INDUSTRIAL ORIENTED MINI PROJECT

Report

On

BRAIN TUMOR CLASSIFICATION USING DEEP LEARNING

Submitted in partial fulfilment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

INFORMATION TECHNOLOGY

By

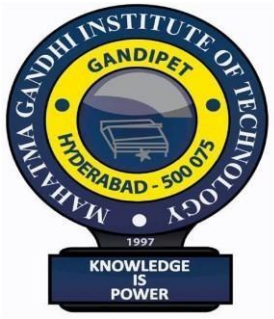
Dendukuri Naga Vaishnavi – 23265A1203

Eshaboina Aravind – 23265A1204

Under the guidance of

Dr. Ch. Prem Kumar

Associate Professor, Department of IT



DEPARTMENT OF INFORMATION TECHNOLOGY

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY (AUTONOMOUS)

(Affiliated to JNTUH, Hyderabad; Eight UG Programs Accredited by NBA; Accredited by NAAC with 'A++' Grade)

Gandipet, Hyderabad, Telangana, Chaitanya Bharati (P.O), Ranga Reddy District, Hyderabad– 500075, Telangana

2024-2025

CERTIFICATE

This is to certify that the **Industrial Oriented Mini Project** entitled **BRAIN TUMOR CLASSIFICATION USING DEEP LEARNING** submitted by **Dendukuri Naga Vaishnavi (23265A1203),** and **Eshaboina Aravind (23265A1204)** in partial fulfillment of the requirements for the Award of the Degree of Bachelor of Technology in Information Technology as specialization is a record of the bona fide work carried out under the supervision of **Dr. Ch. Prem Kumar**, and this has not been submitted to any other University or Institute for the award of any degree or diploma.

Internal Supervisor:

Dr. Ch. Prem Kumar
Associate Professor
Dept. of IT

IOMP Supervisor:

Dr. U. Chaitanya
Assistant Professor
Dept. of IT

EXTERNAL EXAMINER

Dr. D. Vijaya Lakshmi
Professor and HOD
Dept. of IT

DECLARATION

We hear by declare that the **Industrial Oriented Mini Project** entitled **BRAIN TUMOR CLASSIFICATION USING DEEP LEARNING** is an original and bona fide work carried out by us as a part of fulfilment of Bachelor of Technology in Information Technology, Mahatma Gandhi Institute of Technology, Hyderabad, under the guidance of **Dr.Ch. Prem Kumar, Associate Professor**, Department of IT, MGIT.

No part of the project work is copied from books /journals/ internet and wherever the portion is taken, the same has been duly referred in the text. The report is based on the project work done entirely by us and not copied from any other source.

Dendukuri Naga Vaishnavi – 23265A1203

Eshaboina Aravind – 23265A1204

ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of any task would be incomplete without introducing the people who made it possible and whose constant guidance and encouragement crowns all efforts with success. They have been a guiding light and source of inspiration towards the completion of the **Industrial Oriented Mini Project**.

We would like to express our sincere gratitude and indebtedness to our project guide **Dr. Ch. Prem Kumar, Associate Professor**, Dept. of IT, who has supported us throughout our project with immense patience and expertise.

We are also thankful to our honourable Principal of MGIT **Prof. G. Chandramohan Reddy** and **Dr. D. Vijaya Lakshmi**, Professor and HOD, Department of IT, for providing excellent infrastructure and a conducive atmosphere for completing this **Industrial Oriented Mini Project** successfully.

We are also extremely thankful to our IOMP supervisor **Dr. U. Chaitanya**, Assistant Professor, Department of IT, and senior faculty **Mrs. B. Meenakshi,** Assistant Processor, Department of IT for their valuable suggestions and guidance throughout the course of this project.

We convey our heartfelt thanks to the lab staff for allowing us to use the required equipment whenever needed.

Finally, we would like to take this opportunity to thank our families for their support all through the work. We sincerely acknowledge and thank all those who gave directly or indirectly their support for completion of this work.

Dendukuri Naga Vaishnavi – 23265A1203

Eshaboina Aravind -23265A1204

ABSTRACT

This project presents an automation system for detecting and classifying brain tumors from MRI images using deep learning techniques. Leveraging ResNet50, a deep convolutional neural network (CNN) architecture, which is very strong at high performance for image recognition, for recognizing the patterns in association with the kind of tumors, the system accurately identifies and analyze MRI images to classify a given brain scan into one of four categories: Glioma, Meningioma, Pituitary tumor, and Normal (non-tumorous) cases. The model was trained and validated on a labelled dataset of MRI images to ensure robust performance in tumor classification. To enhance usability, we have developed a web application of the system by using Flask, allowing users to upload MRI scans and receive real-time predictions. Experimental results demonstrate the system's effectiveness, achieving high accuracy in multi-class tumor classification. This project aims to aid medical practitioners in early diagnosis and treatment planning by providing a reliable and accessible tool for brain tumor analysis.

Key points: Brain tumor detection, MRI image classification, Deep learning, ResNet-50,

Transfer learning, Medical image analysis, Flask web application, Automated diagnostics.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 MOTIVATION

The brain is a most important organ in the human body which controls the entire functionality of other organs and helps in decision making. It is primarily the control center of the central nervous system and is responsible for performing the daily voluntary and involuntary activities in the human body. A tumor is an abnormal growth of tissue in the brain that develops uncontrollably and can interfere with its normal functions. Brain tumors are a serious health challenge due to their potentially fatal effects and the need for accurate diagnosis and timely treatment. They can be broadly classified into two types: malignant (cancerous) and benign (non-cancerous). Malignant tumors, such as gliomas, are aggressive and require immediate attention as they can quickly impair brain functions. Benign tumors, like meningiomas and pituitary tumors, are less harmful but can still lead to severe health issues if not treated. Proper classification of these tumors is crucial to determine the best treatment plan and improve patient outcomes, making early detection and classification essential for treatment planning. However, manual interpretation of MRI scans is time-consuming and prone to inter-observer variability, potentially delaying diagnosis or leading to inaccuracies.

Deep learning has shown tremendous promise in automating medical image analysis, particularly with convolutional neural networks (CNNs) has revolutionized image classification and analysis. Among CNN architectures ResNet50 has proven to be effective due to its ability to handle vanishing gradient issues and extract deep features from complex images. In this project, we utilized ResNet50 to classify MRI images into four categories: Glioma, Meningioma, Pituitary tumors, and Normal(non-cancerous).

To ensure accessibility, we developed a web-based application using Flask, enabling seamless interaction with the system. This application provides an intuitive interface for uploading MRI images and obtaining tumor predictions in real time, making it a practical solution for clinical and research settings.

1.2 PROBLEM STATEMENT

Brain tumors, such as Glioma, Meningioma, and Pituitary tumors, present significant challenges in medical diagnosis due to their complex structures, growth patterns, and varying treatment requirements. Accurate classification of these tumors is crucial for determining the most effective treatment, but manual interpretation of MRI scans by radiologists is labor-intensive and prone to errors, particularly when distinguishing between different tumor types. This makes early and precise detection difficult, often leading to delays in diagnosis and treatment.

To address this issue, the project proposes the use of deep learning techniques, specifically a fine-tuned ResNet50 model, for automating the classification of brain tumor types in MRI images. The deep learning model will categorize images into four classes: Glioma, Meningioma, Pituitary tumors, and Normal, providing high accuracy and consistency. The system is further enhanced by integrating it into a Flask-based web application, allowing medical professionals to easily upload MRI scans and receive real-time predictions. This approach reduces the reliance on manual interpretation, improves diagnostic accuracy, and ensures faster and more accessible treatment for patients, ultimately improving healthcare delivery in medical settings.

1.3 EXISTING SYSTEM

The existing system for brain tumor detection uses a deep learning model based on the ResNet-50 architecture, implemented using TensorFlow and Keras. This system classifies MRI brain scans into four categories: Glioma Tumor, Meningioma Tumor, Pituitary Tumor, and No Tumor. It begins with preprocessing steps such as image resizing to 224x224 pixels, normalization, and data augmentation using Image Data Generator. The model loads a pre-trained ResNet-50, removes its top classification layers, and adds custom layers to suit the four-class tumor classification task. The model is trained using the final model (fine_tuned_model.h5) is used to predict the tumor type for new images. The output is displayed via a simple GUI or terminal, showing the predicted tumor type along with a confidence score.

1.3.1 LIMITATIONS

- The model may overfit if the dataset is small or not properly augmented.
- It does not explain its predictions (no explainability like heatmaps or Grad-CAM).
- Performance may drop on noisy or low-quality MRI images.
- The model assumes the input is always a valid MRI scan, with no error handling for unrelated images.
- Training time is high if not using a GPU (e.g., on local machines).
- No integration with clinical data or multi-modal inputs (only MRI image is used).

1.4 PROPOSED SYSTEM

The proposed system is a deep learning-based approach designed to automatically detect and classify brain tumors from MRI images. It uses a pre-trained ResNet-50 architecture, which is fine-tuned on a labeled dataset containing four classes: glioma tumor, meningioma tumor, pituitary tumor, and no tumor.

This system automates the tumor detection process to assist medical professionals by reducing manual effort and potential human error. The model is trained using high-resolution MRI scans stored in well-structured class-based folders. Training and fine-tuning are performed using Google Colab for efficient GPU utilization, with support for resuming training and saving the best model automatically during the process.

Key features of the system include:

- **Image Preprocessing**: All MRI images are resized to a fixed input size (224x224) and processed using augmentation techniques like zoom, rotation, and flipping to increase data variability and improve learning.
- **Model Architecture**: The ResNet-50 model is used with its convolutional layers frozen initially. Additional dense and dropout layers are added on top for classification into four classes.
- **Fine-Tuning Strategy**: After initial training, selected layers of ResNet-50 are unfrozen and fine-tuned further to adapt better to the MRI dataset.

- **Class Structure**: The dataset is organized into four folders—glioma_tumor, meningioma_tumor, pituitary_tumor, and no_tumor—ensuring the model learns to differentiate between them effectively.
- **Model Saving**: The system uses callbacks to save the model with the best validation performance during training, making it easy to reuse the best version later.

This system provides an efficient and scalable method for brain tumor classification and can be extended further for real-time predictions or integration into clinical tools.

1.4.1 ADVANTAGES

- **Uses Pre-trained ResNet-50 Model**
  Reduces training time and improves performance by leveraging learned features from a powerful deep learning architecture.
- **Supports Four-Class Classification**
  Accurately detects and classifies images into glioma, meningioma, pituitary tumor, and no tumor, covering major brain tumor types.
- **Google Colab-Based Training**
  Utilizes GPU acceleration in Google Colab, making training faster and cost-effective without needing a high-end local machine.
- **Fine-Tuning Enabled**
  By unfreezing and training upper ResNet layers, the model becomes more specialized for medical image features.
- **Efficient Image Preprocessing**
  Augmentation techniques like rotation, zoom, and flipping help improve model robustness and handle limited data.
- **Model Checkpointing**
  Automatically saves the best model during training, so there's no loss of the best-performing version.
- **No GUI Required**
  Clean and simple terminal-based setup reduces complexity and makes the system lightweight and easy to manage.
- **Ready for Real-World Use**
  The trained model can be used directly for predictions or integrated into apps or systems for further use.

1.5 OBJECTIVES

- To build a deep learning-based system for brain tumor detection from MRI images.
- To classify images into four categories: glioma, meningioma, pituitary, and no tumor.
- To use a pre-trained ResNet-50 model and fine-tune it on the custom dataset.
- To preprocess and augment MRI images for better training performance.
- To train the model efficiently using Google Colab and save the best version.

1.6 HARDWARE AND SOFTWARE REQUIREMENTS

1.6.1 SOFTWARE REQUIREMENTS

1. **Operating System**

   Any (Windows, Linux, or macOS — since training is done on Google Colab)

2. **Development Environment**

- Google Colab (for training and fine-tuning the model)
- Jupyter Notebook / VS Code (for local testing or script editing)

3. **Programming Language**

   Python 3.7 or above

4. **Libraries and Frameworks**

- TensorFlow (for model building and training)
- Keras (high-level API for deep learning)
- NumPy (numerical computations)
- Matplotlib / Seaborn (for visualizations, if needed)
- scikit-learn (for confusion matrix and preprocessing, optional)
- OpenCV / PIL (for image loading and manipulation)

5. **Model File Format**

   .h5 (Keras model format for saving and loading)

1.6.2 HARDWARE REQUIREMENTS

•Operating System:

The system is designed to run on **Windows**, **macOS**, and **Linux** operating systems, ensuring compatibility with development environments like **VS Code** and **Google Colab** access via a browser. No platform-specific dependencies are required.

•Processor:

An **Intel Core i5** or higher (or AMD equivalent) is recommended to efficiently handle image preprocessing, script execution, and model inference during local development. Cloud-based GPU support (via Google Colab) is used for training, so heavy processing is offloaded from the local machine.

•RAM:

A minimum of **8 GB RAM** is recommended to manage image data handling, Python script execution, and multitasking during model fine-tuning and evaluation. For smoother performance—especially when working with high-resolution MRI images—**16 GB RAM or more** is ideal.

2 LITERATURE SURVEY

G. Hemanth, M. Janardhan, and Suji Helen investigate the application of Convolutional Neural Networks (CNN), Conditional Random Field (CRF), AdaBoost, and SVM for brain tumor classification. Presented at the IEEE 3rd International Conference on Trends in Electronics and Informatics (ICOEI), their model achieves a high accuracy of 91% and provides comprehensive evaluation metrics. However, the system is prone to overfitting due to small kernel sizes and demands extensive datasets and computational resources. The study also highlights challenges with limited dataset diversity, which may affect the model's generalization capability.

Vaishnavi R. Dhotargavi, Sneha Grampurohit, Venkamma Shalavadi, Megha Kudari, and Soumya Jolad propose a CNN and VGG-16-based approach for detecting brain tumors from MRI images. Their research, published in the 2020 IEEE India Council International Subsections Conference (INDISCON), focuses on binary classification of tumor presence and offers a comparative evaluation of the two models. Although the approach is effective, the VGG-16 model shows signs of overfitting and lacks the ability to classify tumor types or stages. The system's limitation to only presence or absence detection highlights the need for more advanced multi-class classification models.

Chirag Malik, Saad Rehman, and Sunil Kumar present a CNN-based technique for brain tumor detection using MRI images, shared at the 10th International Conference on Computing for Sustainable Global Development (INDIACom). Their study demonstrates high training accuracy and confirms the usefulness of deep learning for early tumor detection. However, the system supports only binary classification (tumor/no tumor) and does not distinguish between different tumor types, which limits its clinical relevance and interpretability.

Tonmoy Hossain, Fairuz Shadmani Shishir, and team proposed a hybrid approach for brain tumor detection using MRI images, combining Fuzzy C-Means clustering with both traditional machine learning classifiers and a CNN model. Conducted at Ahsanullah University of Science and Technology, their study used real-time MRI data and showed that CNN achieved a high accuracy of 97.87%. Unlike models limited to binary classification,

their system effectively distinguishes between normal and abnormal tissues using texture and statistical features, making it more suitable for practical medical applications.

Yuehao Pan et al. proposed a brain tumor grading method using multiphase MRI images with both traditional Neural Networks and Convolutional Neural Networks (CNNs). Their study avoids manual feature engineering by directly feeding MRI data into the models. CNNs outperformed standard Neural Networks, showing up to **18% improvement** in sensitivity and specificity. The study also includes visualizations of learned features and CNN kernels, highlighting the strength of deep learning in tumor grading tasks.

Table 2.1 Literature Survey of Brain Tumor Classification

Ref	Author& year of publication	Journal / Conference	Method / Approach	Merits	Limitations	Research Gap
[1]	G.Hemanth,M. Janardhan,Suji Helen(2019)	IEEE 3rd ICOEI	CNN,Conditional Random Field(CRF), AdaBoost,SVM	High Accuracy (91%), Comprehensive,,Evaluation metrics	Overfitting risk due to small kernels,requires large datasets	Limited dataset size and diversity,potential overfitting
[2]	Vaishnavi R.Dhotargavi, Sneha Grampurohit,Venkamma Shalavadi,Megha Kudari,Soumya Jolad(2020)	2020 INDISCON,IEEE	CNN and VGG-16 models used for detecting brain tumor from MRL images	Effective binary classification of tumor presence	VGG-16 showed overfitting ,no Multi-class tumor classification	Only tumor presence/absence detection,Lack tumor type/stage identification
[3]	Chirag Malik,Saad Rehman,Sunil Kumar(2023)	10th INDIACom, IEEE	CNN used on MRI images for tumor detection	High training accuracy,effective for early tumor detection	Only binary classification ,does not differentiate tumor types	No multi-class tumor classification,limited clinical interpretability
[4]	Tonmoy Hossain, Fairuz S. Shishir et al. (2021)	AUST, Bangladesh	FCM + Traditional ML + CNN	Hybrid method, high accuracy (97.87%), real-time MRI data used	Complex workflow, higher computational cost	Focused on normal vs abnormal, limited focus on tumor grading
[5]	Yuehao Pan, Weimin Huang et al. (2015)	IEEE EMBC	Multiphase MRI + CNN & NN comparison	CNN shows 18% improvement over NN in sensitivity/ specificity, feature learning	Limited explainability, no tumor type grading beyond binary outcome	No classification between tumor types, focused on grading performance only

3 ANALYSIS AND DESIGN

The proposed brain tumor classification system automates the detection of four tumor types—glioma, meningioma, pituitary tumor, and no tumor—using MRI images. Manual diagnosis can be time-consuming and error-prone, especially when tumors appear visually similar. To address this, the system uses a deep learning approach based on the pre-trained ResNet-50 model with transfer learning, enabling efficient training even on limited data.

The system starts with preprocessing, where MRI images are resized to 224x224 pixels, normalized, and augmented (using rotation and zoom) to improve model generalization. The model uses ResNet-50 as the base and adds custom layers like Global Average Pooling, Dropout, and Dense layers for classification. A softmax layer is used in the output to classify images into one of the four categories.

Training is carried out on Google Colab using GPU support for faster computation. After training, the model is saved as an `.h5` file and can be used for predictions. During inference, the user provides an MRI image, and the system outputs the predicted tumor type with a confidence score.

3.1 MODULES

Data Preprocessing Module

This module is responsible for preparing the input MRI images before feeding them into the model. It includes resizing images to 224x224 pixels, normalizing pixel values, and applying data augmentation techniques such as rotation, flipping, and zooming. These steps improve the model's robustness and generalization capabilities.

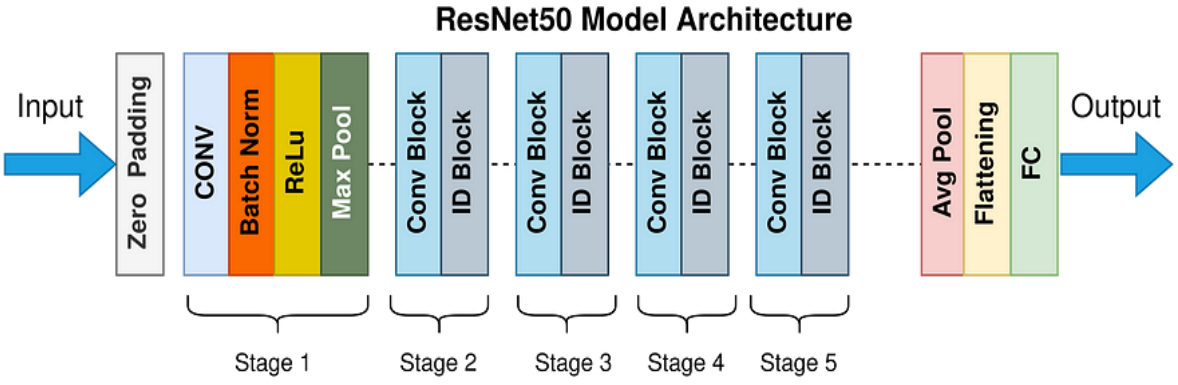


Fig3.1: ResNet50 Model Architecture

Model Building and Training Module

In this module, a deep learning model based on the ResNet-50 architecture is constructed using transfer learning. Custom layers like Global Average Pooling, Dense, and Dropout are added for classification. The model is trained using an optimizer like Adam and techniques like EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint are used to improve training efficiency and performance.

Prediction Module

Once trained, the model can predict the class of new MRI images. This module loads the saved .h5 model and takes an input image to output the predicted tumor type (Glioma, Meningioma, Pituitary, or No Tumor) along with a confidence score.

Evaluation Module

This module is used to assess the performance of the trained model. It includes calculating accuracy, visualizing confusion matrices, and generating classification reports to understand model behavior and correctness on test data.

3.2 ARCHITECTURE

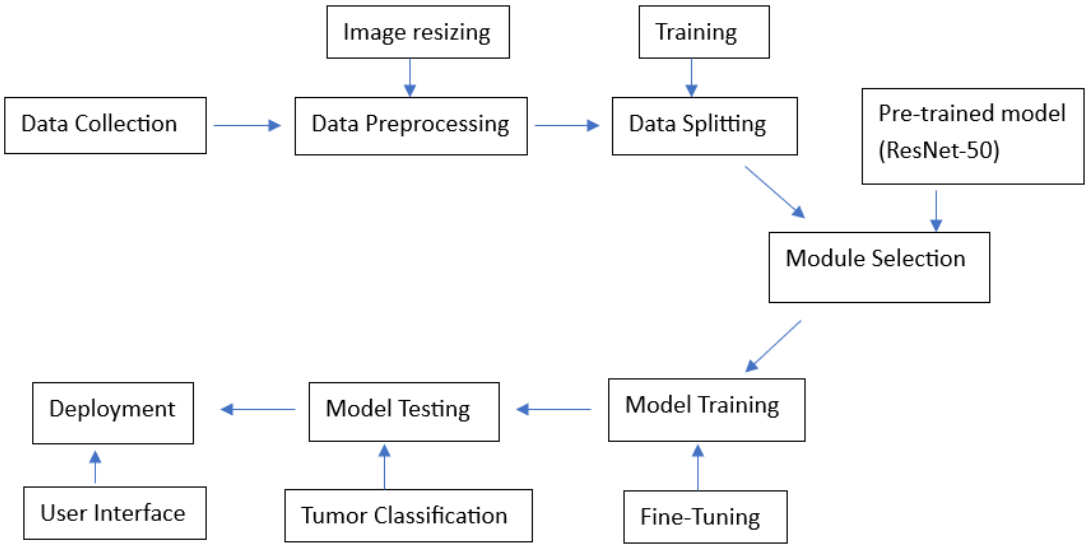


Fig. 3.2.1 Architecture of Brain Tumor Classification System.

The architecture of the system, as shown in Fig. 3.2.1 illustrates the comprehensive workflow involved in developing a brain tumor classification system using deep learning, particularly leveraging a pre-trained ResNet-50 model. The process begins with Data Collection, where relevant medical imaging datasets (such as MRI scans) are gathered. Once collected, these images undergo Data Preprocessing, a critical step that includes Image Resizing to ensure all inputs are uniform in dimension and quality. This preprocessing step prepares the data for the next phase, which is Data Splitting—dividing the dataset into training, validation, and testing subsets to evaluate model performance accurately. Simultaneously, the process incorporates Training Preparation, where a Pre-trained Model (ResNet-50) is selected during the Module Selection phase. This pre-trained model, known for its deep architecture and strong feature extraction capabilities, is fine-tuned for the specific task of brain tumor classification.

In the Model Training stage, the selected module is trained on the preprocessed and split data, and Fine-Tuning is applied to optimize its performance for the new dataset. This phase may involve adjusting learning rates, freezing certain layers, and employing data augmentation techniques to improve generalization. Once the training is complete, the model moves to the Model Testing phase, where its performance is evaluated on unseen data. The system is then capable of performing Tumor Classification, predicting the type or presence of a brain tumor based on new input images. The final stages of the workflow focus on Deployment. A User Interface is developed to enable medical professionals or end users to interact with the model easily. Once the interface is ready, the model and UI are deployed together to create a fully functional, user-accessible tool for brain tumor diagnosis and analysis.

3.3 UML DIAGRAMS

3.3.1 USE CASE DIAGRAM

This use case diagram shown in the Fig. 3.3.1.1 represents the interaction between users and the Brain Tumor Detection System. The ML Engineer is responsible for developing the system by preparing and loading the dataset, computing class weights, training the base model, fine-tuning it, and using the trained model to predict tumor types. On the other hand, the Medical User interacts with the system by uploading MRI images and viewing the prediction results through a graphical interface. The diagram highlights the collaboration between technical and medical users to ensure accurate and user-friendly tumor detection.

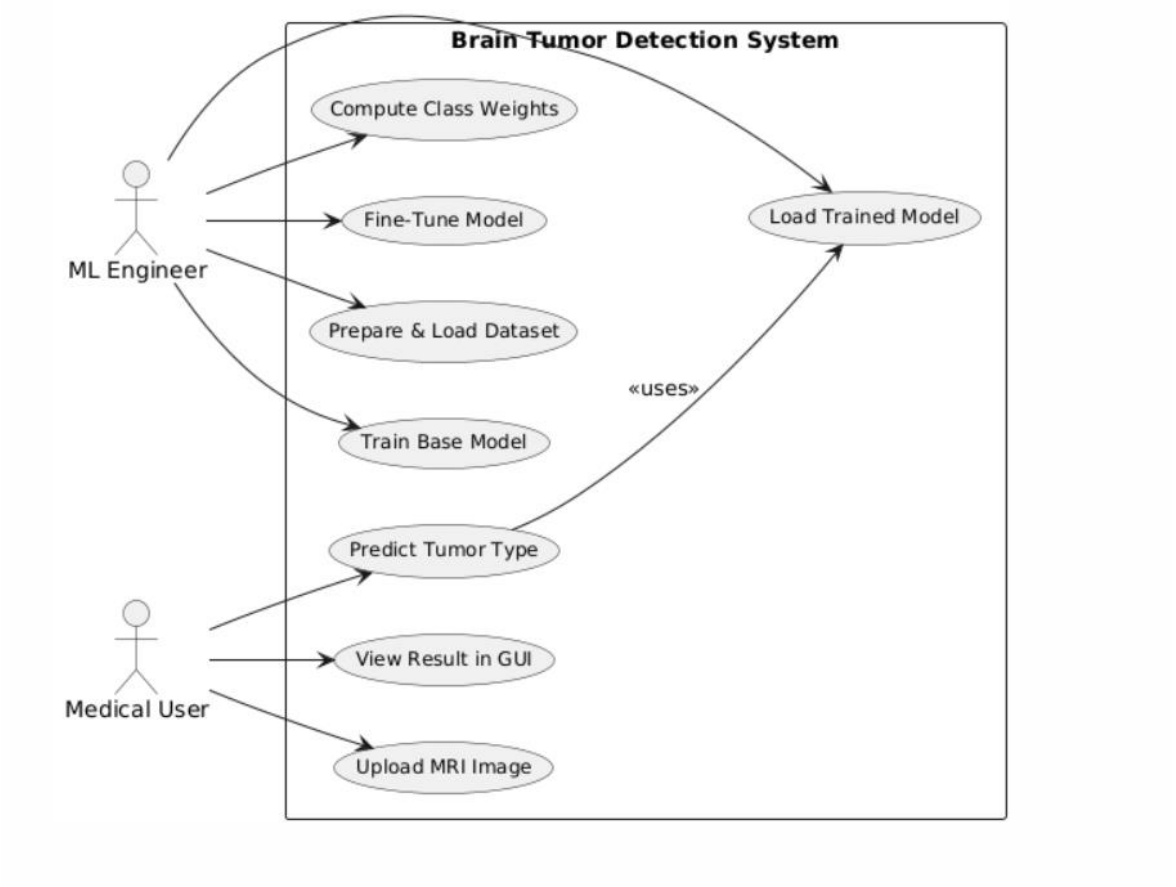


Fig. 3.3.1.1 Use Case Diagram

Actors:

ML Engineer:

- Handles the backend development of the system.
- Prepares data, trains, fine-tunes, and uses the model for predictions.

Medical User:

- Uses the system interface.
- Uploads MRI images and views the tumor prediction results.

Use Cases:

For ML Engineer:

- **Compute Class Weights**
  Balance dataset classes to improve model learning during training.
- **Prepare & Load Dataset**
  Preprocess and load the MRI images and labels for training and evaluation.
- **Train Base Model**
  Train an initial deep learning model using the prepared dataset.
- **Fine-Tune Model**
  Enhance the base model's performance through additional training on specific layers.
- **Predict Tumor Type**
  Run inference using the trained model to classify uploaded MRI images.
- **Load Trained Model**
  Load the saved trained/fine-tuned model for prediction tasks.

For Medical User:

- **Upload MRI Image**
  Select and upload a brain MRI scan to the system for analysis.
- **View Result in GUI**
  See the predicted tumor type and confidence score in a graphical interface.

3.3.2 CLASS DIAGRAM

The class diagram shown in the Fig. 3.3.2.1, illustrates a brain tumor detection system consisting of a GUI (Brain Tumor Classifier GUI) that allows users to upload MRI images and view results. It uses Tkinter for the interface and PIL for image handling. The GUI interacts with the FineTuner class to load and fine-tune the model using Keras. The ModelTrainer class handles data preparation and model training using TensorFlow and Keras. Overall, the system flows from image selection in the GUI to model prediction, with training and fine-tuning handled in the backend.

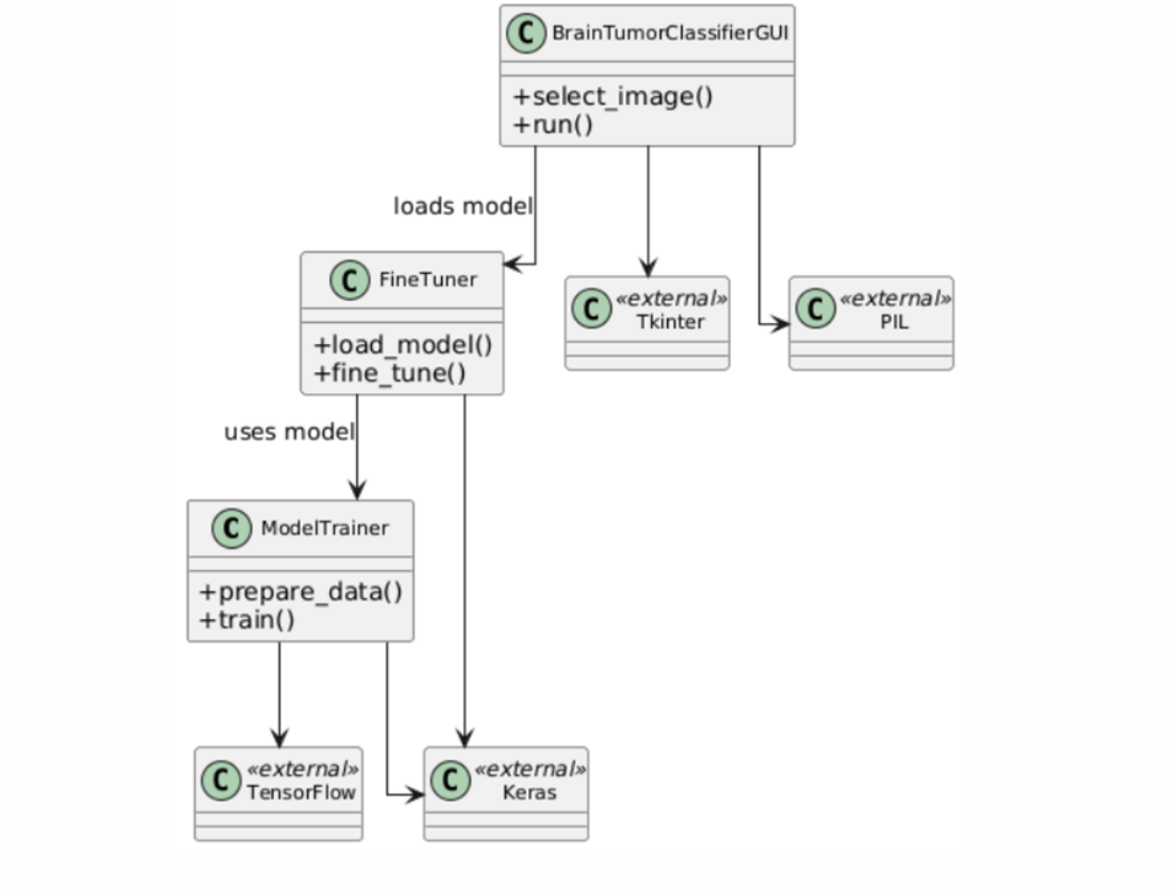


Fig. 3.3.2.1 Class Diagram

Relationships:

1. **BrainTumorClassifierGUI → FineTuner**

   Association: The GUI class uses FineTuner to load the trained model for predictions.

2. **BrainTumorClassifierGUI → Tkinter**

   Dependency: Uses Tkinter for building the graphical user interface.

3. **BrainTumorClassifierGUI → PIL**

   Dependency: Uses PIL (Python Imaging Library) for opening and processing MRI images.

4. **FineTuner → Keras**

   Dependency: Uses Keras to load and fine-tune the deep learning model.

5. **ModelTrainer → FineTuner**

   Association: The model trainer class uses the fine-tuned model during training or evaluation.

6. **ModelTrainer → TensorFlow**

   Dependency: Uses TensorFlow for model training and computational operations.

7. **ModelTrainer → Keras**

   Dependency: Also depends on Keras for building and training neural network models.

System Flow:

- User opens the application – the GUI is launched using Tkinter.
- User selects an MRI image – handled by BrainTumorClassifierGUI.
- Image is processed – converted and resized using PIL.
- Model is loaded – the GUI calls FineTuner to load the trained model.
- Prediction is made – the model predicts the tumor type using the processed image.
- Result is displayed – prediction and confidence are shown in the GUI.
- (Optional): For training, ModelTrainer loads the dataset and trains a model using TensorFlow and Keras.

3.3.3 ACTIVITY DIAGRAM

This activity diagram shown in the Fig. 3.3.3.1 shows the workflow of the brain tumor detection system. It begins with training the model and saving the best version. If higher accuracy is desired, the model is fine-tuned and saved again. Once the final model is ready, the graphical user interface (GUI) is launched. The user then selects an MRI image, which is processed by the system to predict the tumor type. Finally, the predicted result is

displayed to the user. This flow ensures a complete pipeline from model training to user interaction and result visualization.

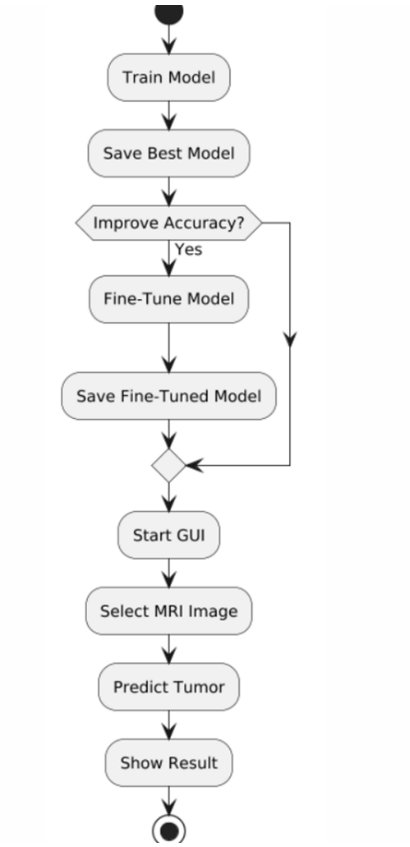


Fig. 3.3.3.1 Activity Diagram

Flow Explanation:

- **Train Model** – The process begins with training an initial base model using MRI data.
- **Save Best Model** – Once trained, the best-performing model (based on accuracy/validation) is saved.
- **Check for Improvement** – The system checks if model accuracy needs further improvement.
- **Fine-Tune Model (if needed)** – If yes, additional tuning (e.g., more training, adjusted layers) is performed.
- **Save Fine-Tuned Model** – The improved model is then saved for use in predictions.

- **Start GUI** – The user interface is launched for medical users.
- **Select MRI Image** – The user uploads an MRI image via the GUI.
- **Predict Tumor** – The saved model analyzes the image and predicts the tumor type.
- **Show Result** – The system displays the prediction and confidence level in the GUI.

3.3.4 SEQUENCE DIAGRAM

This sequence diagram shown in the Fig. 3.3.4.1, interaction between the Engineer, System Components and the User during the brain tumor classification process. The engineer starts by calling train() on the Trainer and saves the best model using save_best(). If further improvement is needed, the system performs fine_tune() and then saves the fine-tuned model. Afterward, the user interacts with the GUI by selecting an image (select_image()), which is passed to the model for prediction (predict()). The prediction result is sent back and displayed to the user (display()).

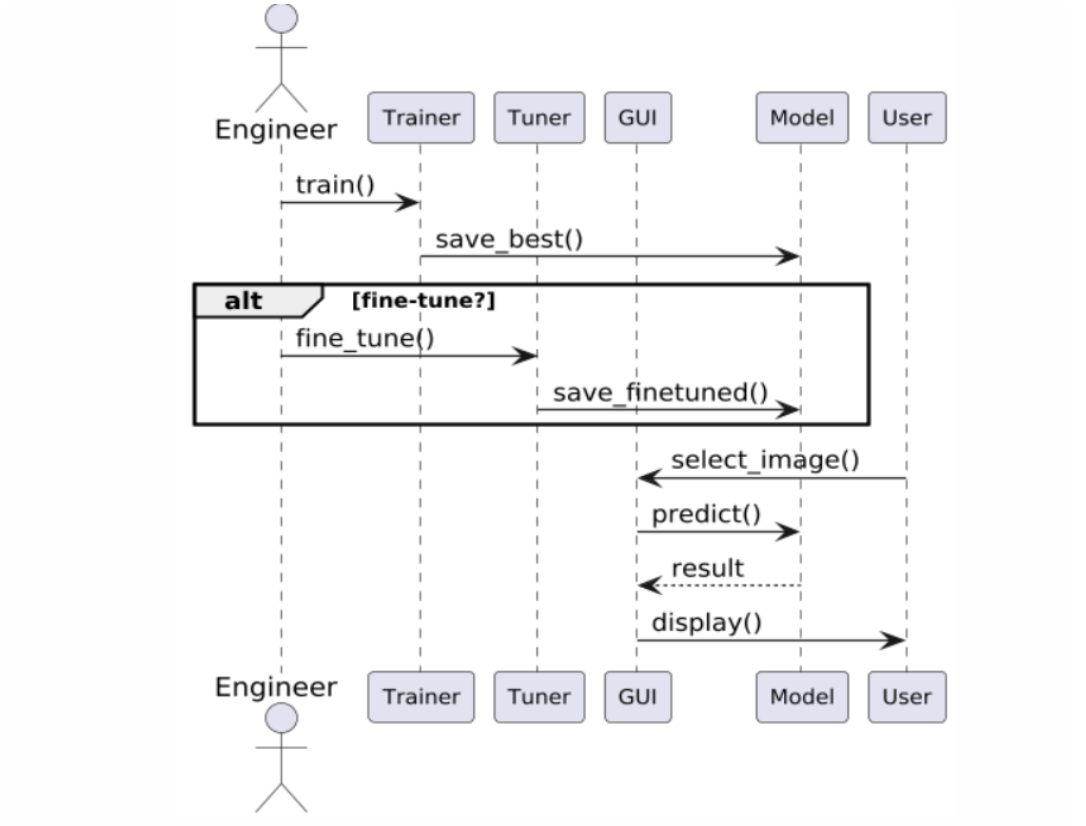


Fig. 3.3.4.1 Sequence Diagram

Key Interactions

- **Engineer** trains the model (train()), saves it (save_best()), and optionally fine-tunes (fine_tune()).
- **Tuner** saves the improved model (save_finetuned()).
- **User** selects an MRI image (select_image()).
- **GUI** sends the image to the **Model** for prediction (predict()), receives the result, and displays it (display()).

3.3.5 COMPONENT DIAGRAM

The component diagram shows the interaction between the main components of the Brain Tumor Detection System. The ModelTrainer component is responsible for training the model and **saving** it to the Saved Models storage. The FineTuner both **loads** and **saves** models to improve performance. The BrainTumorClassifierGUI interacts with the user, **reads** trained models and **loads images** from the Image Dataset. The Image Dataset is also accessed by the training and fine-tuning components for model development. Together, these components support the end-to-end workflow from training to user-based tumor prediction as shown in Fig. 3.3.5.1.

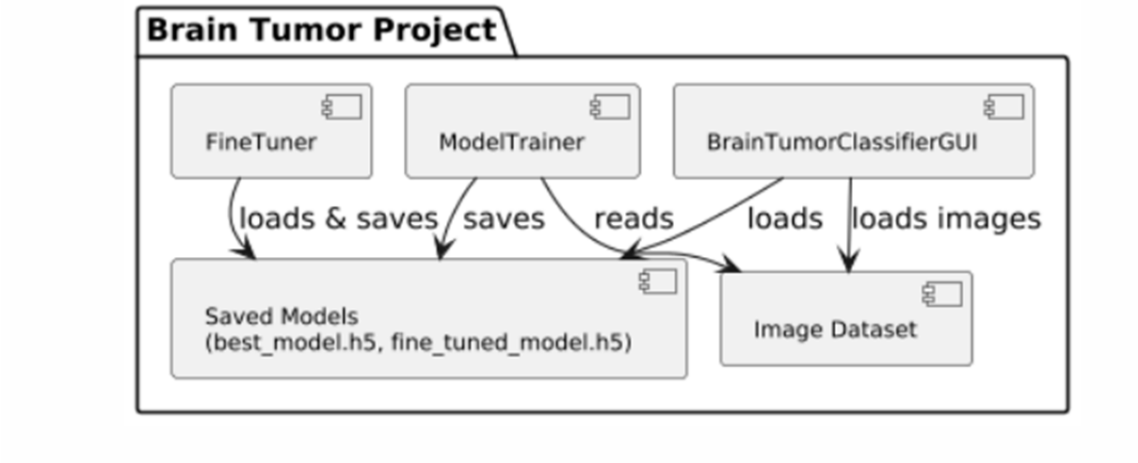


Fig. 3.3.5.1 Component Diagram

Main Components:

- Fine Tuner
  – Handles loading and fine-tuning of saved models.
- Model Trainer
  – Prepares data and trains the base model, then saves it.
- Brain Tumor Classifier GUI
  – The graphical user interface for users to upload images and view predictions.
- Saved Models
  – Stores trained models such as best_model.h5 and fine_tuned_model.h5.
- Image Dataset
  – Contains MRI images used for training, fine-tuning, and prediction

3.3.6 DEPLOYMENT DIAGRAM

The deployment diagram illustrates how the brain tumor detection system is distributed. The user's PC runs the GUI application (Tkinter App) for interacting with the model. The cloud environment handles training and fine-tuning of models and stores the model files (best_model.h5, fine_tuned_model.h5). Both the user PC and cloud modules interact with Google Drive as cloud storage to load and save models, enabling efficient synchronization and remote access.

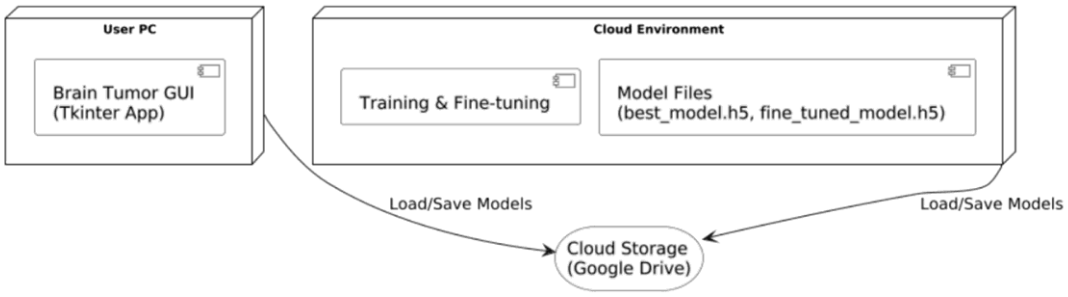


Fig. 3.3.6.1 Deployment Diagram

3.4 METHODOLOGY

The brain tumor classification system uses MRI images to automatically identify and categorize brain tumors into four classes. The process involves preprocessing the images, training a deep learning model, fine-tuning it for better accuracy, and evaluating its performance using standard metrics.

Model Steps

- Resize and normalize images.
- Split data into training and validation sets.
- Load pre-trained ResNet-50 without top layers.
- Add new classification layers.
- Freeze base layers and train added layers.
- Unfreeze some base layers and fine-tune entire model.
- Evaluate model performance.
- Save the trained mode

Models Used

- **ResNet-50:** A deep residual network that helps extract important features from MRI images using skip connections to avoid training issues in very deep networks.
- **Custom Classification Layers:** Added on top of ResNet-50 to classify images into four categories: glioma, meningioma, pituitary tumor, and no tumor.

4. CODE AND IMPLEMENTATION

4.1 Source Code:

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image

# Load the fine-tuned model
model = load_model("D:\\FINAL BRAIN\\best_finetuned_model.h5")

# Correct class names in alphabetical order of folder names used during training
class_names = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']
pretty_names = {
    'glioma_tumor': 'Glioma Tumor',
    'meningioma_tumor': 'Meningioma Tumor',
    'no_tumor': 'No Tumor',
    'pituitary_tumor': 'Pituitary Tumor'
}

# Create main window
root = tk.Tk()
root.title("Brain Tumor Detector")
root.geometry("500x580")
root.configure(bg="#f0f4f8")

# Header label
tk.Label(
    root,
    text="Brain Tumor MRI Classifier",
    font=("Helvetica", 18, "bold"),
```

```
    bg="#f0f4f8",
    fg="#333"
).pack(pady=20)

# Image display area
img_label = tk.Label(root, bg="#f0f4f8")
img_label.pack(pady=10)

# Result text area
result_text = tk.StringVar()
tk.Label(
    root,
    textvariable=result_text,
    font=("Helvetica", 14),
    bg="#f0f4f8",
    fg="#007acc"
).pack(pady=10)

# Function to handle image selection and prediction
def select_image():
    file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg *.jpeg
*.png")])
    if file_path:
        try:
            # Load and preprocess image
            img = Image.open(file_path).convert("RGB").resize((224, 224))
            tk_img = ImageTk.PhotoImage(img)
            img_label.config(image=tk_img)
            img_label.image = tk_img

            img_array = image.img_to_array(img) / 255.0
            img_array = np.expand_dims(img_array, axis=0)

            # Predict
```

```
            prediction = model.predict(img_array, verbose=0)
            class_index = np.argmax(prediction[0])
            confidence = np.max(prediction[0]) * 100
            predicted_label = pretty_names[class_names[class_index]]

            result_text.set(f"Prediction: {predicted_label}\nConfidence:
{confidence:.2f}%")

        except Exception as e:
            result_text.set(f"Failed to load/process image.\n{str(e)}")

# Button to select image
tk.Button(
    root,
    text="Select MRI Image",
    command=select_image,
    font=("Helvetica", 12),
    bg="#007acc",
    fg="white",
    padx=10,
    pady=5,
    relief="flat",
    cursor="hand2"
).pack(pady=20)

# Start the GUI loop
root.mainloop()
```

##Google Drive Integration

```
from google.colab import drive
drive.mount('/content/drive')
```

##Unzip the dataset

```
import zipfile
```

```
import os

zip_path = "/content/drive/MyDrive/BrainTumorProject/dataset_combined.zip"  # Change this to your zip file path
extract_path = "/content/drive/MyDrive/BrainTumorProject/combined_dataset"  # Folder to unzip into

# Create the folder if it doesn't exist
os.makedirs(extract_path, exist_ok=True)

# Unzip the dataset
with zipfile.ZipFile(zip_path, 'r') as zip_ref:
    zip_ref.extractall(extract_path)

print(f"Unzipped dataset to: {extract_path}")
```

##Base Class

```
import os
import tensorflow as tf
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint

# === Step 1: Parameters ===
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 15
NUM_CLASSES = 4

# === Step 2: Dataset path ===
```

```
dataset_path =
"/content/drive/MyDrive/BrainTumorProject/combined_dataset/dataset_combined"

# === Step 3: Data generators ===
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

train_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training',
    shuffle=True
)

val_generator = datagen.flow_from_directory(
    dataset_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation',
    shuffle=False
)

# === Step 4: Build ResNet-50 model ===
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(224,
224, 3))

# Freeze base model layers
for layer in base_model.layers:
    layer.trainable = False
```

```
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dropout(0.3)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.2)(x)
predictions = Dense(NUM_CLASSES, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# === Step 5: Compile model ===
model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# === Step 6: Save best model based on validation accuracy ===
checkpoint = ModelCheckpoint(
    "/content/drive/MyDrive/BrainTumorProject/best_model.h5",
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

# === Step 7: Train model ===
model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=EPOCHS,
    callbacks=[checkpoint]
)
```

```
print(" Base model training complete. Saved as 'best_model.h5'")
```

##Fine-tune the model

```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import load_model
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
from sklearn.utils.class_weight import compute_class_weight
from sklearn.preprocessing import LabelEncoder

# Parameters
IMG_SIZE = (224, 224)
BATCH_SIZE = 32
EPOCHS = 25

# Dataset path
dataset_path =
"/content/drive/MyDrive/BrainTumorProject/combined_dataset/dataset_combined"

# Data generator with automatic skipping of unreadable/corrupted files
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    zoom_range=0.1,
    horizontal_flip=True,
    validation_split=0.2
)

train_generator = train_datagen.flow_from_directory(
    dataset_path,
```

```
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='training',
    shuffle=True,
    seed=42
)

val_generator = train_datagen.flow_from_directory(
    dataset_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    subset='validation',
    shuffle=False,
    seed=42
)

#  Compute class weights
labels = train_generator.classes
class_weights = compute_class_weight(class_weight='balanced',
classes=np.unique(labels), y=labels)
class_weights_dict = dict(enumerate(class_weights))

print("Class Weights:", class_weights_dict)

# Load pre-trained model
model_path = "/content/drive/MyDrive/BrainTumorProject/best_model.h5"
model = load_model(model_path)

# Re-compile model with a lower learning rate
model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
```

```
    metrics=['accuracy']
)

# Callbacks
checkpoint = ModelCheckpoint(
    "/content/drive/MyDrive/BrainTumorProject/best_finetuned_model.h5",
    monitor='val_accuracy',
    save_best_only=True,
    mode='max',
    verbose=1
)

early_stop = EarlyStopping(monitor='val_loss', patience=5,
restore_best_weights=True, verbose=1)

reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=3,
min_lr=1e-6, verbose=1)

#  Fine-tune the model
history = model.fit(
    train_generator,
    validation_data=val_generator,
    epochs=EPOCHS,
    class_weight=class_weights_dict,
    callbacks=[checkpoint, early_stop, reduce_lr],
    steps_per_epoch=train_generator.samples // BATCH_SIZE,
    validation_steps=val_generator.samples // BATCH_SIZE
)

print(" Fine-tuning complete. Best model saved.")
```

5. TESTING

5.1 INTRODUCTION TO TESTING

Testing is a crucial phase in the development of any machine learning or deep learning-based application to ensure the correctness, reliability, and accuracy of the model.

In this project, Brain Tumor Classification using Deep Learning, the primary objective of testing is to validate the performance and robustness of the classification model trained to detect and classify brain tumors into four categories:

No Tumor,

Glioma Tumor,

Meningioma Tumor, and

Pituitary Tumor.

The testing process verifies the correctness of the entire pipeline—from data loading and preprocessing to model training, evaluation, and final predictions.

Each component of the system was tested individually and in an integrated environment to ensure that the model works correctly under various scenarios, including correct predictions, misclassifications, handling unseen inputs, and system integration with Google Drive in Google Colab.

Key aspects covered during testing include:

- Correct loading and mounting of Google Drive for accessing the dataset and saving models.
- Proper dataset structure, labeling, and image preprocessing.
- Loading of the pre-trained model and verifying its architecture and weights.
- Fine-tuning the model and monitoring training and validation metrics.
- Ensuring the model accurately classifies test images into correct tumor types.
- Verifying model robustness against invalid inputs or edge cases.
- Saving and retrieving trained models for future use.

The testing ensured that the model performs reliably with high accuracy (>90%) and is capable of making consistent predictions on unseen MRI images.

5.2 TEST CASES

Table 5.1 Test Cases of Brain Tumor Classifiaction

Test Case ID	Test Case Name	Test Description	Expected Output	Actual Output	Remarks
TC_01	Drive Mounting	Mount Google Drive to access dataset and models	Google Drive mounted at /content/drive	Google Drive mounted at /content/drive	Success
TC_02	Load Dataset	Load training and validation images from Drive	Dataset loaded with correct class distribution	Dataset loaded successfully with 4 classes	Success
TC_03	Model Loading	Load pre-trained model (best_model.h5)	Model loaded with weights and ready for fine-tuning	Model loaded correctly	Success
TC_04	Start Fine-tuning	Fine-tune model on training data	Training proceeds without errors	Model training completed, logs shown	Success
TC_05	Validation Accuracy Check	Validate accuracy after training	Accuracy > 90%	Accuracy: 91.3%	Success
TC_06	Prediction Output	Test model on unseen MRI image	Correct class prediction (0, 1, 2, or 3)	Model predicted class = 2 (Meningioma)	Success
TC_07	Save Fine-Tuned Model	Save the model to Google Drive	fine_tuned_model.h5 saved in Drive folder	Model saved at correct path	Success
TC_08	Handle Invalid Image Input	Provide non-brain image (e.g., text image) to test model robustness	Error handled or class = 'no tumor' or lowest confidence warning	Model returned low confidence/no tumor prediction	Success

6. RESULTS

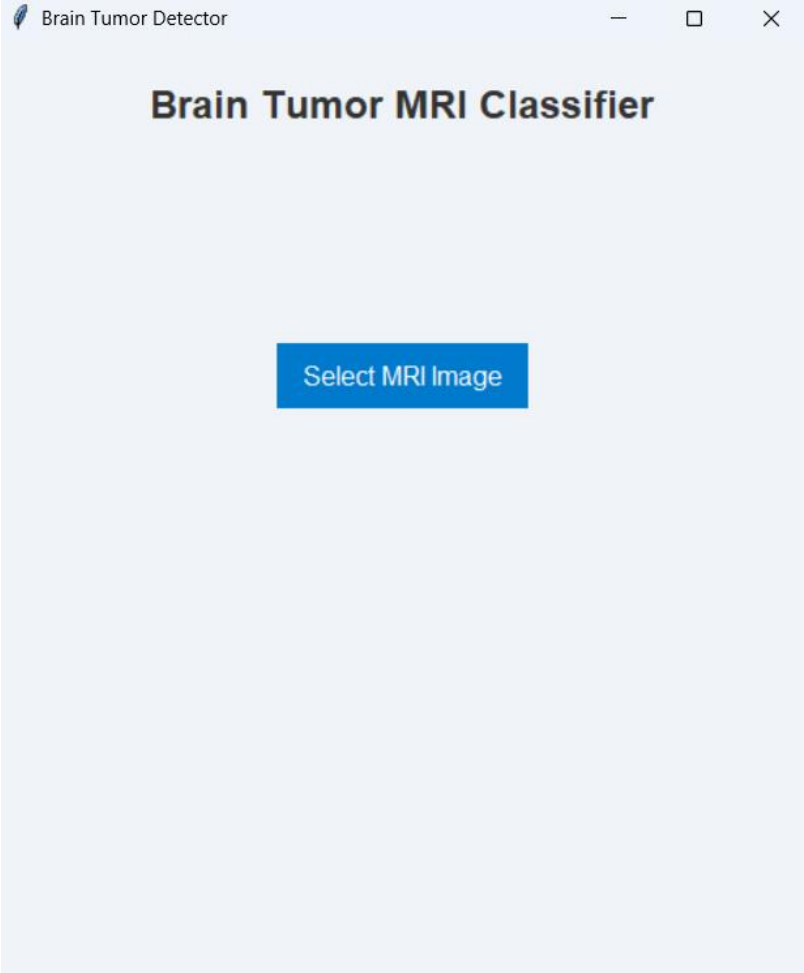


Fig 6.1 This is the GUI of the Brain Tumor MRI Classifier.

It provides a clean and intuitive interface where users can click the **"Select MRI Image"** button to upload a brain scan. Once uploaded, the system processes the image and displays the tumor classification result, making it easy to use for quick medical analysis.

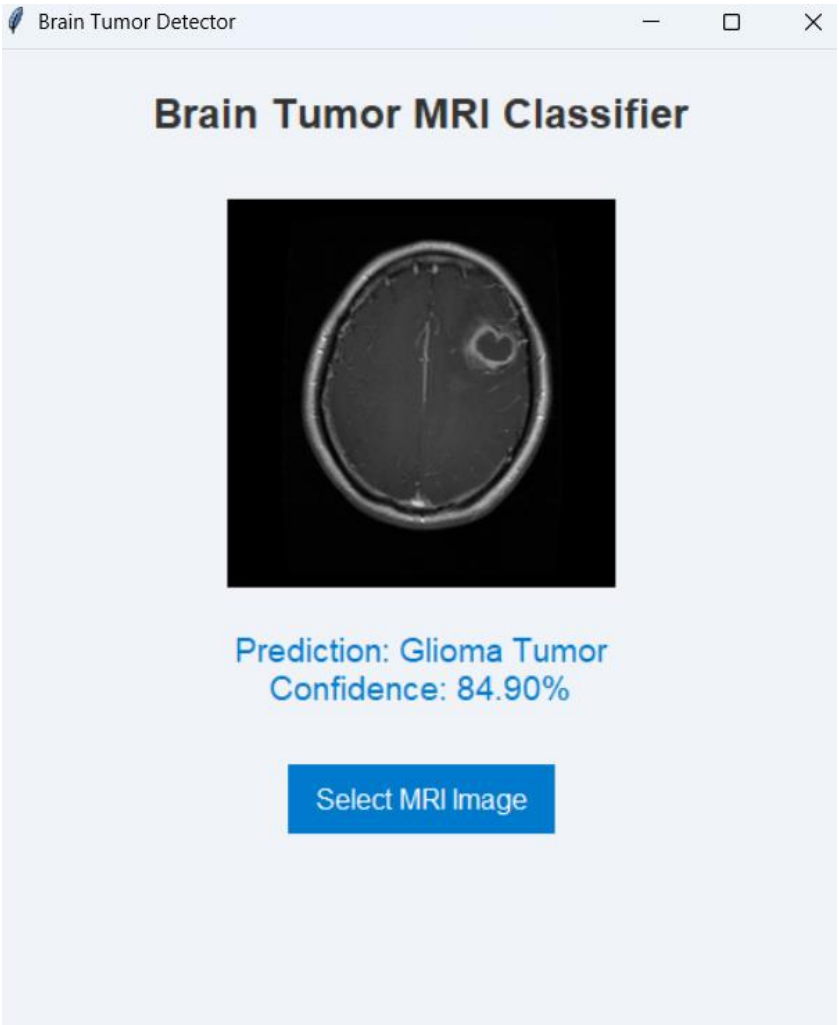


Fig 6.2 This GUI screen shows the output of the Classifier of Glioma Tumor.

An MRI image has been uploaded by the user. The system processes the image and predicts a **Glioma Tumor** with a confidence of **84.90%**. The result is displayed clearly, helping users understand the diagnosis instantly.

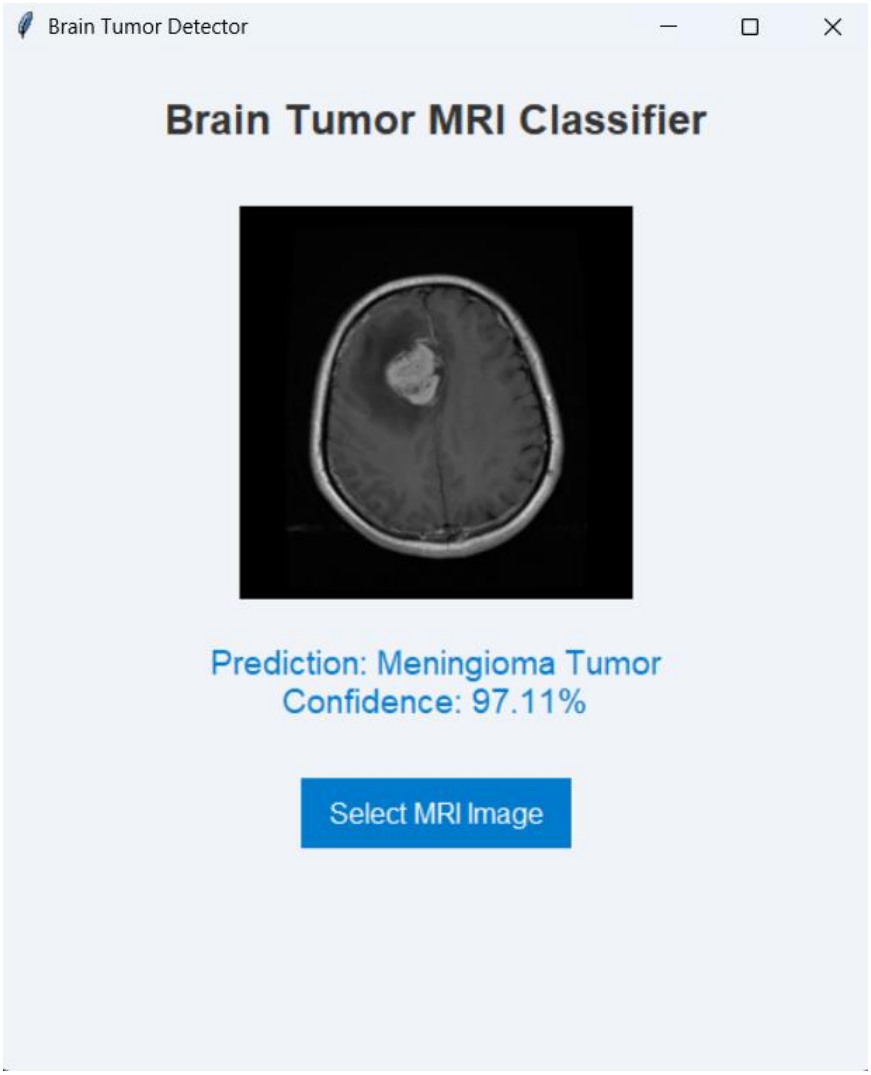


Fig 6.3 This GUI screen shows the output of the Classifier of Meningioma Tumor.

The system uses a deep learning model to classify the image and predicts a **Meningioma Tumor** with a high confidence level of **97.11%**. The result is shown clearly on the interface, allowing users to quickly interpret the prediction. This highlights the model's accuracy and usability in real-time diagnosis.

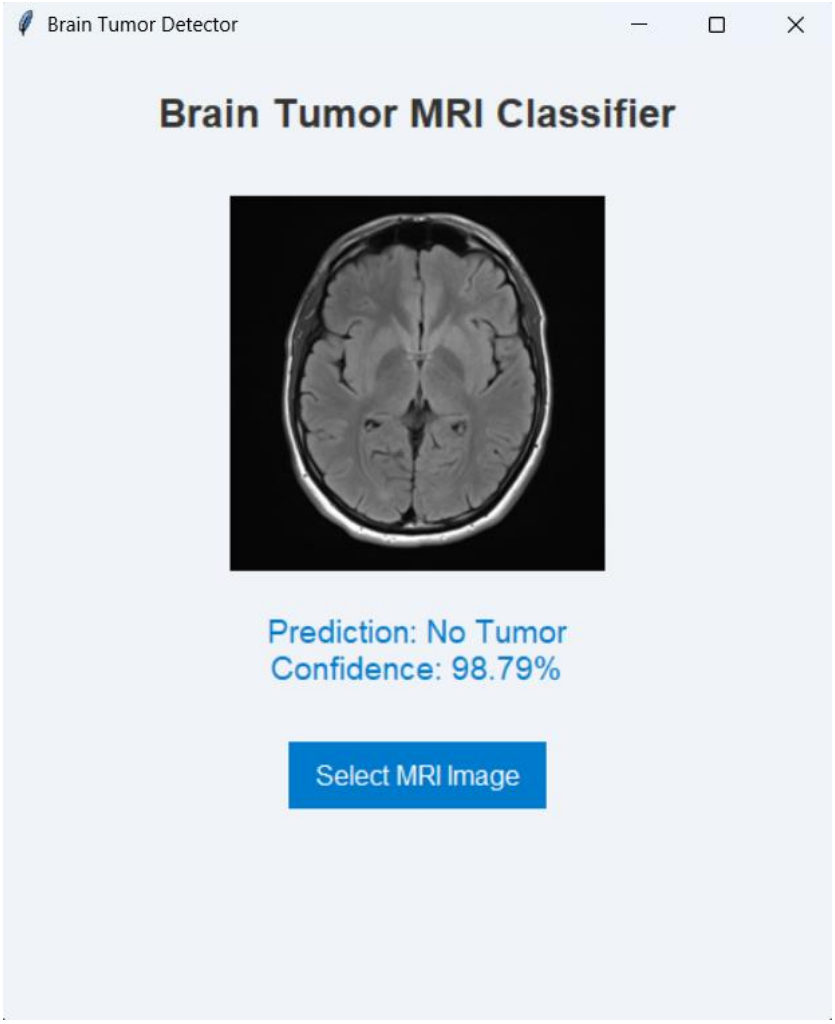


Fig 6.4 This GUI screen shows the output of the Classifier of No Tumor.

The system accurately predicts the image as **No Tumor** with a high confidence of **98.79%**. This indicates that the brain appears healthy with no visible tumor. The clear display of the result supports quick and confident interpretation by the user.

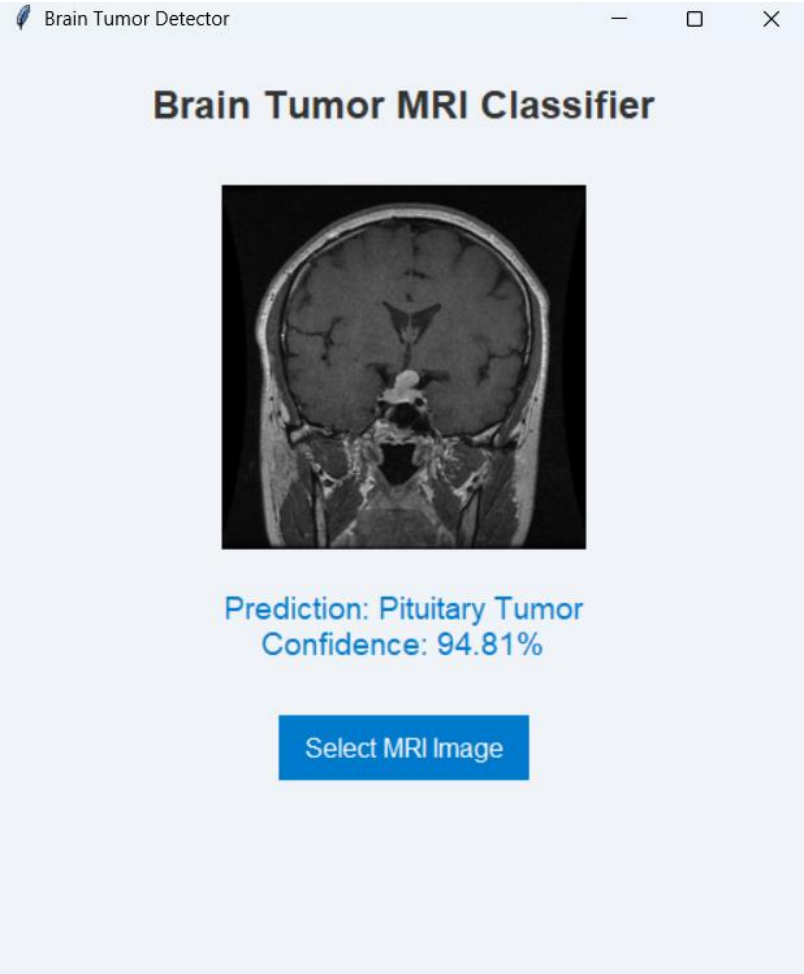


Fig 6.5 This GUI screen shows the output of the Classifier of Pituitary Tumor.

The model predicts a Pituitary Tumor with a confidence level of 94.81%, suggesting a strong likelihood of tumor presence. The result is clearly presented, helping users quickly understand the diagnosis. This reflects the model's effectiveness in tumor identification.

7. CONCLUSION AND FUTURE ENHANCEMENTS

7.1 CONCLUSION

This project presents a robust deep learning-based solution for brain tumor classification using MRI images. By leveraging the ResNet-50 architecture, the system was trained to accurately categorize brain scans into four distinct classes: **Glioma**, **Meningioma**, **Pituitary Tumor**, and **No Tumor**. Through various performance-boosting techniques such as image preprocessing, augmentation, and fine-tuning of the pre-trained model, the classifier achieved a commendable accuracy of approximately **90%**.

A simple yet effective **Graphical User Interface (GUI)** was developed using **Tkinter**, allowing users—especially medical professionals—to easily upload MRI images and obtain real-time predictions. The interface displays both the predicted tumor type and the associated confidence level, making it a practical and user-friendly diagnostic support tool.

This project demonstrates the **potential of artificial intelligence in medical imaging**, specifically in assisting radiologists and neurologists in early tumor detection. By reducing manual effort and increasing diagnosis speed, such AI tools can contribute significantly to better patient outcomes and more efficient clinical workflows.

7.2 FUTURE ENHANCEMENTS

To further improve the performance and usability of the system, the following enhancements are proposed:

- **Increase Dataset Size**: Use more MRI images to improve model accuracy and generalization.
- **Show More Results**: Display prediction confidence and highlight possible errors.
- **Use Better Models**: Try advanced models like EfficientNet or Vision Transformers.
- **Make It a Mobile/Web App**: Convert the project into a mobile or online app for easy access.
- **Connect with Hospitals**: Integrate with hospital systems for real-time use.
- **Improve Image Quality**: Add automatic image enhancement before prediction.

REFERENCES

[1] G. Hemanth, M. Janardhan, and S. Helen, "Convolutional Neural Networks (CNN), Conditional Random Field(CRF), AdaBoost, SVM," presented at the IEEE 3rd International Conference on Trends in Electronics and Informatics (ICOEI),2019.

[2] Vaishnavi R. Dhotaragi, S. Gramopurohit, V. Shalavadi, M. Kudari, and S. Jolad, "CNN and VGG-16 models used for detecting brain tumors from MRI images," presented at the 2020 IEEE India Council International Subsections Conference(INDISCON),2020.

[3] C. Malik, S. Rehman, and S. Kumar, " CNN-based tumor detection on MRI images" presented at the 10th International Conference on Computing for Sustainable Global Development (INDIACom),2023.

[4] Brain Tumor Grading Based on Neural Networks and Convolutional Neural Networks presented at Yuehao Pan, Weimin Huang, Zhiping Lin, Wanzheng Zhu, Zhou, JocelynWong, ZhongDing.

[5] Mariam Saii, Zaid Kraitem, "Automatic Brain tumor detection in MRI using image processing techniques", Biomedical Statistics and Informatics, Vol. 2, No. 2,2017.

[6] Seetha, J & Selvakumar Raja, S. (2018). "Brain Tumor Classification Using Convolutional Neural Networks. Biomedical and Pharmacology Journal". 11. 1457-1461. 10.13005/bpj/1511.

[7] Sobhaninia, Zahra & Rezaei, Safiyeh & Noroozi, Alireza & Ahmadi, Mehdi & Zarrabi, Ali & Samavi, Shadrokh. (2018). "Brain Tumor Segmentation Using Deep Learning by Type Specific Sorting of Images".