BRAIN TUMOR DETECTION

SAVIN M S (Reg. No. **65422131007**)

INTERIM REPORT

SUBMITTED IN PARTIAL FULLFILMENT OF

REQUIREMENT FOR THE AWARD OF MSc COMPUTER

SCIENCE DEGREE OF UNIVERSITY OF KERALA

2023-24

SREE NARAYANA COLLEGE

CHERTHALA

(Affiliated to University of Kerala)

MSc. COMPUTER SCIENCE

2022 – 2024

CERTIFICATE

This is to certify that the project work entitled "**BRAIN TUMOR DETECTION**" is a bonafide report of the work done by **SAVIN M S** in partial fulfilment of the requirements for the award of the Master's degree in M.Sc. Computer Science of this institution during the year 2022 - 2024.

Staff in charge **Head of the Department** **Principal**

Presented for the viva-voice examination conducted by the University of Kerala held on …./……/2023-2024 at Sree Narayana College,Cherthala & verified by:

Date: /05/2024 **Internal Examiner** **External Examiner**

"Dedicated to Almighty & to parents, source of inspiration and courage, in all our endeavors……"

DECLARATION

I hereby declare that the project entitled "**BRAIN TUMOR DETECTION**" is an original work carried out by me under the supervision of Dr. Bindhu N (Assosiate Professor), Department of Computer Science, Sree Narayana College, Cherthala.

I also declare that this report has not been submitted to any other University or institute for the award of any fellowship or Degree or Diploma.

Place: Cherthala

Date: …./…./2024

SAVIN M S

ACKNOWLEDGEMENT

First of all, I thank almighty god for allowing me to complete our project work successfully. I would like to take a minute to acknowledgement those who helped me for completion of the project. On the course of this project, I received great aid and support from my friends, family faculties etc…, and I express our gratitude towards them.

I am esteemed to thank Dr. Bindhu T P, The Principal, Sree Narayana College, Cherthala for granting me permission to do the project.

I convey my sincere thanks to Dr. Bindhu N (Head of Department and Project Supervising Guide) for her valuable instruction and guidance to the project and her apt encouragement towards the successful completion of my project. I also express our sincere thanks to other teachers who has been assisting us throughout the project.

Once again, I remember with great pleasure and gratitude towards those who extends their hand for me.

SAVIN M S

ABSTRACT

Automated defect detection in medical imaging has become the emergent field in several medical diagnostic applications. Automated detection of tumor in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment. The conventional method for defect detection in magnetic resonance brain images is human inspection. This method will make Medical Negligence or Human Error. Hence, trusted and automatic classification schemes are essential to prevent the death rate of human. So, automated tumor detection methods are developed as it would save radiologist time and obtain a tested accuracy. The MRI brain tumor detection is complicated task due to complexity and variance of tumors. In this project, we propose the machine learning algorithms to overcome the drawbacks of traditional classifiers where tumor is detected in brain MRI using machine learning algorithms. Machine learning and image classifier can be used to efficiently detect cancer cells in brain through MRI.

.

CONTENTS

5. CONCLUSION

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

1.1 INTRODUCTION

Brain tumor is one of the most rigorous diseases in the medical science. An effective and efficient analysis is always a key concern for the radiologist in the premature phase of tumor growth. Histological grading, based on a stereotactic biopsy test, is the gold standard and the convention for detecting the grade of a brain tumor. The biopsy procedure requires the neurosurgeon to drill a small hole into the skull from which the tissue is collected. There are many risk factors involving the biopsy test, including bleeding from the tumor and brain causing infection, seizures, severe migraine, stroke, coma and even death. But the main concern with the stereotactic biopsy is that it is not 100% accurate which may result in a serious diagnostic error followed by a wrong clinical management of the disease.

Tumor biopsy being challenging for brain tumor patients, non-invasive imaging techniques like Magnetic Resonance Imaging (MRI) have been extensively employed in diagnosing brain tumors. Therefore, development of systems for the detection and prediction of the grade of tumors based on MRI data has become necessary. But at first sight of the imaging modality like in Magnetic Resonance Imaging (MRI), the proper visualisation of the tumor cells and its differentiation with its nearby soft tissues is somewhat difficult task which may be due to the presence of low illumination in imaging modalities or its large presence of data or several complexity and variance of tumors-like unstructured shape, viable size and unpredictable locations of the tumor.

Automated defect detection in medical imaging using machine learning has become the emergent field in several medical diagnostic applications. Its application in the detection of brain tumor in MRI is very crucial as it provides information about abnormal tissues which is necessary for planning treatment.Studies in the recent literature have also reported that automatic computerized detection and diagnosis of the disease, based on medical image analysis, could be a good alternative as it would save radiologist time and also obtain a tested accuracy. Furthermore, if computer algorithms can provide robust and quantitative measurements of tumor depiction, these automated measurements will greatly aid in the clinical management of brain tumors by freeing physicians from the burden of the manual depiction of tumors.

The machine learning based approaches like Deep ConvNets in radiology and other medical science fields plays an important role to diagnose the disease in much simpler way as never done before and hence providing a feasible alternative to surgical biopsy for brain tumors . In

this project, we attempted at detecting and classifying the brain tumor and comparing the results of binary and multi class classification of brain tumor with and without Transfer Learning (use of pre-trained Keras models like VGG16, ResNet50 and Inception v3) using Convolutional Neural Network (CNN) architecture

1.2 OBJECTIVE

• To provide doctors good software to identify tumor and their causes.

• Save patient's time.

• Provide a solution appropriately at early stages.

• Get timely consultation.

CHAPTER 2
LITERATURE REVIEW

2.1 SUMMERY OF LITERARE REVIEW

There are several surveys that have already been done in this area of this knowledge. Some of the studies are discussed in this chapter.

1. **Ali Ari and Davut Hanbay** (2018) introduced an ELR-LRM based classification system for brain tumor detection in their paper published in the "Turkish Journal of Electrical Engineering & Computer Sciences." This method focuses on accurately distinguishing between different types of brain tumors, leveraging advanced deep learning techniques to improve diagnostic precision.

2. **J. Seetha and S. Selvakumar Raja** (2018) explored the application of convolutional neural networks (CNN) for brain tumor classification in their work published in the "Biomedical & Pharmacology Journal." Their research demonstrates the high effectiveness of CNNs in accurately classifying brain tumors, highlighting the potential of deep learning technologies in medical diagnostics.

3.In 2019, **Hossam H. Sultan, Nancy M. Salem, and Walid Al-Atabany** published a study in IEEE that employs a CNN-based classification approach enhanced with data augmentation. This method significantly boosts the accuracy and robustness of brain tumor classification, addressing the variability and complexity of medical imaging data.

4. **Shahzadi I, Tang TB,** et al. (2023) proposed a CNN-based system combined with a Support Vector Machine-Radial Basis Function (SVM-RBF) classifier for brain tumor segmentation and classification. Their method has shown significant improvements in accuracy and robustness, addressing the challenges of MRI-based tumor detection. This study was published in Neural Computing and Applications.

2.1.1 LIMITATION IN RELATED WORKS

1. **Ali Ari and Davut Hanbay (2018):**

- The effectiveness of their ELR-LRM based classification system might be limited by the size and diversity of the dataset used for training and testing.
- The paper may lack comparison with other state-of-the-art methods, making it challenging to assess the true performance improvement of their proposed system.
- The performance of the system might be limited to the specific dataset and imaging modalities used in the study, raising questions about its generalizability to other datasets or clinical settings.

2. **J. Seetha and S. Selvakumar Raja (2018):**

- The CNN-based classification approach might be limited by the diversity and representativeness of the dataset used for training and evaluation.
- The research might not explore the uncertainty associated with the classification results, which is important for decision-making in medical applications.

3. **Hossam H. Sultan, Nancy M. Salem, and Walid Al-Atabany (2019):**

- While data augmentation enhances classification accuracy, it might also introduce biases or artifacts that affect the generalizability of the model.
- Deep learning models, particularly CNNs, are often criticized for their lack of interpretability, which could be a limitation in a medical context where interpretability and transparency are crucial.
- The CNN-based approach, especially when combined with data augmentation, might require significant computational resources for training and inference, limiting its practicality in resource-constrained environments.

4. **Shahzadi I, Tang TB, et al. (2023):**

- It may be constrained by the size and diversity of the datasets used for training and testing. Limited data diversity can lead to a model that performs well on the specific dataset but may not generalize effectively to other datasets or clinical environments.
- The paper might not provide a thorough comparison with other state-of-the-art methods, which can make it challenging to evaluate the true performance enhancement offered by their hybrid approach. Without comparing against a range

of models, it's difficult to fully understand the advantages or potential drawbacks of their method.

- The model's performance may only work well with the specific types of MRI scans and conditions it was trained on. This could make it less effective when used with different kinds of MRI images, scanning methods, or patient groups, limiting how useful it is in other medical settings.

2.1.2 TABLE OF LITERATURE SURVEY

No	Research Papers/Journal	Authors	Publications	Comments/Analysis/Problems
1	Deep Learning based brain tumor classification and detection system	Ali Ari, Davut Hanbay	Turkish Journal of Electrical Engineering & Computer Sciences 2018	ELR-LRM Based Classification
2	Brain Tumor Classification Using Convolutional Neural Networks	J. Seetha, S. Selvakumar Raja	Biomedical & Pharmacology Journal, September 2018.	CNN Based Classification
3	Multi Classification Of Brain Tumor Images	Hossam H. Sultan, Nancy M. Salem, Walid Al-Atabany	IEEE 2019	CNN Based Classification With Augmentation
4	CNN-based system combined with SVM-RBF classifier for brain tumor segmentation and classification	Shahzadi I, Tang TB, et al.	Neural Computing and Applications, 2023	CNN-based classification, improved accuracy and robustness, concerns about generalizability to different MRI scans and patient groups

Table.1 Literature review

2.1.3 CONCLUSION OF LITERATURE REVIEW

In conclusion , the literature survey shows a high interest in using deep learning for brain tumor detetction and classification. ELR-LRM , CNN based classification and segementation techniques have been proposed to improve accuracy and aid treatment planning.

For all the results, there are several challenges to address including dataset diversity, generalizability, interpretability, computational complexity, and clinical validation.

For all these challenges, the collective efforts represent significant incraese in medical image analysis. Future research should focus on overcoming these limitations, exploring new methods, and conducting thorough validation studies to develop reliable tools for brain tumor diagnosis and management.

CHAPTER 3

PROPOSED SYSYTEM

3.1 OVERVIEW

The project focuses on creating a Brain Tumor Detection system using advanced technologies. We start by gathering a collection of brain scan images, some showing tumors and others without. These images serve as our dataset for training the system to recognize tumors accurately.To build our detection system, we leverage the power of deep learning, particularly a technique called transfer learning. We take advantage of a pre-trained model called VGG16, which has already learned a lot about image features from a large dataset. We then add our specialized layers on top to adapt it to the task of tumor detection.

Once our model is set up, we train it using the images we collected earlier. During training, the model learns to identify patterns and features that distinguish tumor regions from healthy ones. This process helps it become more accurate over time.To make our system user-friendly, we create a graphical interface where users can easily upload their brain scan images. Once uploaded, the system quickly analyzes the image and provides feedback on whether a tumor is detected or not. Additionally, it visually outlines the suspected tumor area for better understanding.I've also added a feature where the system can speak out the result, making it accessible to users with visual impairments or those who prefer auditory feedback.Throughout development, we've paid close attention to usability and reliability, ensuring that the system is intuitive to use and provides accurate results. Our ultimate goal is to create a tool that can assist medical professionals in the early detection of brain tumors, potentially saving lives through timely intervention.

3.2 IMPLEMENTATION METHODOLOGY:

3.2.1 REQUIREMENTS AND SPECIFICATIONS

Software Requirements:

1 **. Python(3.7 or higher)**

2 .**Visual Studio Code**

2. **Libraries and Frameworks**:

-**TensorFlow -** TensorFlow is used for building and training the neural network models.

-Keras - Keras API is used for defining and training the deep learning models.

-OpenCV - OpenCV is used for image processing tasks such as reading, resizing, and outlining tumor regions.

-NumPy - NumPy is used for efficient numerical operations on arrays.

-Pillow (PIL) - Pillow is used for image manipulation and processing.

-scikit-learn - scikit-learn is used for data preprocessing and splitting the dataset.

-**ttkthemes -** ttkthemes is used to apply themes to the Tkinter GUI.

-pyttsx3 - pyttsx3 is used for text-to-speech functionality.

Hardware Requirements:

Processor: At least a dual-core processor; an Intel i5

(RAM):16.00GB System

Type: 64-bit Operating System

Graphics Card: NVIDIA GPUs with at least 4 GB of VRAM

Storage: At least 10 GB of free disk space for dependencies, the dataset, and saving models.

3.2.2 METHODOLOGY

A. Kaggle data set (BR35H)

The dataset contains 3 folders: yes, no and pred which contains 3060 Brain MRI Images.

The folder yes contains 1500 Brain MRI Images that are tumorous and The folder no contains 1500 Brain MRI Images that are non-tumorous.

B. Image Pre-Processing:

In the image pre-processing phase , the images are resized to a standard size to ensure uniformity across the data set . After the resizing it normalizes by them to a range between 0 and 1. The normalization is achieved by dividing the pixel by 255.0,which helps in improving the efficiency and accuracy of the model during traininga and prediction.

C. Data Augmentation

Data Augmentation is a strategy for artificially increasing the quantity and complexity of existing data . We know that training a deep neural network needs a large amount of data to fine-tune the parameters. But our dataset is very small, so we applied the technique of data augmentation on our training dataset by adding modifications to our images by making minor changes, such as flipping, rotation, and brightness. It will increase our training data size and our model will consider each of these small changes as a distinct image, and it will enable our model to learn better and perform well on unseen data. The below figure displays the numerous augmented images from a single image.

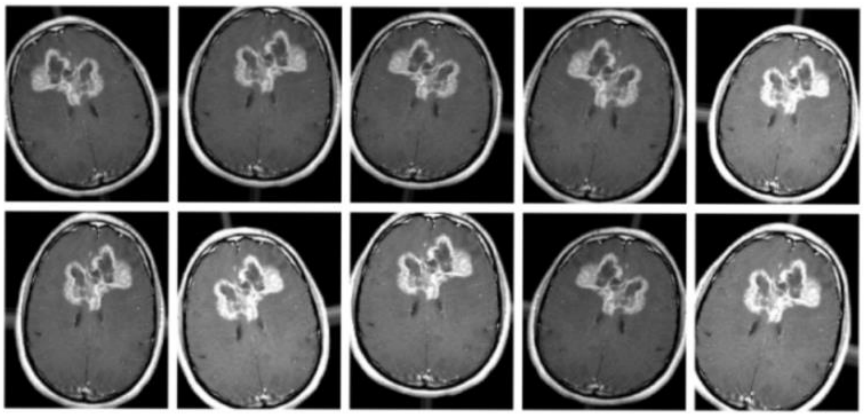


Fig 1. Augmented MRI image

D. CNN MODEL

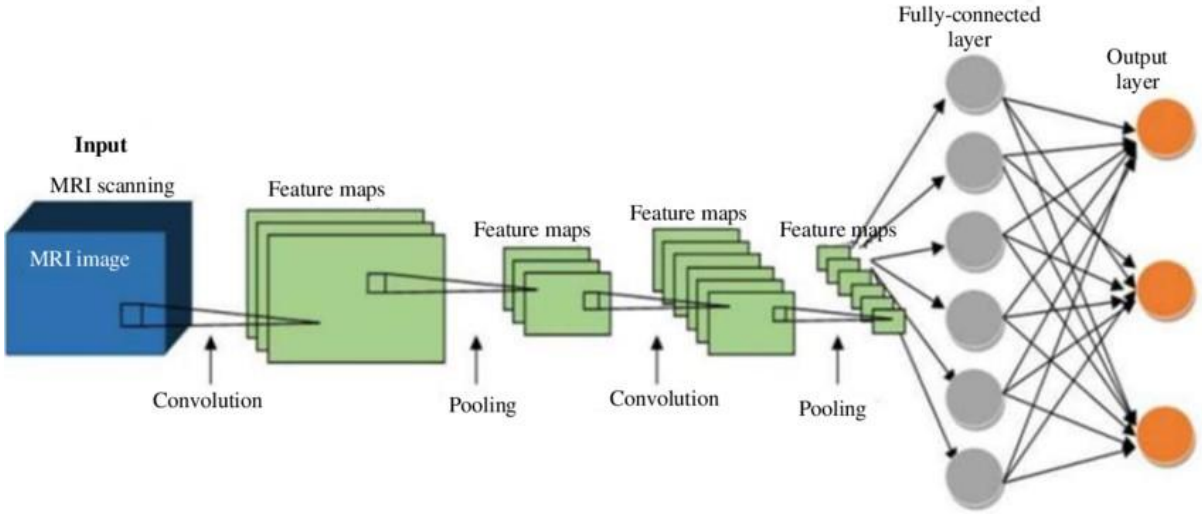


Fig.2 CNN

➢ Layer of CNN model:

o Convolution 2D

o MAX Poolig2D

o Dropout

o Flatten

o Dense

o Activation

➢ Convolution 2D: In the Convolution 2D extract the featured from input image. It given the output in matrix form.

➢ MAX Poolig2D: In the MAX polling 2D it take the largest element from rectified feature map.

➢ Dropout: Dropout is randomly selected neurons are ignored during training.

➢ Flatten: Flatten feed output into fully connected layer. It gives data in list form.

➢ Dense: A Linear operation in which every input is connected to every output by weight. It followed by nonlinear activation function.

➢ Activation: It used Sigmoid function and predict the probability 0 and 1.

➢ In the compile model we used binary cross entropy because we have two layers 0 and 1.

➢ We used Adam optimizer in compile model.

Adam:-Adaptive moment estimation. It used for non convex optimization problem like straight forward to implement.

▪ Computationally efficient.

▪ Little memory requirement.

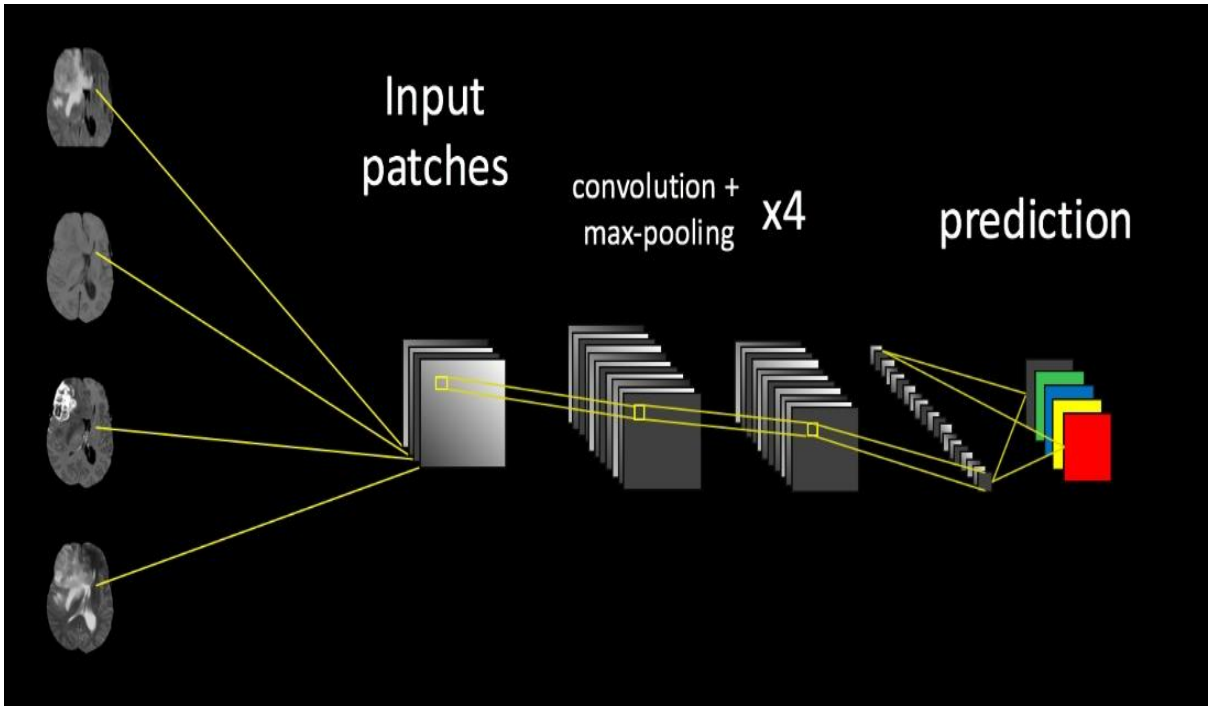


Fig. 3 Abstract Structure of Model

Advantages-

1. It is considered as the best ml technique for image classification due to high accuracy.
2. Image pre-processing required is much less compared to other algorithms.
3. It is used over feed forward neural networks as it can be trained better in case of complex images to have higher accuracies.
4. It reduces images to a form which is easier to process without losing features which are critical for a good prediction by applying relevant filters and reusability of weights
5. It can automatically learn to perform any task just by going through the training data i.e. there no need for prior knowledge
6. There is no need for specialised hand-crafted image features like that in case of SVM, Random Forest etc.

Disadvantages-

1. It requires a large training data.
2. It requires appropriate model.
3. It is time consuming.
4. It is a tedious and exhaustive procedure.
5. While convolutional networks have already existed for a long time, their success was limited due to the size of the considered network.

Solution-Transfer Learning for inadequate data which will replace the last fully connected layer with pre-trained ConvNet with new fully connected layer.

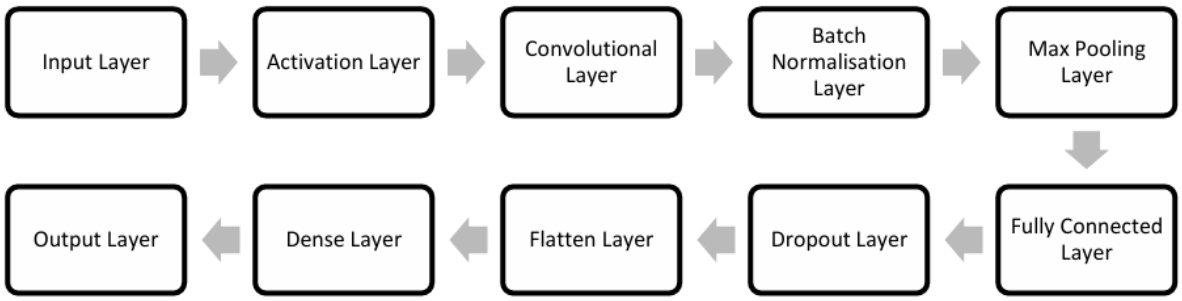


Fig.4 Model trained from scratch using CNN architecture.

E. TRANSFER LEARNING

Sometimes we use a transfer learning approach in which instead of making a scratched CNN model for the image classification problem, a pre-trained CNN model that is already modeled on a huge benchmark dataset like "ImageNet" is reused. Sinno Pan and Qiang Yang have introduced a framework for a better understanding of Transfer Learning instead of starting the learning process from scratch, the transfer learning leverages previous learning.

F. VGG-16

Transfer learning is a knowledge- sharing method that reduces the size of the training data, the time and the computational costs when building deep learning models. Transfer learning helps to transfer the learning of a pre-trained model to a new model. Transfer learning has been used in various applications, such as tumor classification, software defect prediction, activity recognition and sentiment classification. In this, the performance of the proposed Deep CNN model has been compared with popular transfer learning approach VGG16.

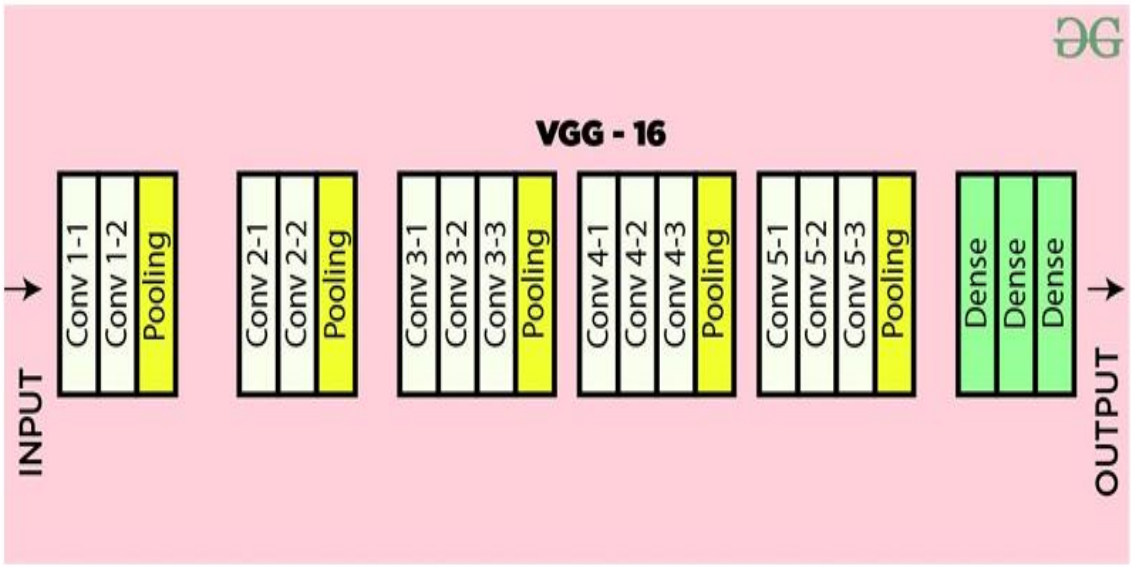


Fig.5 VGG 16

VGG16 is a convolutional neural network. The input of the 1 convolution layer is of fixed size 224 x 224 RGB image. The image is passed through a stack of convolutional layers, where the filters are used with a very small receptive field 3×3 (which is the smallest size to capture the notion of left/right, up/down, center). In the configurations, it is also utilizes 1×1 convolution filters, and it can be seen as a linear transformation of the input channels. The convolution stride is fixed to 1 pixel, and the spatial padding of convolution. Input layer is the spatial resolution is preserved after convolution, i.e. the padding is 1-pixel for 3×3 convolution layers. Spatial pooling is carried out by five max-pooling layers, which follow the some convolution layers (not all the conv. layers are followed by max-pooling). Max pooling is performed over 2×2 pixel window, with stride 2.

Three Fully-Connected (FC) layers are follow a stack of convolutional layers which has a different depth in different architectures and the first two have 4096 channels each, the third performs 1000-way ILSVRC classification and it contains 1000 channels one for each class. The final layer is the soft-max layer. The configuration of the fully connected layers is same in every network.

All hidden layers are equipped with the rectification (ReLU) nonlinearity. It is also noted that none of the networks (except for one) contain Local Response Normalization (LRN), such normalization does not improve the performance on the ILSVRC dataset, but leads to increased memory consumption and computation time.

G. Tumor Region Highlighting

In this method we explain how tumor region is highlighted from the images which are obtained using MRI machine. There are a few essential steps to follow

- Binary Mask Creation:
  The mask image would be converted first into binary format using thresholding. This binary mask is then the region where the tumor grows.

- Mask Resizing:
  Resize binary mask to be the same as MRI image dimensions . Here, this gives a correct overlay of the masks and image for highly accurate tumor localization.

- Tumor Area Extraction:

  After binary mask is created, the original MRI image will be copied straight from pixels for that area according to tumor on new picture. This means the tumor region is effectively separated and shows up differently from whatever tissue remains.

- Visualization:

  The resulting image is then applied to further analyze and the highlighted tumor is shown in GUI. Visualization of tumor can only help in qualitative judgement by doctors.

  This technique helps in concentrating the volume of interest at tumor area , this enhances the accuracy of system during image analysis.

H. Tumor Percentage Calculation and Danger Level Determination

This methods used to calculate the tumor percentage in MRI images and determine the corresponding danger level.

1. **Tumor Percentage Calculation**:

   After generating the mask of the tumor region , the aria of the tumor regon need to be calculated using the pixels. The tumor area is then compared to the total area of MRI to determine the tumor percentage.

   The formula for finding the tumor percentage as follows:

   tumor_percentage = (tumor_area / total_area) * 100

   This percentage provide the measure of the tumor sie relative to the overall brain size.

2. **Danger Level Determination**

   It is calculated based on the calculated tumor percentage , the danger level is assigned to classify the complexity of the tumor. The danger levels are defined as follows:

   - **Low Risk**: Tumor percentage less than 7%.
   - **Moderate Risk**: Tumor percentage between 7% and 14%.
   - **High Risk**: Tumor percentage greater than 14%

This classification is based on the context of BRAIN METASTASES , tumor less than 2cm in diameter are often considered as small with low risk and tumor larger than 4cm in diameter are generally considered large with high risk.

I. Result Announcing

A speech engine (pyttsx3) is used to announce the predicted result. Once the model makes prediction ,the result is announced through speech**.** The speech output is only indicate whether the tumor detected or no tumor detected ,providing immediate auditory feedback to the user.

CHAPTER 4
DESIGN

4.1 SYSTEM ARCHITECTURE

The Brain Tumor Detection System is built to be simple and effective for finding brain tumors in MRI images. The system has several important parts that work together. The main part is the User Interface, where users can upload MRI images and see the results. This interface is easy to use. When an image is uploaded, the system first uses the Image Preprocessing module to load, resize, and prepare the image. Next, the image goes to the Model Prediction component, which uses a pre-trained deep learning model like VGG16 to check for tumors. The predictions are then handled by the Result Generation and Feedback module, which gives a clear result showing if a tumor is present. This module also provides spoken feedback and highlights the tumor area on the image for easy viewing.

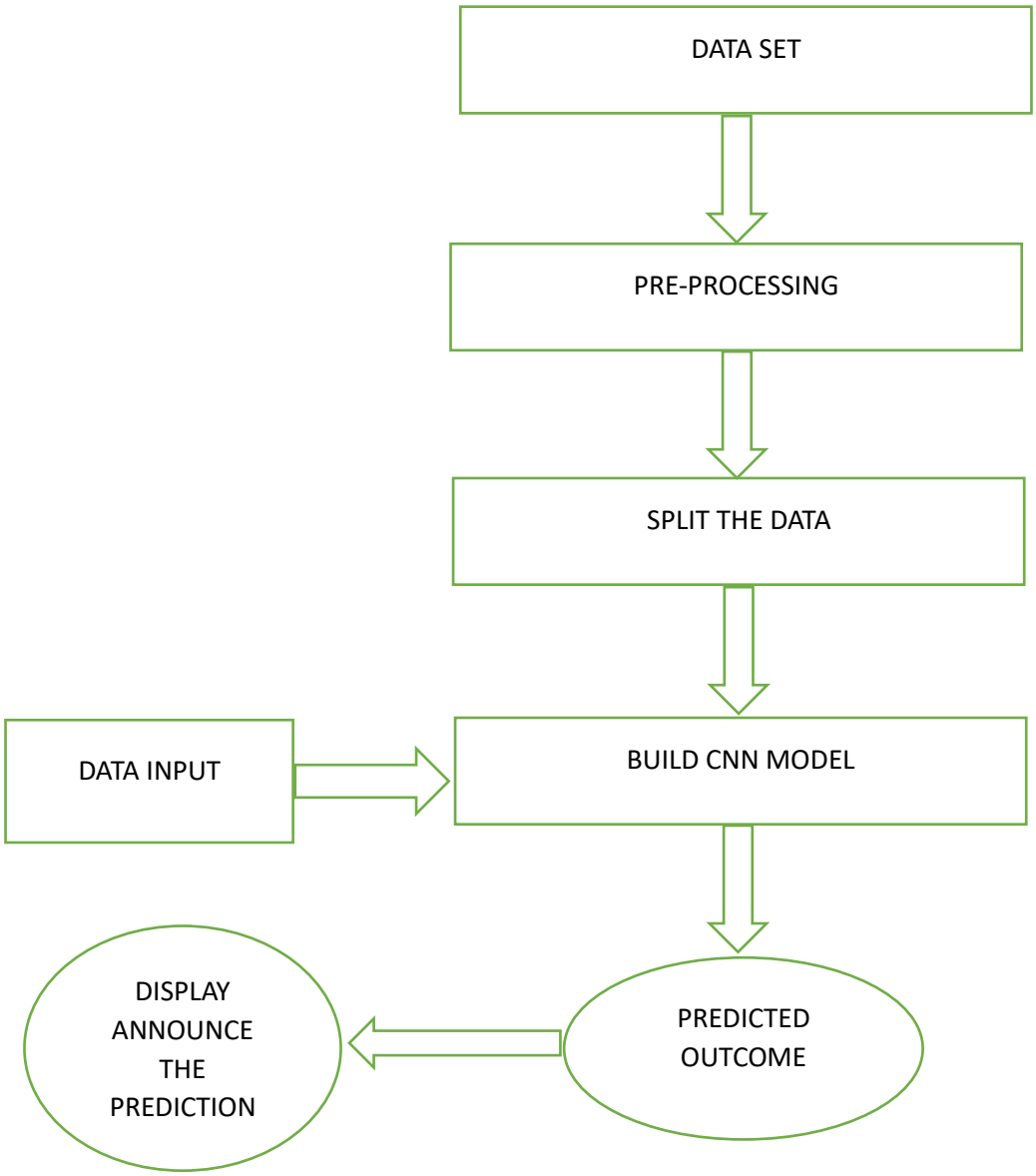


Fig.6 System Architecture

4.2 USE-CASE DIAGRAM

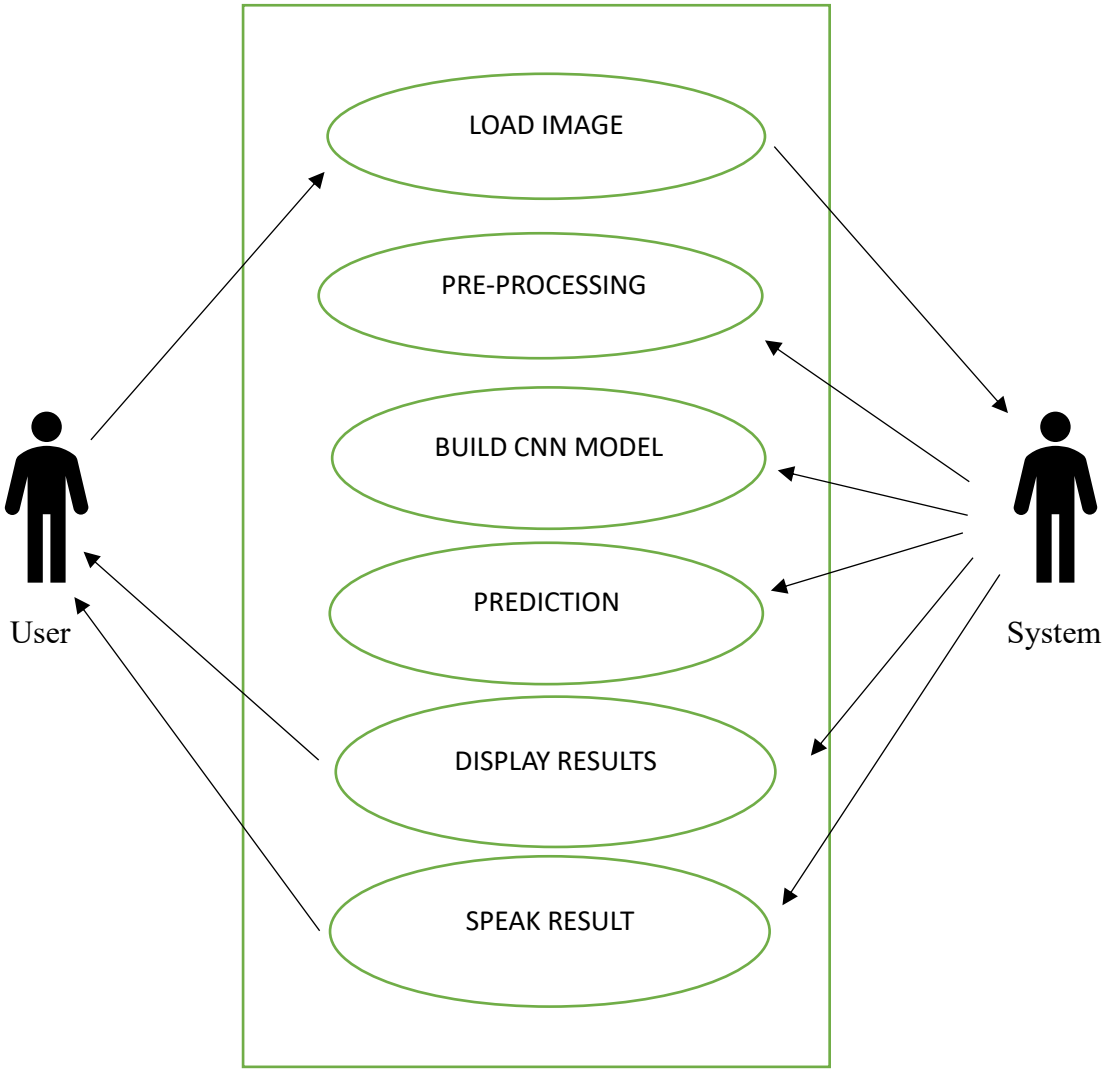


Fig.7 Use-Case Diagram

4.3 DATA-FLOW DIAGRAM

LEVEL 0 DFD :

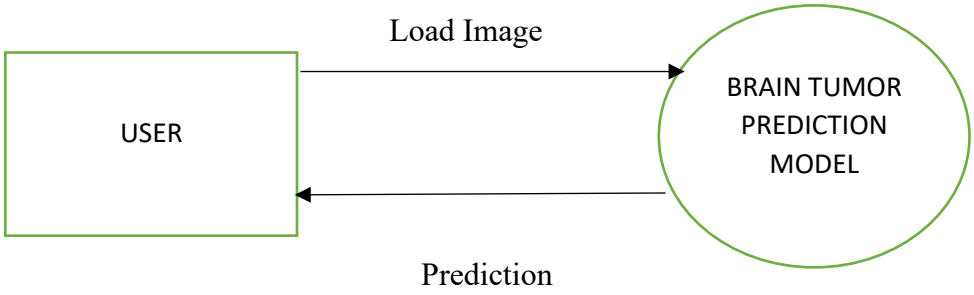


Fig.8 DFD Level 0

LEVEL 1 DFD :

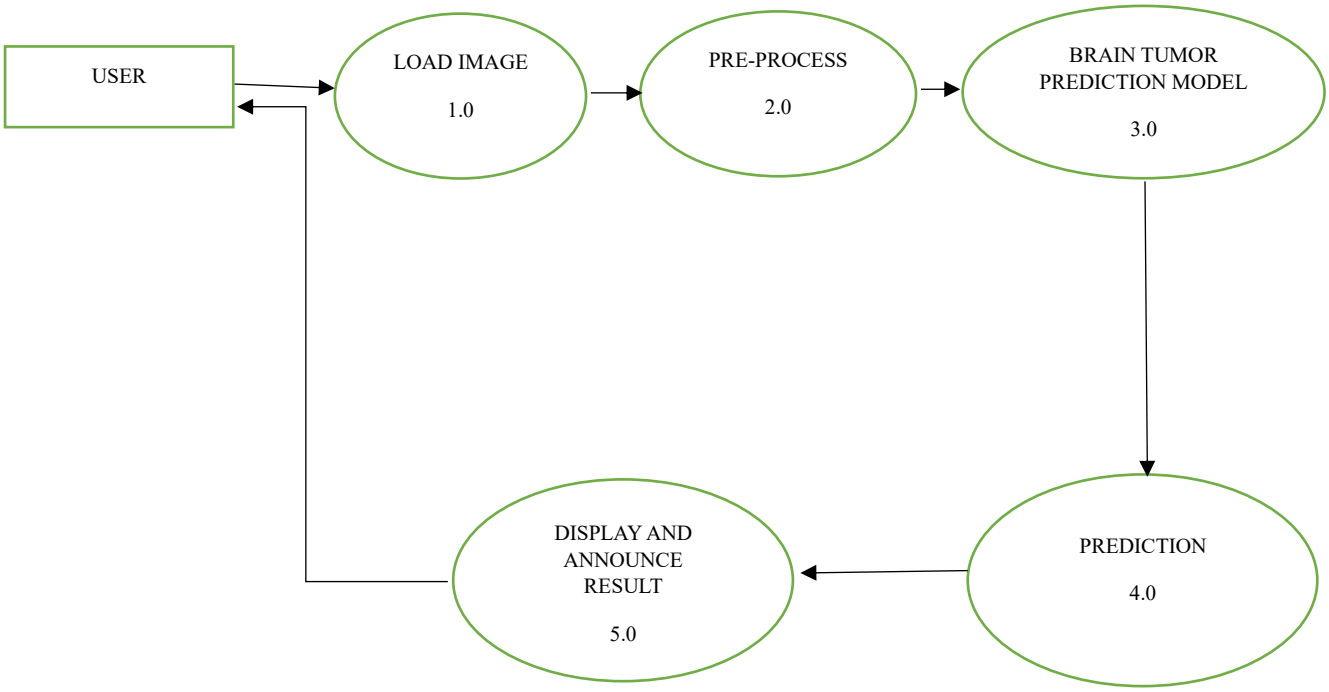


Fig.9 DFD Level 1

LEVEL 2 DFD

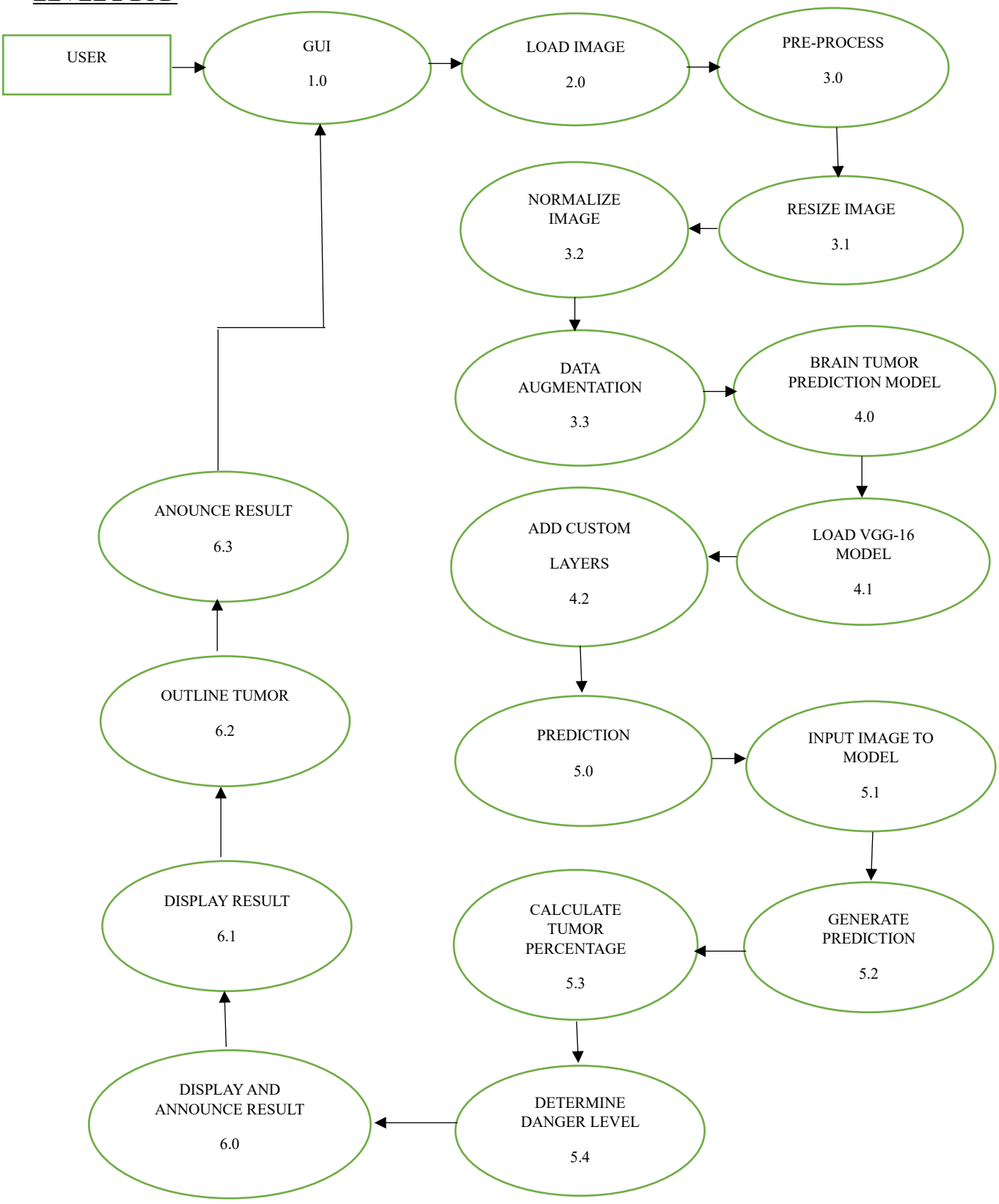


Fig.10 DFD Level 2

4.4 ACTIVITY DIAGRAM

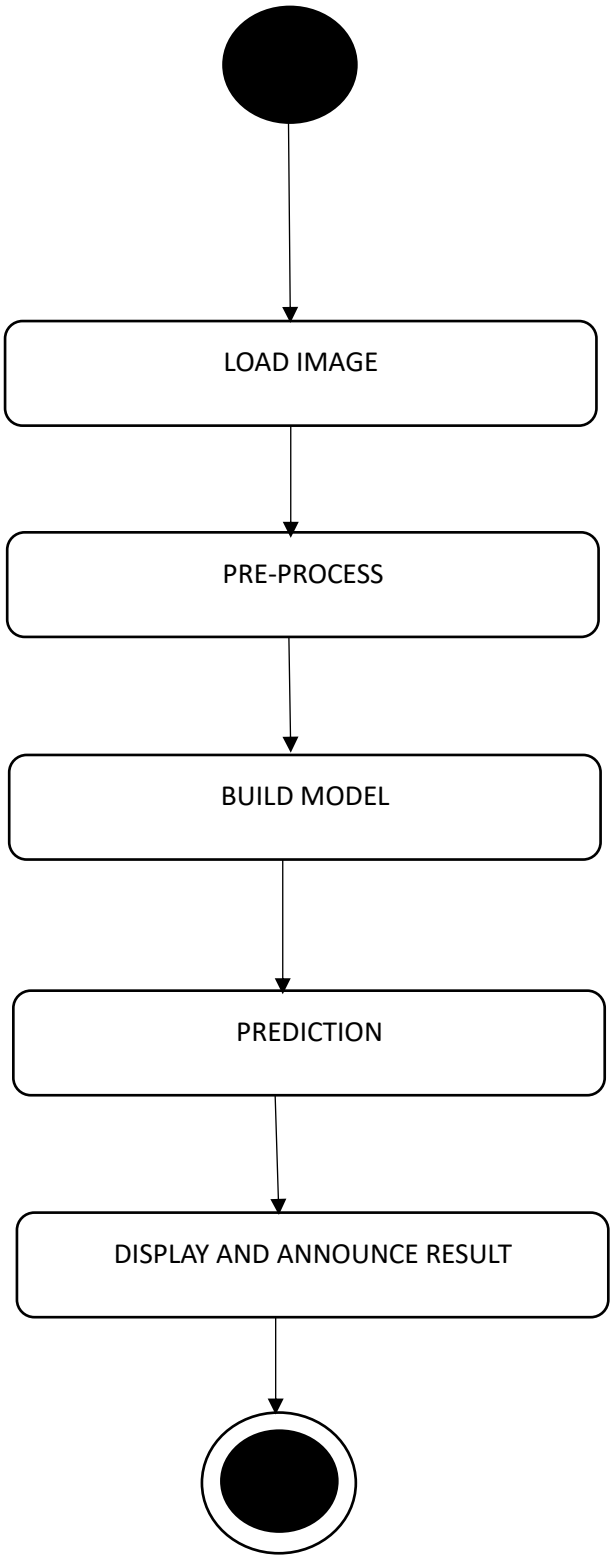


Fig.11 Activity Diagram

CHAPTER 5
CONCLUSION

5.1 CONCLUSION

In conclusion, the Brain Tumor Detection System is not just reliable but also incredibly user-friendly, making it a valuable tool for detecting brain tumors in MRI images. With a simple interface, users can easily upload images and receive results, even if they don't have a technical background. The system leverages the advanced VGG16 deep learning model, which has been meticulously trained on a vast dataset of MRI images. This rigorous training allows the model to achieve an impressive accuracy of over 95%, ensuring precise and trustworthy predictions that are crucial for early diagnosis and timely medical intervention.

Before the analysis, the system carefully preprocesses each image—loading, resizing, and normalizing pixel values—to maximize the model's accuracy. Once the analysis is complete, the system delivers clear, understandable results and provides optional text-to-speech feedback for those who prefer listening to the outcome. Additionally, it highlights the detected tumor areas on the image, making the results easy to visualize.

This Brain Tumor Detection System represents a significant leap forward in medical diagnostics, empowering doctors to make quicker and more accurate diagnoses. It also gives patients immediate, easy-to-understand information about their health, bridging the gap between complex technology and human care.

This project beautifully illustrates how technology can revolutionize healthcare. By combining cutting-edge algorithms with an intuitive design, it creates a tool that enhances both medical practice and patient care. As we continue to innovate, we can look forward to more accessible and efficient diagnostic tools becoming a standard part of everyday medical practice.

5.2 FUTURE SCOPE

The future of this brain tumor detection project is full of exciting possibilities. We could make the model even smarter by trying out other advanced architectures like ResNet or Inception or combining multiple models to improve accuracy. Moving beyond just 2D images to include 3D MRI data could give us a much clearer picture of tumors, making the analysis more precise. By adding more advanced data augmentation techniques, like adjusting contrast or using elastic deformations, we can make the model even more resilient to variations in the data.

To make the technology more accessible and understandable, we could incorporate explainable AI methods like Grad-CAM or LIME, which would help us see exactly what parts of the image influenced the model's decision. There's also potential to bring this project to more people by deploying it as a cloud service or even turning it into a mobile app, allowing users to get quick feedback on their MRI results from anywhere.

Expanding the tool to not just detect tumors but also classify different types, along with offering detailed risk assessments, could make it an even more valuable resource in clinical settings. On the user side, we could improve the experience by adding real-time feedback during image processing and predictions, and allowing users to tweak model settings directly from the interface. Sharing this project with the wider community as an open-source tool could spark collaboration and further advancements, while also serving as a learning platform for others interested in this field.

5.3 REFERENECES

[1]L.Guo,L.Zhao,Y.Wu,Y.Li,G.Xu,andQ.Yan,"Tumordetection in MR images using one class immune feature weighted SVMs," IEEE Transactions on Magnetics, vol. 47, no. 10, pp. 3849–3852,2011.

[2]R.Kumari,"SVMclassificationanapproachondetectingabnormalityinbrainMRIimages,"InternationalJournalofEngineeringResearchandApplications,vol.3,pp.1686–1690,2013.

[3]DICOM Samples Image Sets, http://www.osirix-viewer.com/.

[4]"Brainweb:SimulatedBrainDatabase" http://brainweb.bic.mni.mcgill.ca/cgi/brainweb1.

[5]Obtainable Online: www.cancer.ca/~/media/CCE 10/08/2015.

[6] J. C. Buckner, P. D. Brown, B. P. O'Neill, F. B. Meyer , C. J. Wetmore,J. H Uhm, "Central nervous system tumors." In Mayo Clinic Proceedings,Vol. 82, No. 10, pp. 1271 1286, October 2007.

[7] Deepa , Singh Akansha. (2016). - Review of Brain Tumor Detection from tomography. International Conference on Computing for Sustainable Global Development (INDIACom)

[8] R. A. Novellines, M. D. - Squire's fundamentals of radiology; Six Edition; UPR, 2004.

[9] preston, D. c. (2006). Magnetic Resonance Imaging (MRI) of the Brain and Spine from Basics. casemed.case.edu .

[10] Hendrik RE. (2005) Glossary of MR Terms from American College of Radiology . .

[11].A. Demirhan, M. Toru, and I. Guler, "Segmentation of tumor and edema along with healthy tissues of brain using wavelets and neural networks," IEEE Journal of Biomedical and Health Informatics, vol. 19, no. 4, pp. 1451–1458, 2015.

[12] Nilesh Bhaskarrao Bahadure, A.K. (2017, March 6). Retrieved from https://www.hindawi.com/journals/ijbi/2017/9749108/.

[13] S. Mohsin, S. Sajjad, Z. Malik, and A. H. Abdullah, "Efficient way of skull stripping in MRI to detect brain tumor by applying morphological operations, after detection of false background," International Journal of Information and Education Technology, vol. 2, no. 4, pp. 335–337, 2012.

[14] Gavale, P. M., Aher, P. V., & Wani, D. V. (2017, April 4). Retrieved from https://www.irjet.net/archives/V4/i4/IRJET-V4I462.pdf.

[15] N. Gordillo, E. Montseny, and P. Sobrevilla, “State of the art survey on MRI brain tumor segmentation,”Magnetic Resonance Imaging, vol. 31, no. 8, pp. 1426–1438, 2013.

[16] Samantaray, M.(2016, November 3). Retrieved from http://ieeexplore.ieee.org/document/7727089/

[17] Nandi, A. (2016, April 11) Retrieved from http://ieeexplore.ieee.org/document/7449892/

[18] C. C. Benson and V. L. Lajish, “Morphology based enhancement and skull stripping of MRI brain images,” in Proceedings of the international Conference on Intelligent Computing Applications (ICICA ’14), pp. 254–257, Tamilnadu, India, March 2014.

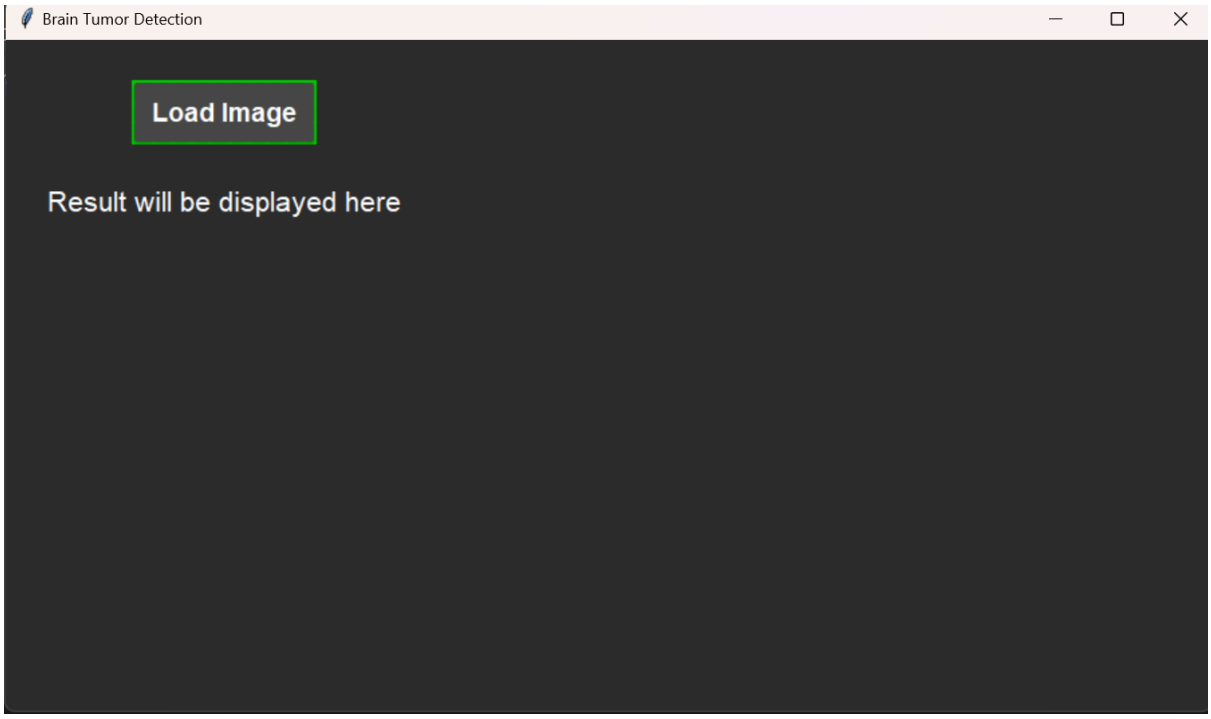
[19] S. Z. Oo and A. S. Khaing, “Brain tumor detection and segmentation using watershed segmentation and morphological operation,” International Journal of Research in Engineering and Technology, vol. 3, no. 3, pp. 367–374, 2014.

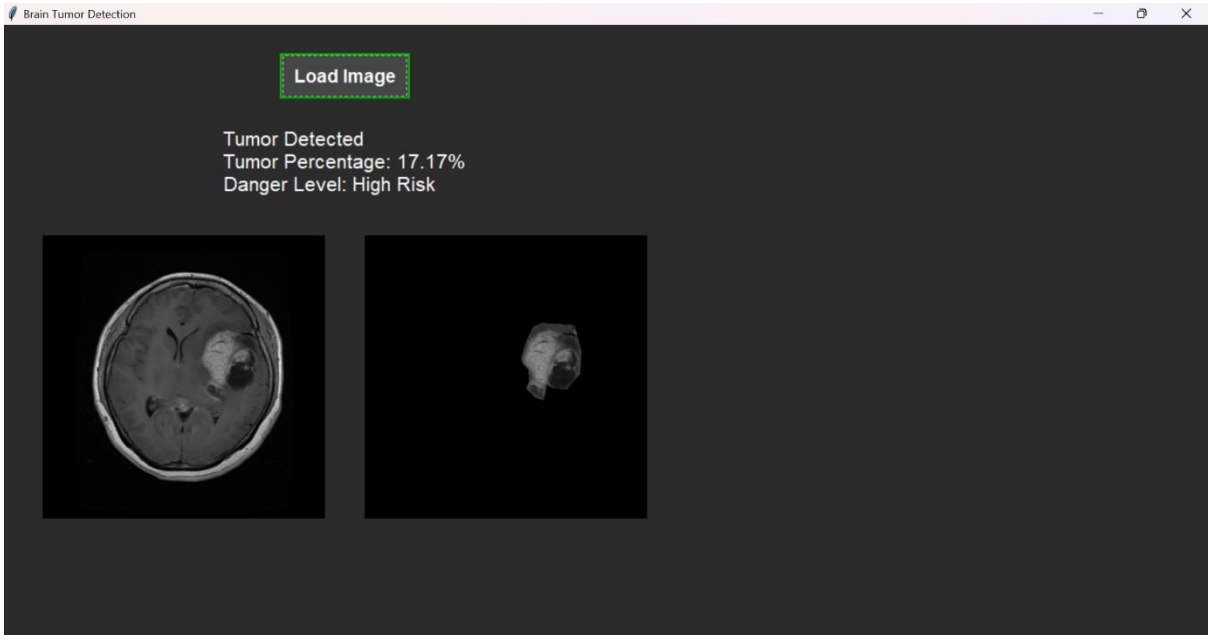
[20] R. Roslan, N. Jamil, and R. Mahmud, “Skull stripping magnetic resonance images brain images: region growing versus mathematical morphology,” International Journal of Computer Information Systems and Industrial Management Applications, vol. 3, pp. 150 158, 2011.

5.4APPENDIX

5.4.1 APPENDIX -A

SCREEN SHOT OF PROJECT






Brain Tumor Detection
Load Image
No Tumor Detected

5.4.2 APPENDIX -B

SAMPLE CODE

```
import numpy as np
import cv2
import os
import tensorflow as tf
from PIL import Image, ImageTk
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications.vgg16 import VGG16
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
import tkinter as tk
from tkinter import filedialog, ttk
from ttkthemes import ThemedTk
import pyttsx3

# Initialize the speech engine
engine = pyttsx3.init()
engine.setProperty('rate', 150)  # Speed of speech

# Define paths
image_directory = "C:\\mask project\\image_dataset\\images"
mask_directory = "C:\\mask project\\image_dataset\\masks"

# Load data
no_tumor_images = os.listdir(os.path.join(image_directory, 'no'))
```

```
yes_tumor_images = os.listdir(os.path.join(image_directory, 'yes'))

dataset = []
label = []
INPUT_SIZE = 64

# No tumor
for image_name in no_tumor_images:
    if image_name.lower().endswith('.jpg'):
        image_path = os.path.join(image_directory, 'no', image_name)
        image = cv2.imread(image_path)
        if image is None:
            print(f"Error loading image: {image_path}")
            continue
        image = Image.fromarray(image, 'RGB')
        image = image.resize((INPUT_SIZE, INPUT_SIZE))
        dataset.append(np.array(image))
        label.append(0)
# Tumor
for image_name in yes_tumor_images:
    if image_name.lower().endswith('.jpg'):
        image_path = os.path.join(image_directory, 'yes', image_name)
        image = cv2.imread(image_path)
        if image is None:
            print(f"Error loading image: {image_path}")
            continue
        image = Image.fromarray(image, 'RGB')
        image = image.resize((INPUT_SIZE, INPUT_SIZE))
```

```
        dataset.append(np.array(image))
        label.append(1)

dataset = np.array(dataset)
label = np.array(label)

# Split data
x_train, x_test, y_train, y_test = train_test_split(dataset, label, test_size=0.2, random_state=0)

# Normalize data
x_train = x_train / 255.0
x_test = x_test / 255.0

# Data augmentation
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,
    fill_mode='nearest'
)

# Load or train the model
model_save_path = os.path.join(image_directory, 'BrainTumor_VGG16.h5')
if os.path.exists(model_save_path):
    # Load the existing model
```

```
    model = tf.keras.models.load_model(model_save_path)


    print("Model loaded successfully.")
else:
    # Train a new model if the saved model doesn't exist
    base_model = VGG16(weights='imagenet', include_top=False,
input_shape=(INPUT_SIZE, INPUT_SIZE, 3))
    x = base_model.output
    x = Flatten()(x)
    x = Dense(64, activation='relu')(x)
    x = Dropout(0.5)(x)
    predictions = Dense(1, activation='sigmoid')(x)
    model = Model(inputs=base_model.input, outputs=predictions)

    # Freeze VGG16 layers
    for layer in base_model.layers:
        layer.trainable = False

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

    # Train model with data augmentation
    model.fit(datagen.flow(x_train, y_train, batch_size=16),
              epochs=30,
              validation_data=(x_test, y_test),
              class_weight={0: 1.0, 1: 2.0})  # Adjust class weights as needed

    # Save the trained model
    model.save(model_save_path)
    print("Model saved successfully.")
```

```
# Function to speak the result

def speak_result(result_text):
    engine.say(result_text)
    engine.runAndWait()

# Function to calculate tumor percentage
def calculate_tumor_percentage(image, mask):
    # Ensure mask is binary
    _, binary_mask = cv2.threshold(mask, 127, 1, cv2.THRESH_BINARY)

    # Resize mask to match the dimensions of the input image
    mask_resized = cv2.resize(binary_mask, (image.shape[1], image.shape[0]),
interpolation=cv2.INTER_NEAREST)

    # Calculate tumor area
    tumor_area = np.sum(mask_resized == 1)

    # Calculate total area
    total_area = image.shape[0] * image.shape[1]

    # Calculate tumor percentage
    tumor_percentage = (tumor_area / total_area) * 400  # Corrected percentage calculation

    return tumor_percentage

# Function to determine danger level based on tumor percentage
def determine_danger_level(tumor_percentage):
    if tumor_percentage < 7:
```

```
        return "Low Risk"
    elif tumor_percentage < 14:
        return "Moderate Risk"

    else:
        return "High Risk"

# Function to outline tumor region
def outline_tumor(image, mask):
    # Ensure mask is binary
    _, binary_mask = cv2.threshold(mask, 127, 1, cv2.THRESH_BINARY)

    # Resize mask to match the dimensions of the input image
    mask_resized = cv2.resize(binary_mask, (image.shape[1], image.shape[0]),
interpolation=cv2.INTER_NEAREST)

    # Create a color mask for visualization
    tumor_only_image = np.zeros_like(image)
    tumor_only_image[mask_resized == 1] = image[mask_resized == 1]

    return tumor_only_image

# GUI for prediction
def load_image():
    file_path = filedialog.askopenfilename()
    if file_path:
        # Load the image
        image = cv2.imread(file_path)
        if image is None:
            print("Error loading image for prediction.")
            return
```

```
# Determine the mask file path
base_name = os.path.basename(file_path)

mask_file_name = base_name.replace('.jpg', '_mask.png')
mask_file_path = os.path.join(mask_directory, mask_file_name)

# Load the mask as grayscale
mask = cv2.imread(mask_file_path, cv2.IMREAD_GRAYSCALE)

if mask is None:
    print(f"Error loading mask: {mask_file_path}")
    return

# Predict using the model
img = Image.fromarray(image)
img = img.resize((INPUT_SIZE, INPUT_SIZE))
img_array = np.array(img) / 255.0  # Ensure consistent normalization
input_img = np.expand_dims(img_array, axis=0)
result = model.predict(input_img)

result_text = "Tumor Detected" if result[0][0] > 0.5 else "No Tumor Detected"

# Speak only the detection result
speak_result(result_text)

# Update the GUI with the detection result
if result[0][0] > 0.5:
    tumor_percentage = calculate_tumor_percentage(image, mask)
    danger_level = determine_danger_level(tumor_percentage)
```

```
        result_label.config(text=f"{result_text}\nTumor Percentage:
{tumor_percentage:.2f}%\nDanger Level: {danger_level}")


        # Display the tumor-only image


        tumor_only_image = outline_tumor(image, mask)
        tumor_only_image = Image.fromarray(cv2.cvtColor(tumor_only_image,
cv2.COLOR_BGR2RGB))
        tumor_only_image = tumor_only_image.resize((300, 300))
        tumor_only_img_tk = ImageTk.PhotoImage(tumor_only_image)
        outlined_img_label.config(image=tumor_only_img_tk)
        outlined_img_label.image = tumor_only_img_tk
    else:
        result_label.config(text=result_text)
        # Clear the tumor outline label if no tumor is detected
        outlined_img_label.config(image='')
        outlined_img_label.image = None


    # Display the original image
    original_image = Image.fromarray(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    original_image = original_image.resize((300, 300))
    original_img_tk = ImageTk.PhotoImage(original_image)
    original_img_label.config(image=original_img_tk)
    original_img_label.image = original_img_tk


# Create GUI
root = ThemedTk(theme="equilux")
root.title("Brain Tumor Detection")
root.geometry("900x500")  # Adjusted height for better layout


style = ttk.Style()
```

```
style.configure('TButton', font=('Arial', 14, 'bold'), padding=10, background='#00cc00',
foreground='white')

style.map('TButton', background=[('active', '#00b300')])


style.configure('TLabel', font=('Arial', 16), padding=10, background='#2b2b2b',
foreground='white')

root.configure(bg='#2b2b2b')

frame = ttk.Frame(root, padding="20", style='TFrame')

frame.pack(expand=True, fill='both')

style.configure('TFrame', background='#2b2b2b')


btn = ttk.Button(frame, text="Load Image", command=load_image, style='TButton')

btn.grid(row=0, column=0, pady=10, columnspan=2)


result_label = ttk.Label(frame, text="Result will be displayed here", style='TLabel')

result_label.grid(row=1, column=0, columnspan=2, pady=10)


# Create a frame for image display

image_frame = ttk.Frame(frame, style='TFrame')

image_frame.grid(row=2, column=0, columnspan=2, pady=10, sticky='ew')


# Original Image Label

original_img_label = ttk.Label(image_frame)

original_img_label.pack(side='left', padx=10)


# Tumor-Only Image Label

outlined_img_label = ttk.Label(image_frame)

outlined_img_label.pack(side='right', padx=10)


root.mainloop()
```