511884e-05
65. Feature: 51, Weight: -4.3925232889508734e-05
66. Feature: 61, Weight: 3.753250164226738e-05
67. Feature: 6, Weight: 3.638386601675895e-05
68. Feature: 19, Weight: -3.24838318548251e-05
```

IMAGE OUTPUT:

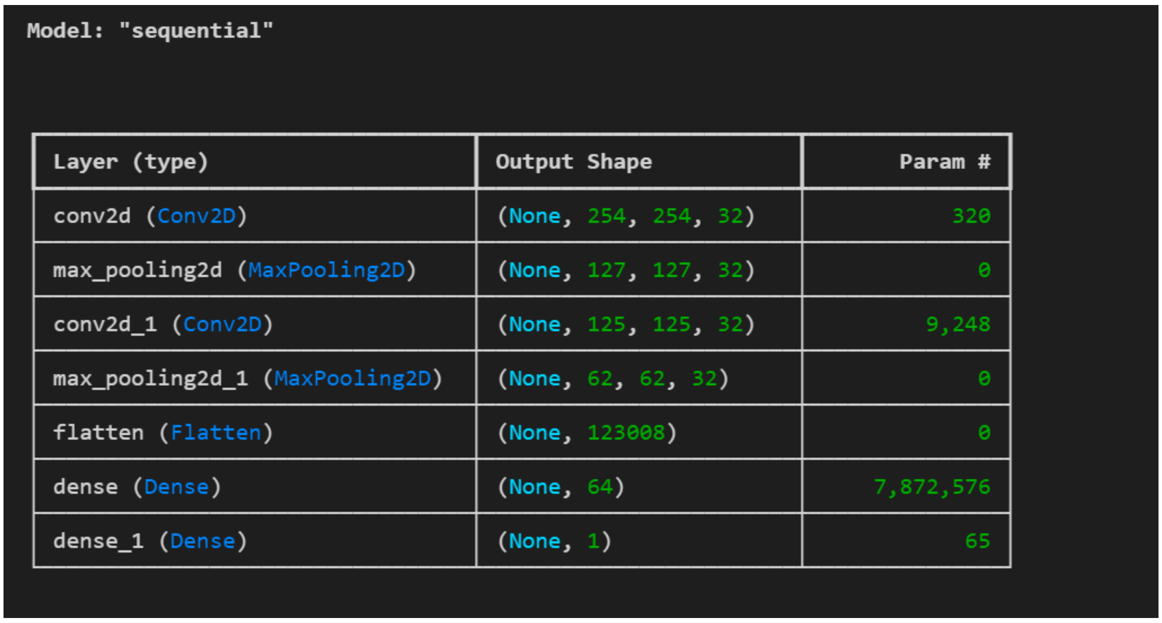


- **GRAD-CAM:**



Chapter 04: RESULTS

4.1 Discussion on the Results Achieved: -



```
Model: "sequential"

┌─────────────────────────────────┬───────────────────────┬────────────┐
│ Layer (type)                    │ Output Shape          │    Param # │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ conv2d (Conv2D)                 │ (None, 254, 254, 32)  │        320 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 127, 127, 32)  │          0 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ conv2d_1 (Conv2D)               │ (None, 125, 125, 32)  │      9,248 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 62, 62, 32)    │          0 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ flatten (Flatten)               │ (None, 123008)        │          0 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ dense (Dense)                   │ (None, 64)            │  7,872,576 │
├─────────────────────────────────┼───────────────────────┼────────────┤
│ dense_1 (Dense)                 │ (None, 1)             │         65 │
└─────────────────────────────────┴───────────────────────┴────────────┘
```

Building Block by Block: A Breakdown of the Neural Network Architecture

This network follows a typical structure for convolutional neural networks (CNNs) designed for image classification. Let's dissect each layer and understand its role:

1. **Convolutional Powerhouse (Conv2D):**
   The first layer acts as the foundation, equipped with 32 filters, each of size 3x3. These filters are like tiny feature detectors, scanning the image for specific patterns.
   The 'relu' activation function adds a layer of non-linearity, allowing the network to learn more complex relationships within the extracted features.
   The input shape (256, 256, 1) specifies that the network is designed for grayscale images (1 channel) with a width and height of 256 pixels.
2. **Pooling for Efficiency (MaxPooling2D):**
   This layer performs a down sampling technique called max pooling. It works by sliding a 2x2 window across the feature maps from the previous layer and keeping only the maximum value within each window.
   This reduces the image dimensions while preserving key features, making the network more efficient and less prone to overfitting.
3. **Extracting More Features (Conv2D):**
   Similar to the first convolutional layer, this layer employs a new set of 32 filters, potentially capturing different features from the image.

The 'relu' activation is applied again to introduce non-linearity.

4. **Another Round of Pooling (MaxPooling2D):**

   Just like before, this layer performs max pooling with a 2x2 window, further reducing the dimensionality of the data.

5. **Flattening the Landscape (Flatten):**

   Up to this point, the network has processed the image spatially. This layer transforms the pooled feature maps from a multi-dimensional array into a single, long vector.

   This step prepares the data for the fully-connected layers, which operate on one-dimensional inputs.

6. **Fully-Connected Insights (Dense):**

   The first fully-connected layer introduces 64 neurons. Unlike convolutional layers that work locally on features, fully-connected layers connect every neuron from the previous layer to every neuron in this layer.

   This allows the network to learn more intricate relationships between the extracted features. The 'relu' activation function is used for non-linearity.

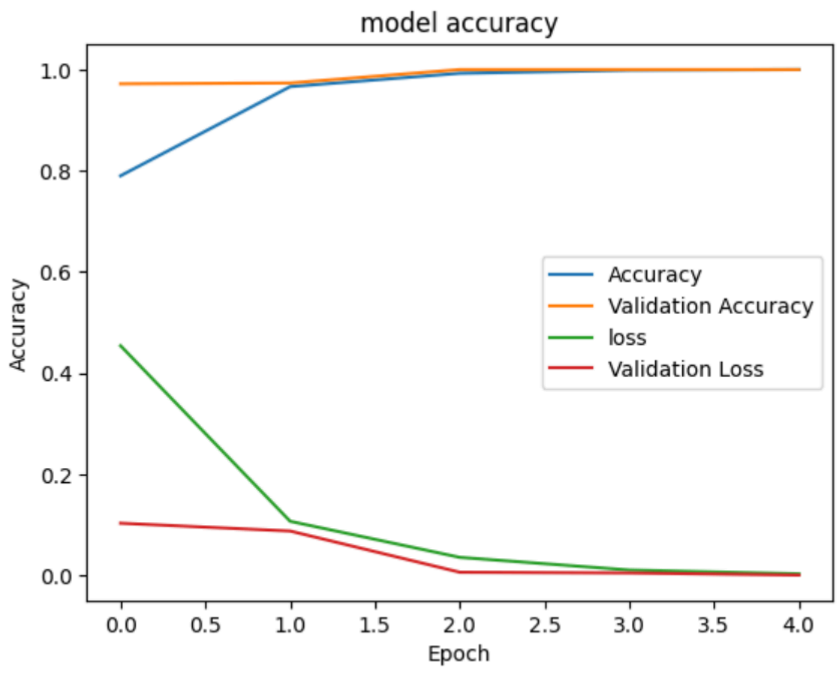
7. **Binary Decision Maker (Dense):**

   The final layer has just one neuron, indicating the network is likely trained for a binary classification task (classifying between two categories).

   The 'sigmoid' activation function ensures the output is between 0 and 1, ideal for binary classification problems.

In essence, the convolutional layers act as feature extractors, while the pooling layers reduce dimensionality. The fully-connected layers then take these features and learn complex combinations to make the final classification decision.

PLOT OF MODEL ACCURACY:



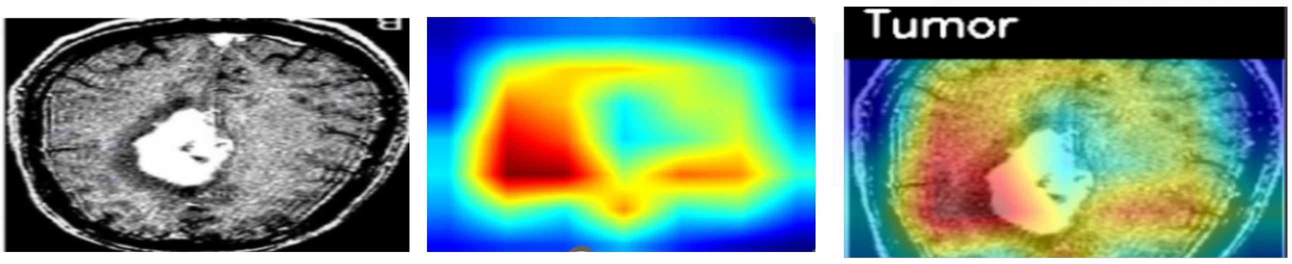
This is the plot of the training and validation accuracy/loss of a model to understand how well it learns and generalizes on unseen data.

EXAI OUTCOMES: -

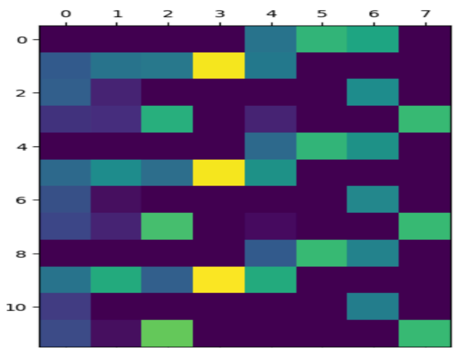
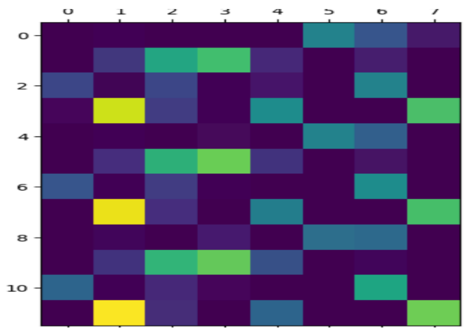
- **Grad-CAM[Gradient Class Activation Map]**

Grad-CAM is a technique that sheds light on how deep learning models arrive at their decisions for images. It functions as a foundation for visualizing the most critical areas within an image that influence the model's prediction. By highlighting these crucial regions, Grad-CAM enhances the interpretability of the model and fosters trust in its results.

This technique is particularly effective when applied to the VGG16 architecture, a popular deep learning model.

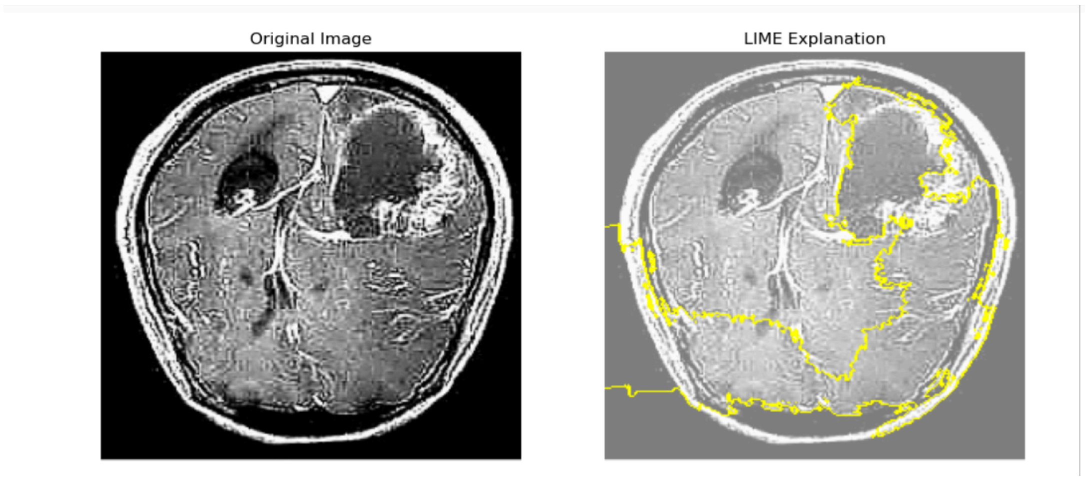


The following below is the heatmap generated for the sequential CNN developed these heatmap are the result of multiple images. This visualization technique helps understand which parts of an image the CNN model focuses on when making predictions.

- **LIME[Local Interpretability Model Agnostic Explanations]**

Local Explanations: LIME provides explanations that are **specific to the data point** being analysed. These explanations are considered "locally faithful" because they accurately reflect the model's behaviour within the immediate vicinity of that particular data point.



4.2 Application of the Minor Project: -

Explainable AI (XAI) is a burgeoning field with myriad applications across various industries. Here, I'll outline original use cases where XAI plays a pivotal role:

1. **Medical Education and Training**: Your project can be used as an educational tool for medical students, residents, and other healthcare professionals to learn about brain tumor anatomy, imaging interpretation, and segmentation techniques. By providing access to annotated medical imaging datasets and visualization tools, your project can facilitate hands-on learning experiences and skill development in medical imaging analysis.

2. **Clinical Decision Support**: Your project can serve as a valuable tool for radiologists and neurosurgeons in the diagnosis and treatment planning of brain tumors. By accurately segmenting tumor regions from medical imaging scans, clinicians can better visualize and quantify tumor size, location, and growth over time, leading to more informed treatment decisions.

3. **Treatment Planning and Monitoring**: Segmentation results from your project can be used to create patient-specific treatment plans, such as radiation therapy or surgical interventions. By precisely delineating tumor boundaries, clinicians can target therapy to the tumor while minimizing damage to healthy surrounding tissues, thereby improving treatment efficacy and patient outcomes.

4. **Research and Clinical Trials**: Your project's segmentation capabilities can support research efforts aimed at understanding the underlying biology of brain tumors and developing novel therapies. Segmentation results can be used to analyze tumor characteristics, such as shape, texture, and heterogeneity, to identify prognostic markers and therapeutic targets. Additionally, your project can facilitate the enrollment and monitoring of patients in clinical trials investigating new treatment modalities for brain tumors.

5. **Integration with Clinical Decision Support Systems**: Your project's segmentation capabilities can be integrated into existing clinical decision support systems to enhance their functionality and usability. By providing explainable AI-driven segmentation results, your project can improve the transparency and trustworthiness of decision support systems, enabling clinicians to make more confident and evidence-based decisions in patient care.

4.3 Limitation of the Minor Project

1. **Limited Training Data**: Segmentation models require a large and diverse dataset for training, especially for complex tasks like brain tumor segmentation. If the dataset used for training the model is limited in size or lacks diversity, it may result in suboptimal performance and generalization to unseen data.

2. **Model Complexity**: The choice of segmentation model architecture can impact both performance and explainability. More complex models may achieve higher segmentation accuracy but can be challenging to interpret using XAI techniques. Simplifying the model architecture to improve interpretability may sacrifice segmentation performance.

3. **XAI Interpretability**: While XAI techniques aim to provide insights into the model's decision-making process, the interpretability of the explanations may be limited. For instance, if the model's predictions are based on intricate interactions between features or layers, XAI explanations may struggle to capture the full complexity of these relationships.

4. **Clinical Validation**: The ultimate utility of the segmentation model lies in its clinical applicability and reliability. Clinical validation studies are necessary to assess the model's performance in real-world scenarios and its impact on clinical decision-making. Without rigorous validation, the model's limitations and potential biases may go unnoticed.

4.4 Future Work

1. **Improving Model Performance**: Experiment with different model architectures, such as convolutional neural networks (CNNs) or transformer-based models, to enhance segmentation accuracy and generalization to unseen data. Incorporate techniques like data augmentation and transfer learning to leverage larger datasets and improve model robustness.

2. **Enhancing XAI Techniques**: Develop or refine XAI techniques tailored specifically for medical image analysis tasks, such as attention maps, saliency maps, or feature importance visualization methods. Investigate how these techniques can provide more interpretable explanations of the model's segmentation decisions and aid clinicians in understanding and trusting the results.

3. **Clinical Validation and Integration**: Conduct extensive clinical validation studies to assess the performance and utility of the segmentation model in real-world clinical settings. Collaborate with healthcare professionals to integrate the model into existing clinical workflows and evaluate its impact on diagnostic accuracy, treatment planning, and patient outcomes.

4. **Multi-modal Fusion**: Explore the integration of multi-modal imaging data, such as MRI, CT, PET, and histopathology images, to improve the accuracy and comprehensiveness of tumor segmentation. Investigate fusion techniques, such as multi-modal feature fusion or deep multimodal learning, to leverage complementary information from different imaging modalities.

5. **Interactive Visualization Tools**: Develop interactive visualization tools that allow clinicians to explore and interact with the segmentation results in real-time. Incorporate XAI explanations into the visualization interface to facilitate user understanding and interpretation of the segmentation outcomes.

References

All references must be in any standard style format, like IEEE/APA/etc.

[1] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9862413/
[2] https://ieeexplore.ieee.org/document/10037324
[3] https://tvst.arvojournals.org/article.aspx?articleid=2762344
[4] https://link.springer.com/article/10.1007/s42044-023-00137-w
[5]CAM_Visual_Explanations_ICCV_2017_paper.html
[6] https://ieeexplore.ieee.org/abstract/document/9183167/
[7]https://proceedings.mlr.press/v139/garreau21a/garreau21a.pdf
[8] Harshini Badisa, Madhavi Polireddy, Aslam Mohammed , “ CNN based Brain Tumor

Detection", International Journal of Engineering and Advance Technology (IJEAT), VOL.8 NO.4,
April2019.
[9] Manjunath S, Sanjay Pande M B, Raveesh B N, Madhusudhan G K, "Brain Tumor Detection
and Classification using Convolution Neural Network", International Journal of Engineering and
Advance Technology (IJEAT), VOL.8 NO.1C, May2019.
[10] J. Seetha1 and S. Selvakumar Raja2

rptttt

ORIGINALITY REPORT

11%	8%	4%	5%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to Jaypee University of Information Technology Student Paper	1%
2	www.researchgate.net Internet Source	1%
3	www.cidrap.umn.edu Internet Source	1%
4	www.mdpi.com Internet Source	1%
5	www.ncbi.nlm.nih.gov Internet Source	1%
6	Submitted to University of Westminster Student Paper	1%
7	fastercapital.com Internet Source	1%
8	Submitted to Liverpool John Moores University Student Paper	1%
9	Submitted to Midlands State University	

	Student Paper	<1 %
10	dadun.unav.edu Internet Source	<1 %
11	www.oreilly.com Internet Source	<1 %
12	Asmita Dixit, Manish Kumar Thakur. "Advancements and emerging trends in brain tumor classification using MRI: a systematic review", Network Modeling Analysis in Health Informatics and Bioinformatics, 2023 Publication	<1 %
13	Submitted to CSU, San Jose State University Student Paper	<1 %
14	meatybytes.io Internet Source	<1 %
15	Submitted to Sharda University Student Paper	<1 %
16	dokumen.pub Internet Source	<1 %
17	Ahmad Chaddad, Jihao Peng, Jian Xu, Ahmed Bouridane. "Survey of Explainable AI Techniques in Healthcare", Sensors, 2023 Publication	<1 %
18	repozitorij.etfos.hr Internet Source	

		<1%
19	Submitted to Aston University Student Paper	<1%
20	encyclopedia.pub Internet Source	<1%
21	www.researchid.co Internet Source	<1%
22	amu.ac.in Internet Source	<1%
23	link.springer.com Internet Source	<1%
24	0-www-mdpi-com.brum.beds.ac.uk Internet Source	<1%
25	core.ac.uk Internet Source	<1%
26	pch.health.wa.gov.au Internet Source	<1%
27	www.powertechjournal.com Internet Source	<1%
28	www.carecloud.com Internet Source	<1%
29	Kamini Lamba, Shalli Rani. "A novel approach of brain-computer interfacing (BCI) and Grad-	<1%

CERTIFICATE

I hereby certify that the work which is being presented in the project report titled "Development of XAI for Transparent Decision Making" in partial fulfilment of the requirements for the award of the degree of B.Tech in Computer Science and Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out during the period from January 2024 to May 2024 under the supervision of Dr.Diksha Hooda, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

The matter presented in this project report has not been submitted for the award of any other degree of this or any other university.

Siddharth Thakur (211157) [illegible]
Aditya Singh Verma (211196) [illegible]
Achintya Misra (211166) AchintyaMisra

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

[illegible]
17/5/24

Dr.Diksha Hooda
Assistant Professor
Computer Science & Engineering and Information Technology
Jaypee University of Information Technology, Waknaghat,

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: 17 May, 2024

Type of Document (Tick): [✓ **B.Tech.**]**/B.Sc./BBA/Other**

Name: Achintya Misra, Siddharth Thakur and Aditya Singh Verma **Department:** CSE **Enrolment No** 211166, 211157, 211196

Contact No. 6006545857, 9882055848, 8800352534 **E-mail.** 211166@juitsolan.in, 211157@juitsolan.in, 211196@juitsolan.in

Name of the Supervisor: Dr. Diksha Hooda

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): DEPLOYMENT OF XAI FOR TRANSPARENT DECISION MAKING

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

- Total No. of Pages = 42
- Total No. of Preliminary pages = 4
- Total No. of pages accommodate bibliography/references = 1

[illegible]

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at 11 (%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

[illegible] 18/5/24

(Signature of Guide/Supervisor)