CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Brain tumors are a critical medical condition that requires early and accurate detection for effective treatment. This project focuses on developing an automated brain tumor detection system using MRI scans to assist radiologists in diagnosing and classifying tumors efficiently. The dataset comprises MRI images categorized into different tumor types, including Glioma, Meningioma, and Pituitary Tumor, along with normal brain scans. Brain tumor detection through MRI analysis is crucial, as timely diagnosis can significantly improve treatment outcomes and survival rates. Traditional diagnostic methods often rely on manual assessment by radiologists, which can be time-consuming and prone to human error. Automating the detection process can enhance accuracy, consistency, and efficiency in medical diagnosis. By utilizing medical imaging data, this project aims to develop a reliable system that improves sensitivity and specificity in tumor identification, reducing the chances of misdiagnosis

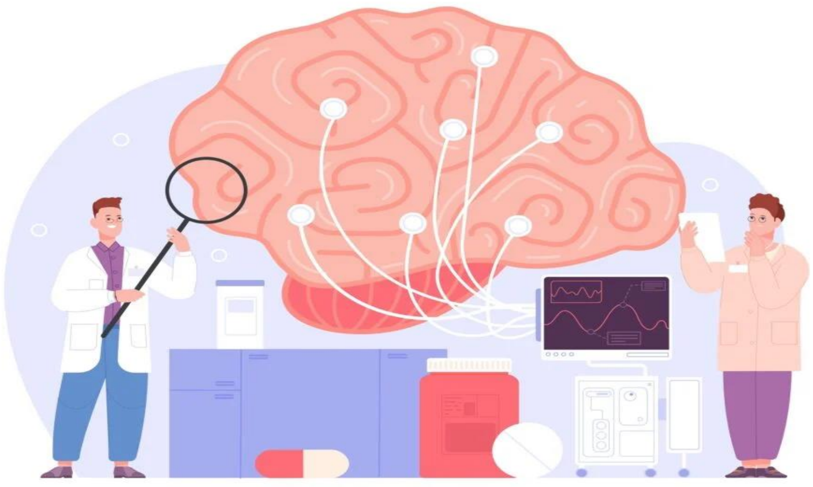


Figure 1.1 Overview Of Brain Tumor

The implementation of this system can aid healthcare professionals by providing an additional diagnostic tool, ultimately optimizing patient care. The project also emphasizes the importance of integrating AI-based solutions into healthcare, making brain tumor detection more accessible and efficient. The ability to identify tumors accurately at an early stage can lead to timely medical intervention, improving patient outcomes and reducing the burden on healthcare systems. Furthermore, this approach can be extended to other medical imaging applications, revolutionizing diagnostic capabilities in the medical field. By ensuring reliable and effective brain tumor detection, this project has the potential to contribute significantly to the advancement of medical diagnostics and patient care.

This project also explores the potential of deep learning techniques to automate the analysis of MRI scans, reducing the reliance on manual input and increasing the speed of diagnosis. By leveraging convolutional neural networks (CNNs) and pre-trained models, the system can efficiently process complex imaging data, identifying tumor regions with high accuracy. The integration of AI models can help standardize the detection process, reducing variability in diagnostic results. Additionally, continuous improvements through model training on larger datasets will ensure that the system remains robust and adaptable to new tumor types and imaging technologies. This approach fosters greater collaboration between healthcare professionals and AI tools, leading to more informed decision-making. As technology advances, the scalability of such automated systems can lead to broader implementation in healthcare facilities worldwide.

1.2 INTRODUCTION TO MACHINE LEARNING

Machine learning is a branch of artificial intelligence that enables computers to learn from data and improve their performance without being explicitly programmed. It allows systems to recognize patterns, make predictions, and adapt to new information over time. Machine learning is used in various fields such as healthcare, finance, retail, and autonomous systems to automate tasks, enhance decision-making, and improve efficiency. With continuous advancements in computing power and data availability, machine learning is transforming industries and driving innovation.

1.2.1 TYPES OF MACHINE LEARNING

There are several types of machine learning, including supervised learning, unsupervised learning, semi-supervised learning, reinforcement learning, transfer learning and deep learning.

1. **Supervised Learning** – The model is trained on labeled data, meaning it learns from input-output pairs to make predictions. It is commonly used for classification tasks such as spam detection and regression tasks like predicting house prices.

   Examples:

   1) Classification: Identifying spam emails, tumor detection.

   2)Regression: Predicting house prices, stock market trends.

2. **Unsupervised Learning** – The model is trained on unlabeled data and identifies patterns or structures without predefined outputs. It is used for clustering, such as customer segmentation, and dimensionality reduction techniques like principal component analysis.

Examples:

1. Clustering: Customer segmentation, anomaly detection.
2. Dimensionality Reduction: Principal Component Analysis (PCA) for feature selection.

3. Semi-Supervised Learning – This approach combines a small amount of labeled data with a large amount of unlabeled data to improve learning accuracy. It is useful when labeled data is expensive or difficult to obtain, such as in medical diagnosis.

Examples:

1) Medical image classification with limited labeled samples.

2)Web page classification using a mix of labeled and unlabeled data.

4. Reinforcement Learning – The model, called an agent, learns by interacting with an environment and receiving rewards or penalties for its actions. It is used in robotics, gaming, and self-driving cars to optimize decision-making through trial and error.

Examples:

1) Robotics: Training robots to perform tasks.

2) Gaming: AlphaGo and Chess AI.

3) Self-driving cars: Learning optimal driving strategies.

CHAPTER 2

LITERATURE SURVEY

2.1 INTRODUCTION

This literature survey aims to provide a comprehensive overview of the current methods used to predict the presence of maternal health risk. The survey will review various studies that have explored this topic and analyze the different approaches and techniques used in these studies

2.2 RELATED WORKS

1.MRI brain tumor detection using deep learning and machine learning approaches

Author: Shenbagarajan Anantharajan, Venkatesh R, Shenbagalakshmi Gunasekaran, Thavasi Subramanian

Year:2024

Abstract: The development of aberrant brain cells, some of which may become cancerous, is known as a brain tumor. Early and timely detection, along with effective treatment planning, significantly enhances patients' quality of life and life expectancy. Magnetic Resonance Imaging (MRI) scans are the most commonly used technique for detecting brain tumors. However, identifying, segmenting, and analyzingtumor regions from MRI images is a complex, iterative, and labor-intensive process that heavily relies on the expertise of radiologists and clinical professionals. Traditional imaging methods often struggle to accurately locate abnormal brain regions. In recent years, Deep Learning (DL) has gained significant attention as an advanced branch of Machine Learning (ML), demonstrating effectiveness in solving complex problems. This research proposes a novel MRI-based brain tumor detection method that integrates DL and ML techniques. Initially, MRI images undergo preprocessing using the Adaptive Contrast Enhancement Algorithm (ACEA) and a median filter. Segmentation is performed using the Fuzzy c-means algorithm, followed by feature extraction—focusing on energy, mean, entropy, and contrast—using the Gray-Level Co-Occurrence Matrix (GLCM). The extracted features are then classified using the proposed Ensemble Deep Neural Support Vector Machine (EDN-SVM) classifier. The experimental results demonstrate high classification performance, achieving an accuracy of 97.93%, sensitivity of 92%, and specificity of 98% in distinguishing abnormal and normal brain tissues, highlighting the effectiveness of the proposed approach.

2.Utilizing Deep Improved ResNet-50 for Brain Tumor Classification Based on MRI

Author:Karrar Neamah, Farhan Mohamed (Senior Member, IEEE), Safa Riyadh Waheed, Waleed Hadi Madhloom Kurdi, Adil Yaseen Taha, Karrar Abdulameer Kadhim

Year:2024

Abstract:A robust approach for brain tumor classification is developed using deep convolutional neural networks (CNNs). This study utilizes an open-source dataset derived from the MRI BraTS2015 brain tumor dataset. Preprocessing steps include intensity normalization, contrast enhancement, and downscaling, along with data augmentation techniques such as rotation and flipping. The core of the proposed method is a modified ResNet-50 architecture for feature extraction, integrating transfer learning with a spatial pyramid pooling layer to leverage pre-trained parameters from ImageNet, effectively mitigating overfitting. The model's performance is systematically evaluated using multiple hyperparameters and compared against existing methods based on precision, recall, F1-score, sensitivity, and specificity. Comparative analysis against prominent CNN architectures reaffirms the model's superior performance. The optimization parameters, including 25 epochs, a learning rate of 1e-4, and a balanced batch size, contribute to its robustness and real-world applicability. This study highlights the effectiveness of deep learning, transfer learning, and spatial pyramid pooling in MRI-based brain tumor classification, providing a promising tool for medical image analysis.

3.Brain Tumor Image Identification and Classification on the Internet of Medical Things Using Deep Learning

Author: B. Raghuram, Bhukya Hanumanthu

Year: 2024

Abstract: The health services research network is showing increasing interest in the Internet of Medical Things (IoMT), where the internet is utilized to compile important health-related data. A brain tumor, caused by a mass of random cells within the brain, poses a serious threat to brain health. Accurate recognition of brain images remains challenging. To address this, this research proposes a Support Value-Based Deep Neural Network (SDNN) for brain tumor identification and classification within the e-healthcare administration framework using IoMT technology. A database for investigation is created using image data from IoT innovation and clinical images. During preprocessing, skull stripping is applied to isolate the desired brain region. The preprocessed images are then used to extract key features, including entropy, geometric, and texture features. Finally, the proposed SDNN classification system is employed to categorize the brain images as normal or abnormal based on the extracted features. Experimental results demonstrate that the proposed recognition approach significantly outperforms existing methods in terms of accuracy and efficiency.

4. Automated Segmentation of Brain Tumor MRI Images Using Deep Learning

Author: Surendran Rajendran, Suresh Kumar Rajagopal, K. Shankar, TamilvizhiThanarajan , Sachin Kumar , Najah M. Alsubaie, Mohamad Khairi Ishak, Samih M. Mostafa

Year: 2024

Abstract: Segmenting brain tumors automatically using MRI data is crucial for disease investigation and monitoring. Due to the aggressive nature and diversity of gliomas, precise and organized segmentation methods are essential for intra-tumoral classification. This study proposes a technique that utilizes Gray Level Co-occurrence Matrix (GLCM) to extract features, eliminating unnecessary details from MRI images. Compared to existing methods, the accuracy of brain tumor segmentation is significantly improved using Convolutional Neural Networks (CNNs), which are widely employed in biomedical image segmentation. By combining the results of two separate segmentation networks, the proposed method introduces a simple yet effective combinatorial strategy, which leads to more precise and complete predictions. The approach uses a U-Net and a 3D Convolutional Neural Network (CNN) to segment the images into their components. The predictions were constructed by merging the outputs of these two models in various ways. When compared to state-of-the-art methods, the proposed technique demonstrates improved precision, F-score, and sensitivity for the whole tumor, enhanced tumor, and tumor core on the validation set.

5. A Review on Brain Tumor Segmentation Based on Deep Learning Methods with Federated Learning Techniques

Author:Md. Faysal Ahamed, Md. Robiul Islam, Md. Munawar Hossain, Mominul Ahsan, Md. Nahiduzzaman, Julfikar Haider, Md. Rabiul Islam

Year: 2023

Abstract: Brain tumors have become a severe medical concern in recent years due to their high fatality rate. Traditionally, radiologists manually segment tumors, a process that is time-consuming, error-prone, and costly. In recent years, automated segmentation using deep learning has shown significant promise in addressing computer vision challenges such as image classification and segmentation. Brain tumor segmentation plays a crucial role in medical imaging by accurately determining the tumor's location, size, and shape using automated methods. Researchers have explored various machine learning and deep learning approaches, particularly convolutional techniques, to optimize segmentation accuracy. This review paper presents an analysis of the most effective segmentation methods based on widely used, publicly available datasets. Additionally, we survey federated learning methodologies to enhance global segmentation performance while ensuring data privacy. By reviewing over 100 papers, we provide a comprehensive overview of recent segmentation techniques, multi-modality fusion mechanisms, and federated learning approaches. Furthermore, we discuss unresolved challenges in brain tumor segmentation and propose a client-based federated model training strategy. This review serves as a guide for future researchers in identifying optimal solutions to existing challenges in brain tumor segmentation.

6. Magnetic Resonance Imaging-Based Brain Tumor Image Classification Performance Enhancement

Author: Belayneh Sisay Alemu, Olalekan Salau, Sultan Feisso, Endris Abdu Mohammed, Ayodeji

Year: 2023

Abstract: Brain cancer is one of the most fatal diseases, caused by the abnormal growth of defective brain tissue. Generally, brain cancer is categorized into benign and malignant tumors; however, according to the World Health Organization, it can also be classified into grades I, II, III, and IV. Magnetic Resonance Imaging (MRI) plays a vital role in diagnosing and treating brain tumors, yet accurate classification of tumor images remains a challenge due to the complexity and heterogeneity of tumor characteristics. This paper proposes a Support Vector Machine (SVM)-based classification method for brain tumor detection. The approach includes preprocessing steps such as noise reduction using median filtering or wavelet transform, segmentation using Otsu's thresholding, and feature extraction using the Gray-Level Co-occurrence Matrix (GLCM). Finally, classification is performed using an SVM model. The proposed method demonstrates significant improvements in brain tumor classification compared to previous approaches, showcasing its potential for enhancing diagnostic accuracy in MRI-based tumor detection.

7. Hybrid UNet Transformer for Brain Lesion and Tumor Segmentation

Author: Wei Kwek Soh, Hing Yee Yuen, Jagath C. Rajapakse

Year: 2023

Abstract: Supervised deep learning networks like UNet have demonstrated strong performance in segmenting brain anomalies such as lesions and tumors. However, traditional methods are designed primarily for either single-modality or multi-modality images. This study proposes the Hybrid UNet Transformer (HUT) to enhance performance in both single-modality lesion segmentation and multi-modality brain tumor segmentation. The HUT consists of two parallel pipelines: one UNet-based and the other Transformer-based. The Transformer-based pipeline utilizes feature maps from the intermediate layers of the UNet decoder during training, improving global attention and long-range correlations between voxel patches. The network processes 3D brain volumes and embeds them into voxel patches, allowing for better contextual learning. Additionally, a self-supervised training approach is introduced to further refine segmentation accuracy. Experimental results show that HUT outperforms state-of-the-art methods, demonstrating its effectiveness on datasets such as Anatomical Tracings of Lesions After Stroke (ATLAS) and Brain Tumor Segmentation (BraTS20).

8. DenseUNet+: A Novel Hybrid Segmentation Approach Based on Multi-Modality Images for Brain Tumor Segmentation

Author: Halit Çetiner, Sedat Metlek

Year: 2023

Abstract: Accurate segmentation of brain tumors is crucial for clinical diagnosis and treatment. Experts rely on multimodal magnetic resonance imaging (MRI) systems to delineate tumor boundaries, but in some cases, these boundaries may be unclear, leading to potential misdiagnoses. This study introduces DenseUNet+, a deep learning-based model designed to improve segmentation accuracy using multimodal images. The model incorporates data from four different MRI modalities into dense block structures, followed by linear operations and concatenation before passing the results to the decoder layer. DenseUNet+ was evaluated on the BraTS2021 and FeTS2021 datasets and demonstrated superior segmentation performance compared to state-of-the-art methods.

9. 3D MRI Segmentation Using U-Net Architecture for the Detection of Brain Tumor

Author: Smarta Sangui, Tamim Iqbal, Piyush Chandra Chandra, Swarup Kr Ghosh, Anupam Ghosh

Year: 2023

Abstract: Brain tumor segmentation from 3D MRI images is a crucial yet challenging task in medical image processing. Manual segmentation can be error-prone and inefficient, particularly with large datasets. The complexity of brain tumor appearances and their similarity to normal tissues further complicate the process. This study presents a modified U-Net architecture within a deep learning framework for automated brain tumor detection and segmentation. The model was evaluated on the BRATS 2020 dataset, demonstrating improved segmentation performance compared to other deep learning-based approaches.

10. Brain Tumor Detection Using 3D-UNet Segmentation Features and Hybrid Machine Learning Model

Author: Bhargav Mallampati, Sultan Alfarhood, Abid Ishaq, Furqan Rustam, Venukuthala, Imran Ashraf

Year: 2023

Abstract: Machine learning has revolutionized disease diagnosis by enhancing efficiency and accuracy in healthcare. Brain tumor detection, a critical task in medical imaging, greatly benefits from automated prediction models. This study proposes a machine learning-based approach for brain tumor detection using MRI features. The model utilizes both 3D-UNet and 2D-UNet segmentation features, incorporating statistical, shape-based, and texture-based metrics such as gray level size zone matrix, gray level co-occurrence matrix, and gray level run length matrix. A hybrid classification model is introduced, combining K-Nearest Neighbors (KNN) and Gradient Boosting Classifier (GBC) using soft voting. The combination helps balance performance variations between the two classifiers. The model achieves 64% accuracy with 2D-UNet features and a significantly improved 71% accuracy using 3D-UNet features, surpassing state-of-the-art models utilizing similar segmentation techniques.

CHAPTER 3

EXISTING WORK

3.1 EXISTING SYSTEM

The existing system for brain tumor detection relies on Magnetic Resonance Imaging (MRI) as the primary diagnostic tool. Radiologists manually analyze MRI scans to identify, segment, and classify brain tumors, which is a time-consuming and expertise-dependent process. Simple imaging techniques often fail to accurately detect abnormal brain regions due to the complexity of tumor structures.

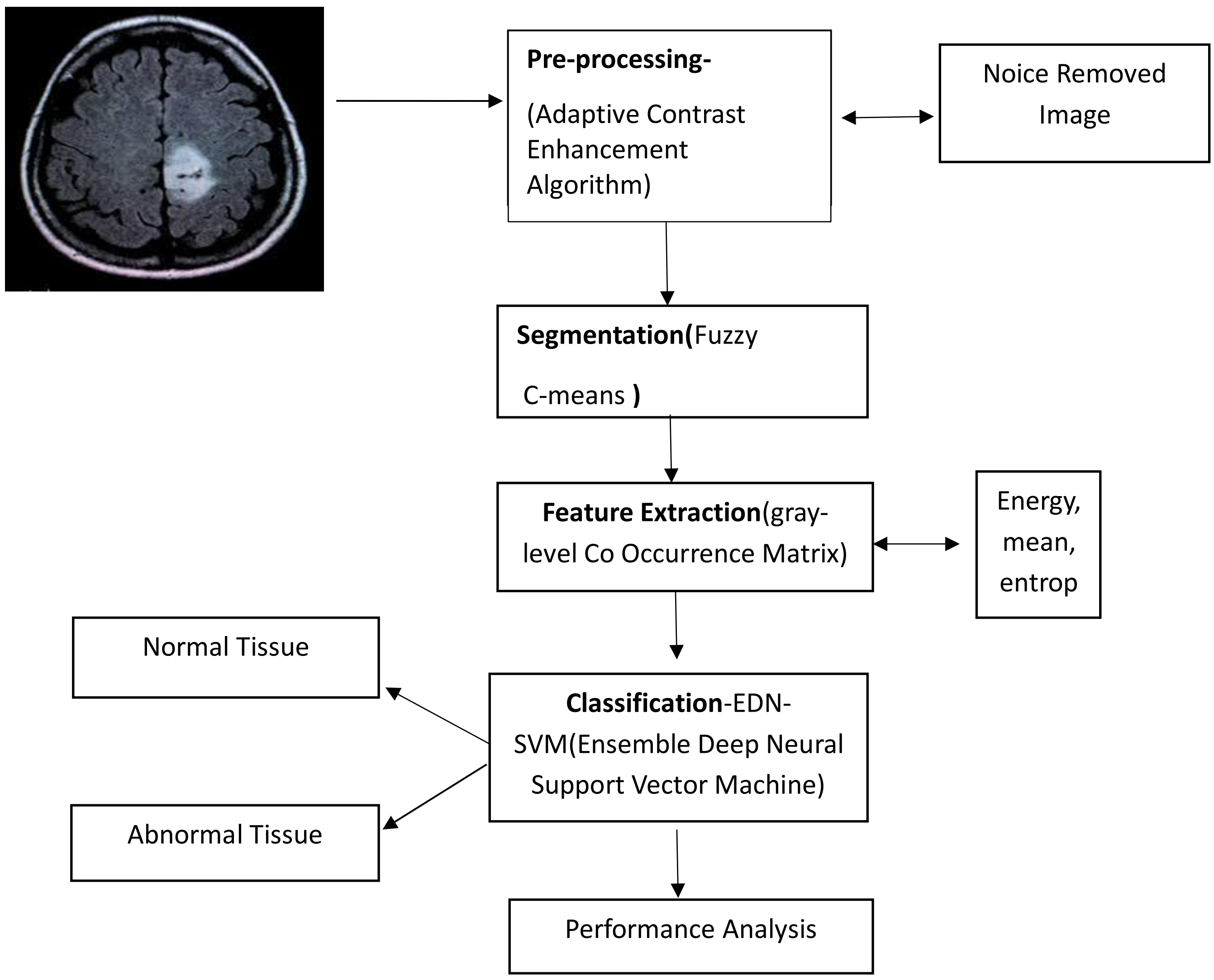
In recent years, Deep Learning (DL) and Machine Learning (ML) techniques have emerged as powerful tools for automating brain tumor detection. Several traditional methods have been used in existing systems, including:

1. Preprocessing:
    1. MRI images are enhanced using Adaptive Contrast Enhancement Algorithm (ACEA).
    2. Noise is reduced using a median filter for improved clarity.
2. Segmentation:
    1. Fuzzy C-Means (FCM) clustering is used to separate tumor and non-tumor regions.
3. Feature Extraction:
    1. Texture-based features such as energy, mean, entropy, and contrast are extracted using the Gray-Level Co-Occurrence Matrix (GLCM).
    2. This step transforms raw data into numerical features for modeling.

4. Classification:

    1. The Ensemble Deep Neural Support Vector Machine (EDN-SVM) classifier is applied to distinguish between normal and abnormal brain tissues.
    2. The approach achieves 97.93% accuracy, 92% sensitivity, and 98% specificity.

3.2 EXISTING SYSTEM ARCHITECTURE



Pre-processing-
(Adaptive Contrast Enhancement Algorithm)
Noice Removed Image
Segmentation(Fuzzy C-means)
Feature Extraction(gray-level Co Occurrence Matrix)
Energy, mean, entrop
Normal Tissue
Abnormal Tissue
Classification-EDN-SVM(Ensemble Deep Neural Support Vector Machine)
Performance Analysis

Figure 3.1 Existing System Architecture

3.3 EXISTING MODEL AND ACCURACY

Model	**Accuracy**
Sensitivity	92%
Specificity	98%
EDN-SVM	97.93%

Table 3.1 Existing Model and accuracy

3.4 ADVANTAGES

1. **High Accuracy**: The system achieves an accuracy of 97.93%, which indicates that it is highly reliable in detecting brain tumors.
2. **Advanced Preprocessing**: The use of the Adaptive Contrast Enhancement Algorithm (ACEA) and median filter enhances image quality, which aids in the accurate identification of abnormal brain regions.
3. **Effective Segmentation:** Fuzzy c-means-based segmentation allows precise separation of tumor regions from surrounding healthy tissues, improving the accuracy of the analysis.
4. **Feature Extraction with GLCM:** The system extracts essential features like energy, mean, entropy, and contrast using the Gray-Level Co-Occurrence Matrix (GLCM), which provides critical information for distinguishing between normal and abnormal tissues.
5. **Ensemble Classifier:** The Ensemble Deep Neural Support Vector Machine (EDN-SVM) classifier combines the strengths of deep learning and machine learning, improving classification performance and making the system more robust.

6. **Handling Complex Tumor Shapes:** The system is capable of detecting tumors with complex shapes and accurately classifying them, making it suitable for a wide range of tumor types.

3.3 DISADVANTAGES

1.Dependence on MRI Quality: The system's performance heavily depends on the quality and resolution of the MRI images. Low-quality or noisy images may lead to inaccurate results.

2. Limited Tumor Type Recognition: The system might not perform as well for all types of brain tumors, especially rare or atypical ones, limiting its effectiveness in diagnosing diverse tumor types.

3. **Overfitting**:Overfitting reduces model generalization performance.

3.4 PROPOSED SYSTEM

The proposed system is designed to classify images by combining basic image processing, deep learning, and machine learning techniques in a unified framework. Initially, images are preprocessed by converting them to grayscale to reduce complexity, removing noise to enhance clarity, detecting edges to highlight key structures, and cropping to focus on important regions. After preprocessing, features are extracted using four well-known pre-trained deep learning models: EfficientNetB7, MobileNetV2, Xception, and InceptionV3. These models transform each image into a numerical representation that captures its essential patterns and characteristics. The feature vectors from all four models are then concatenated into a single, comprehensive feature set, providing a rich and diverse description of the image content. This combined feature set is used to train an XGBoost classifier, which is known for its high performance and efficiency. To further improve classification accuracy, the system applies data augmentation techniques such as rotating and flipping the images, effectively increasing the size and diversity of the training dataset. The proposed method is evaluated on both a custom image dataset and a publicly available brain tumor dataset. The results demonstrate strong performance, confirming that the integration of deep learning-based feature extraction with a traditional machine learning classifier is an effective approach for image classification tasks across different domains.

3.4 Algorithm

3.4.1 XGBoost

XGBoost (Extreme Gradient Boosting) is a machine learning algorithm that builds decision trees sequentially, where each tree corrects errors from the previous one. It uses gradient descent to minimize prediction errors. XGBoost includes regularization (L1 and L2) to avoid overfitting, ensuring better model generalization.

It also features tree pruning, stopping unnecessary tree growth, and automatically handles missing data, reducing the need for extra preprocessing. XGBoost is highly parallelized, improving processing speed, especially on large datasets. The learning rate controls the contribution of each tree to the final prediction, optimizing model performance.

XGBoost is popular for classification, regression, and ranking tasks due to its speed, accuracy, and efficiency in handling large datasets. It’s commonly used in tasks like predicting house prices, classifying diseases, and ranking search results.

1.Preventing Overfitting

XGBoost uses a **learning rate** (eta) to control each tree's contribution, making the model more conservative. Regularization, shrinkage, and pruning help prevent overfitting and improve generalization by building trees level by level, trimming ineffective splits.

2. Tree Structure

XGBoost grows trees **level-wise** (breadth-first), considering all features at each level. This reduces overhead, avoids revisiting features, and better handles complex feature interactions.

3. Handling Missing Data

XGBoost uses the **Sparsity Aware Split Finding** algorithm to treat missing values as separate categories when splitting. During prediction, missing values follow a default branch for accurate results.

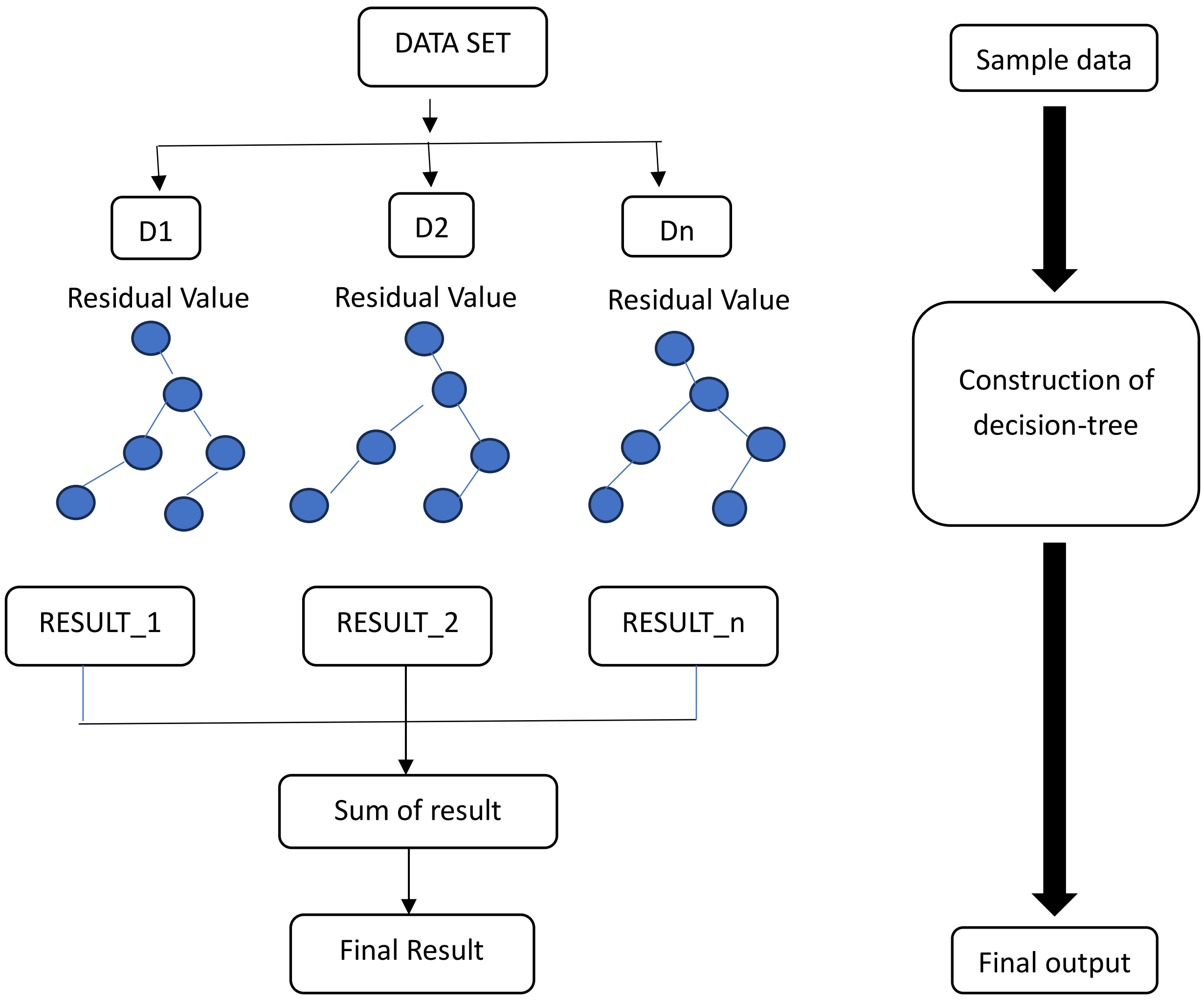


Figure 3.2 Architecture of XGBoost Algorithm

3.5 ADVANTAGES

1. **Efficient Preprocessing**: This method uses methods like grayscale conversion, blurring, edge detection, and cropping to improve image clarity and focus on important regions.
2. **Comprehensive Feature Extraction**: Combines deep features from multiple powerful pre-trained models (EfficientNetB7, MobileNetV2, Xception, and InceptionV3) for better image understanding.
3. **High Accuracy with XGBoost:** Uses XGBoost, a fast and reliable machine learning algorithm, for accurate classification based on extracted features.
4. **Hyperparameter Optimization:** Improves model performance through automatic tuning with Optuna, finding the best settings for higher accuracy.
5. **Data Augmentation:** Applies techniques like rotation, shifting, and flipping to make the model more robust and reliable on new images.
6. **Valuable for Healthcare:** Can support early detection of conditions like tumors, helping doctors make better decisions.
7. **Clear Performance Metrics:** Measures model success with metrics like accuracy, precision, recall, and F1 score for transparent evaluation.
8. **Scalable and Deployable:** Can be easily applied to new image datasets or integrated into clinical systems for real-time predictions.
9. **Interpretability:** Provides understandable outputs by combining interpretable feature-based machine learning with deep learning models.

CHAPTER 4

IMPLEMENTATION

4.1 SYSTEM REQUIREMENTS

Software and hardware specifications are critical components of the requirement specification document. They provide a detailed description of the hardware and software components required for the development, deployment, and operation of the software system.

4.1.1 SOFTWARE REQUIREMENTS

Operating System: Windows 10 or 11

Programming Languages: Python

Platform: GoogleColab or Jupyter notebook

Algorithm: XGBooster

4.1.2 HARDWARE REQUIREMENTS

GPU: 4GB and above

RAM: 8GB and above

Hard disk: NVIDIA K40 and above

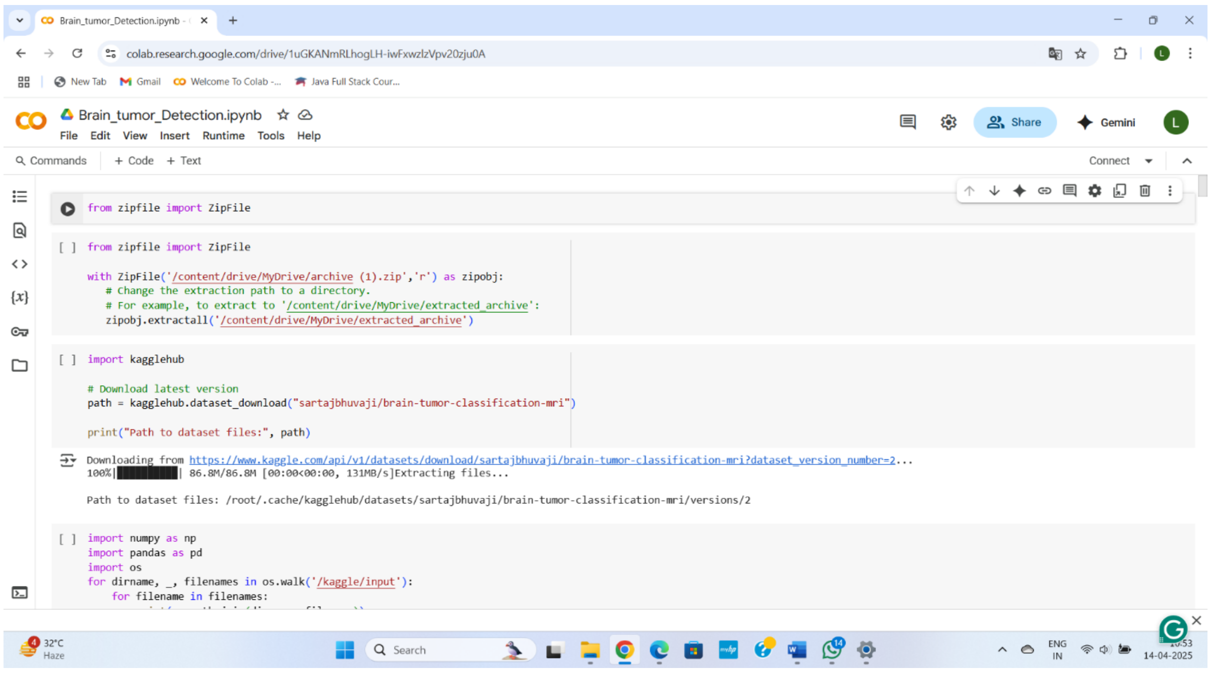
Processor: Intel i5 7th gen

4.2 SOFTWARE SPECIFICATION

4.2.1 Google Colab

Google Colaboratory (Colab) is a free, cloud-based Jupyter notebook environment that eliminates the need for local setup. You can write and execute Python code directly in your browser. A key advantage is its collaborative nature, allowing multiple team members to simultaneously edit notebooks, similar to Google Docs. Colab supports a wide range of popular machine learning libraries, easily loadable within your notebook.

Colab functions much like traditional Jupyter notebooks, but with the convenience of cloud hosting, freeing you from the need for local computing resources. Sharing notebooks is straightforward, enabling others to run your code within a standardized environment, independent of their local setups. While most libraries are pre-installed, you might need to install additional ones during initialization.


Brain_tumor_Detection.ipynb
colab.research.google.com/drive/1uGKANmRLhogLH-iwFxwzlzVpv20zju0A
New Tab
Gmail
Welcome To Colab -...
Java Full Stack Cour...
File Edit View Insert Runtime Tools Help
Share
Gemini
Commands
+ Code
+ Text
Connect
32°C Haze
Search
ENG IN
14-04-2025

```
from zipfile import ZipFile
```

```
from zipfile import ZipFile

with ZipFile('/content/drive/MyDrive/archive (1).zip','r') as zipobj:
   # Change the extraction path to a directory.
   # For example, to extract to '/content/drive/MyDrive/extracted_archive':
   zipobj.extractall('/content/drive/MyDrive/extracted_archive')
```

```
import kagglehub

# Download latest version
path = kagglehub.dataset_download("sartajbhuvaji/brain-tumor-classification-mri")

print("Path to dataset files:", path)
```

```
Downloading from https://www.kaggle.com/api/v1/datasets/download/sartajbhuvaji/brain-tumor-classification-mri?dataset_version_number=2...
100%|██████████| 86.8M/86.8M [00:00<00:00, 131MB/s]Extracting files...

Path to dataset files: /root/.cache/kagglehub/datasets/sartajbhuvaji/brain-tumor-classification-mri/versions/2
```

```
import numpy as np
import pandas as pd
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
```

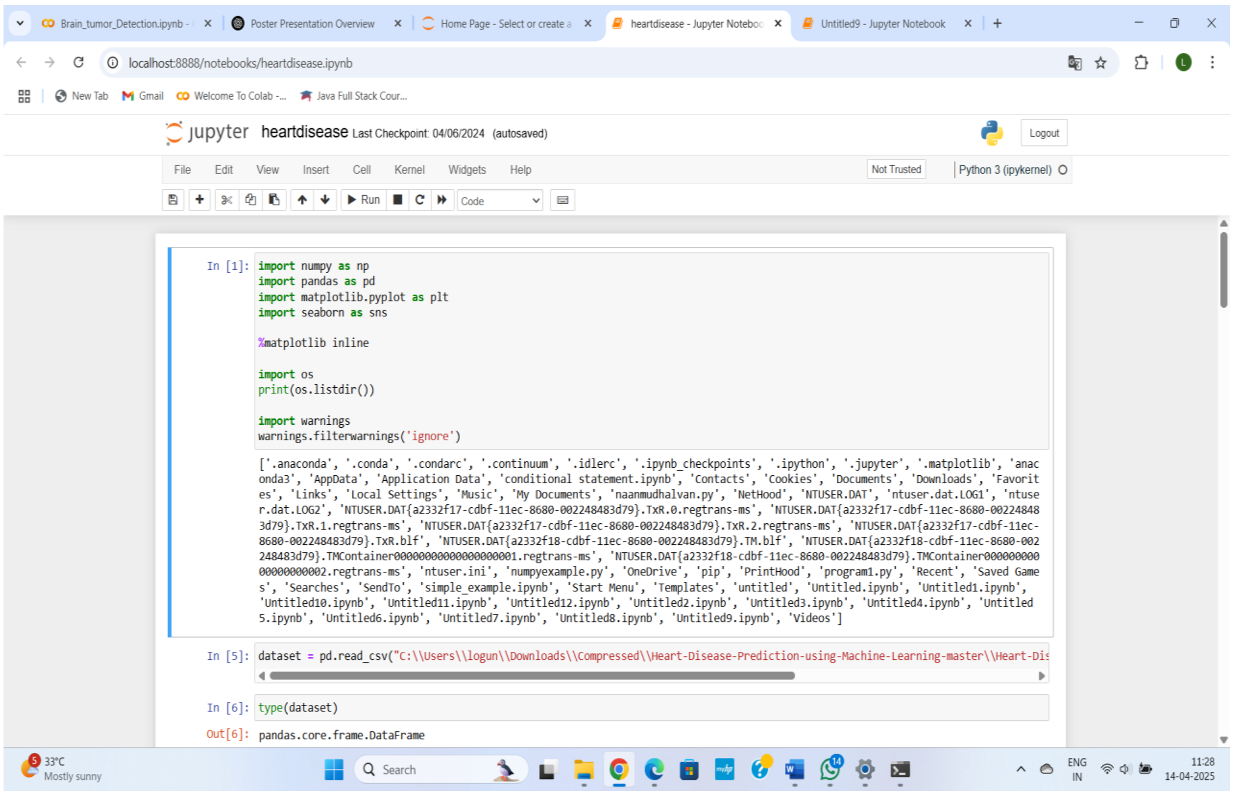
Figure 4.1 Google Colab

A Colab notebook consists of cells, which can contain either explanatory text (Markdown) or executable code and its output. Cells can be selected by clicking, and new cells can be added using the '+ CODE' and '+ TEXT' buttons, either between cells or in the toolbar. Cell order can be adjusted using the 'Cell Up' and 'Cell Down' options in the toolbar. Multiple cells can be selected using lasso selection (dragging) for consecutive cells, or by holding Ctrl (or Cmd) for non-adjacent cells and Shift for intermediate cells.

For long-running Python processes, execution can be interrupted via 'Runtime - Interrupt execution' (Ctrl/Cmd-M I). Colab inherits Jupyter's 'magic' commands, providing shorthand notations that alter cell execution. For more information, refer to Jupyter's magic commands documentation. Colab also offers automatic code completion and documentation string previews, enhancing code exploration. For instance, you can import the NumPy module and use autocompletion to explore attributes.

4.2.2 Jupyter Notebook

Jupyter Notebook is an open-source, browser-based interactive computing environment that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports several programming languages such as Python, R, and Julia, making it a versatile tool across different domains. Jupyter notebooks are commonly used for tasks like data cleaning, transformation, numerical simulation, statistical modeling, machine learning, and exploratory data analysis. Each notebook consists of cells that allow the user to write and execute code or formatted text using Markdown syntax. This modular cell structure helps in organizing work more efficiently and documenting it clearly. One of its major advantages is interactivity, allowing users to modify data and parameters in real time and immediately see the output. Visualization tools like Matplotlib, Plotly, and Bokeh can be used to create interactive charts and graphs directly within the notebook. Notebooks can be saved, exported, and shared in various formats, including HTML and PDF, and hosted on platforms like GitHub or Binder. Jupyter is also integrated into cloud-based environments like Google Colab, making it accessible without installation. Overall, Jupyter Notebook is a powerful and user-friendly platform widely adopted in education, research, and industry.


localhost:8888/notebooks/heartdisease.ipynb
jupyter heartdisease Last Checkpoint: 04/06/2024 (autosaved)
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns

        %matplotlib inline

        import os
        print(os.listdir())

        import warnings
        warnings.filterwarnings('ignore')

        ['.anaconda', '.conda', '.condarc', '.continuum', '.idlerc', '.ipynb_checkpoints', '.ipython', '.jupyter', '.matplotlib', 'anac
        onda3', 'AppData', 'Application Data', 'conditional statement.ipynb', 'Contacts', 'Cookies', 'Documents', 'Downloads', 'Favorit
        es', 'Links', 'Local Settings', 'Music', 'My Documents', 'naanmudhalvan.py', 'NetHood', 'NTUSER.DAT', 'ntuser.dat.LOG1', 'ntuse
        r.dat.LOG2', 'NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.0.regtrans-ms', 'NTUSER.DAT{a2332f17-cdbf-11ec-8680-00224848
        3d79}.TxR.1.regtrans-ms', 'NTUSER.DAT{a2332f17-cdbf-11ec-8680-002248483d79}.TxR.2.regtrans-ms', 'NTUSER.DAT{a2332f17-cdbf-11ec-
        8680-002248483d79}.TxR.blf', 'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TM.blf', 'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002
        248483d79}.TMContainer00000000000000000001.regtrans-ms', 'NTUSER.DAT{a2332f18-cdbf-11ec-8680-002248483d79}.TMContainer000000000
        00000000002.regtrans-ms', 'ntuser.ini', 'numpyexample.py', 'OneDrive', 'pip', 'PrintHood', 'program1.py', 'Recent', 'Saved Game
        s', 'Searches', 'SendTo', 'simple_example.ipynb', 'Start Menu', 'Templates', 'untitled', 'Untitled.ipynb', 'Untitled1.ipynb',
        'Untitled10.ipynb', 'Untitled11.ipynb', 'Untitled12.ipynb', 'Untitled2.ipynb', 'Untitled3.ipynb', 'Untitled4.ipynb', 'Untitled
        5.ipynb', 'Untitled6.ipynb', 'Untitled7.ipynb', 'Untitled8.ipynb', 'Untitled9.ipynb', 'Videos']

In [5]: dataset = pd.read_csv("C:\\Users\\logun\\Downloads\\Compressed\\Heart-Disease-Prediction-using-Machine-Learning-master\\Heart-Dis

In [6]: type(dataset)

Out[6]: pandas.core.frame.DataFrame
```

Figure 4.2 Jupyter Notebook Setup

4.2.3 Python

Python is a powerful general-purpose programming language. It is used in web development, data science, creating software prototypes, and so on. Fortunately for beginners, Python has simple easy-to-use syntax. This makes Python an excellent language to learn to program for beginners.

Python's simplicity and readability, coupled with its vast libraries and dynamic typing, drive its popularity across diverse applications like web development and data science. Its clear syntax and cross-platform compatibility enable rapid development, making it ideal for both beginners and experienced programmers. As the demand for automation and data-driven solutions grows, Python's role in fields like machine learning and AI continues to expand. Its vibrant community and ongoing development ensure that Python remains a leading language for innovation.

Figure 4.3 python

4.2.4 Library

4.2.4.1 Numpy

The numpy library is used in the brain tumor detection project for performing fast and efficient numerical operations on image data. It allows MRI images to be represented as multidimensional arrays, enabling tasks such as reshaping, normalization, filtering, and feature extraction. numpy is essential for handling pixel values, computing statistical features like mean, variance, and standard deviation, and preparing the data for model input. Its compatibility with other libraries like Opencv, scikit-learn, and pandas makes it a foundational tool for processing and analysing medical image data in machine learning workflows.

4.2.4.2 Pandas

Pandas are used to read, clean, and manage MRI feature data from CSV files. It helps in selecting features, handling labels, and preparing data for model training and testing. It also supports saving results back to CSV. Additionally, it enables efficient manipulation of large datasets with minimal code.Pandas also provides powerful tools for data exploration, such as filtering, grouping, and statistical summaries.

4.2.4.3 Matplotlib

The matplotlib.pyplot library is used in the brain tumor detection project to visualize MRI images and their processed outputs. It helps display original, segmented, and preprocessed images side by side for better understanding and comparison. Visualization using pyplot allows researchers and healthcare professionals to interpret image transformations, observe tumor regions, and evaluate the effectiveness of preprocessing steps.

4.2.4.4 Sklearn

The sklearn (scikit-learn) library is used in the brain tumor detection project for various machine learning tasks, including data preprocessing, model evaluation, and performance measurement. It provides tools for splitting the dataset into training and testing sets, scaling features using StandardScaler, and encoding tumor class labels with LabelEncoder. Additionally, sklearn offers essential metrics such as accuracy, precision, recall, F1 score, and confusion matrix to evaluate the classification model's performance.

4.2.4.5 cv2 (OpenCV)

The cv2 (OpenCV) library is used in the brain tumor detection project for image processing tasks on MRI scans. It enables reading and displaying MRI images, converting them to grayscale, resizing, and applying filters like Gaussian blur for noise reduction. cv2 is also used for important operations such as cropping tumor regions, performing edge detection (e.g., using Canny), and preparing images for feature extraction. Furthermore, it facilitates real-time visualization and debugging of image transformations during the preprocessing pipeline. OpenCV also supports integration with other libraries like NumPy, enhancing the flexibility and speed of pixel-level operations.

4.3 SYSTEM ARCHITECTURE

The proposed system architecture is designed to classify MRI images through a combination of image pre-processing, deep learning-based feature extraction, and machine learning classification. The process begins with an MRI image dataset, which is subjected to pre-processing techniques to improve the quality of input data. Specifically, a Gaussian blur is applied to reduce image noise and enhance important structural details. This step helps in smoothing the images, making them more suitable for downstream feature extraction by minimizing irrelevant variations.

Once the images are pre-processed, they are passed through four powerful pre-trained convolutional neural networks: EfficientNetB7, MobileNetV2, Xception, and InceptionV3. These deep learning models are known for their high performance in image analysis tasks and are used to extract meaningful and high-level features from the MRI images. Each model captures different patterns and representations, and the outputs from all four networks are concatenated to form a single comprehensive feature vector. This fusion ensures that the most informative and diverse characteristics of the images are represented, increasing the robustness of the feature set used for classification.

The final stage involves feeding the combined feature vector into an XGBoost classifier (Extreme Gradient Boosting), a highly efficient machine learning algorithm known for its strong performance on structured data. XGBoost learns to associate the input features with their corresponding class labels, such as tumor types or normal tissues. After classification, the system evaluates its performance using standard metrics like accuracy, precision, recall, and F1-score. This architecture demonstrates the effectiveness of integrating deep learning-based feature extraction with a machine learning classifier, particularly in medical imaging scenarios where interpretability and accuracy are critical.

Overall, the proposed method offers a practical and scalable approach for medical image classification. By leveraging pre-trained deep learning models and a powerful classifier like XGBoost, the system avoids the need for large-scale end-to-end training while still achieving high accuracy. This makes it well-suited for applications involving limited datasets, such as brain MRI scans, where extracting relevant features and achieving precise classifications are essential for clinical decision-making.

The integration of multiple pre-trained CNN models allows the system to capture complementary features that a single model might miss, enhancing overall classification performance. Pre-processing plays a crucial role in reducing noise and improving feature quality, which directly impacts the accuracy of the extracted representations. Using XGBoost as the classifier combines the strengths of gradient boosting with deep features, enabling efficient learning from limited data. This hybrid approach balances computational efficiency with high predictive power, making it ideal for medical image analysis tasks.

To further enhance the model's generalization ability, cross-validation can be incorporated into the training pipeline. Cross-validation ensures that the model's performance is assessed across multiple subsets of the dataset, providing a more reliable estimate of its effectiveness and reducing the risk of overfitting. This technique helps in validating the robustness of the classifier and ensures consistent results, which is especially important in medical applications where accuracy and dependability are critical. It also aids in selecting optimal hyperparameters for the XGBoost classifier, improving overall performance. By systematically evaluating the model on different data folds, cross-validation supports a more stable and trustworthy deployment in clinical settings.

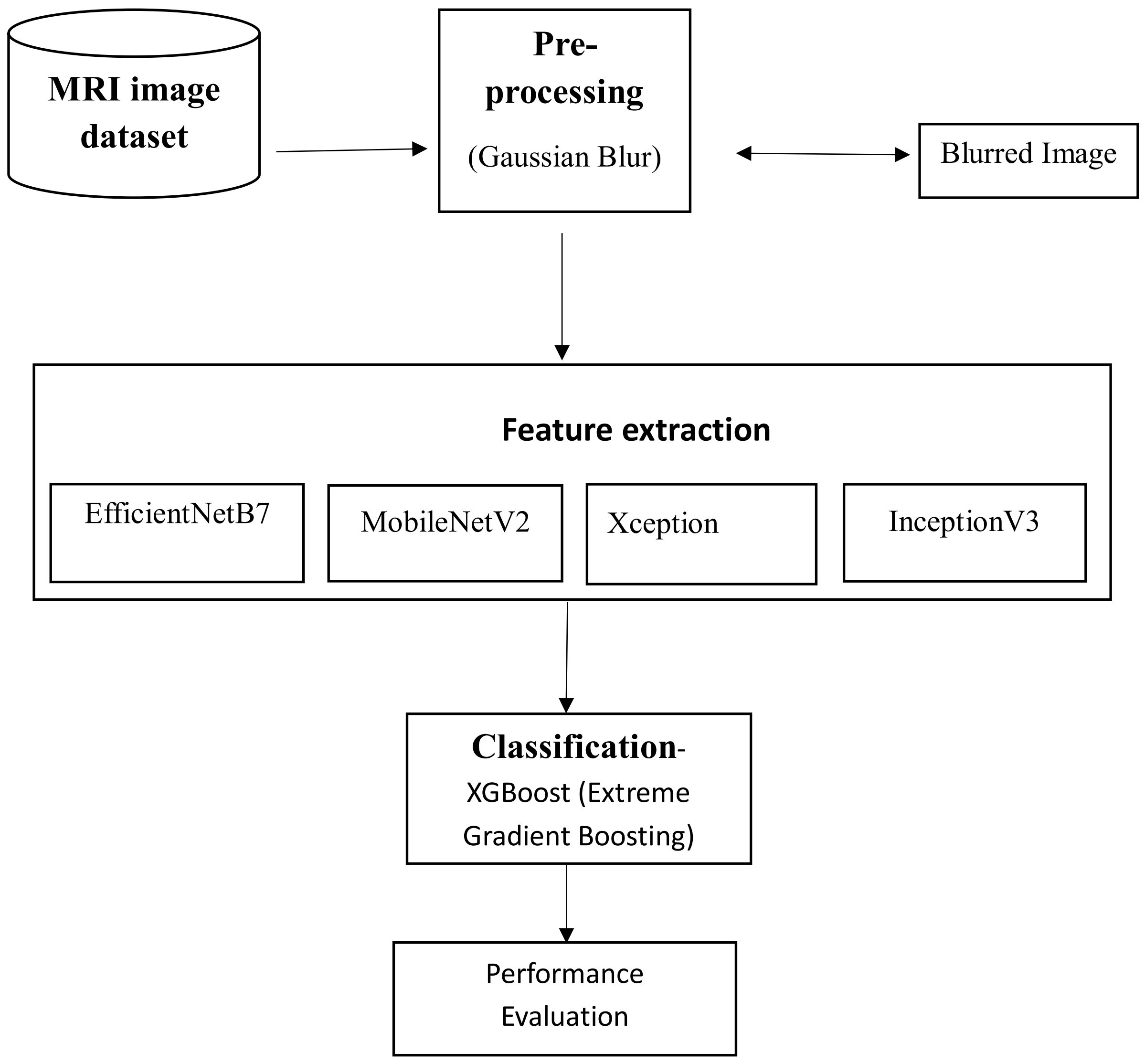


Figure 4.4 System Architecture

4.4 DATA FLOW DIAGRAM

The data flow diagram represents a machine learning-based MRI image analysis system where the process starts with the user providing MRI images that are first preprocessed to enhance their quality. These images, along with an extracted dataset, undergo further preprocessing and optional feature extraction to prepare the data for model training. The trained model is then capable of making predictions on new input images. A model evaluation step assesses the accuracy of the predictions and generates a report. This evaluation feedback supports continuous model refinement, ensuring improved diagnostic performance over time.

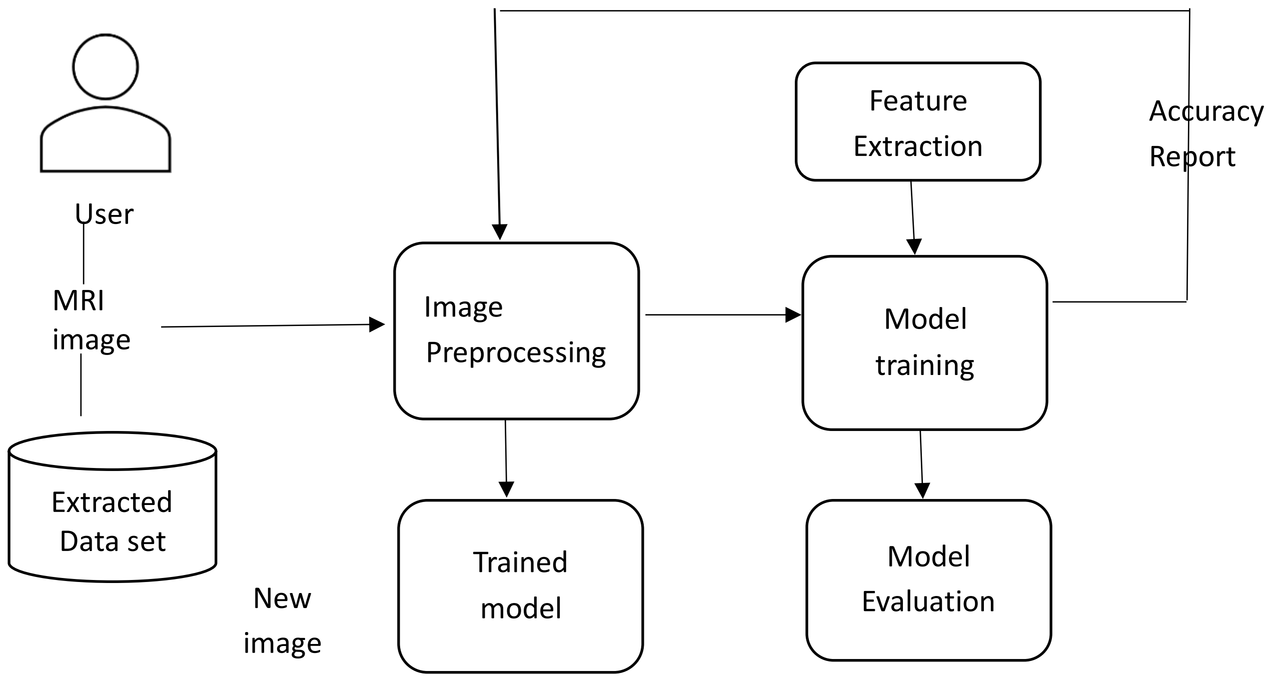


Figure 4.5 Data Flow diagram

4.6 SYSTEM MODULES

1.Data Collection Module

2.Data Preprocessing Module

3.Feature Extraction Module

4. Model Training Module

5. Model Evaluation Module

6. Model Deployment Module

7. Monitoring and Maintenance Module

4.7 MODULES DESCRIPTION

4.7.1 Data Collection Module

Data collection is the first step in developing a brain tumor classification system using MRI images. It involves gathering high-quality images from trusted sources such as Kaggle, BRATS, or authorized medical institutions. The data must be accurately categorized into classes like Glioma, Meningioma, Pituitary Tumor, and No Tumor, and organized in a structured folder format for ease of processing.

Ensuring data quality is vital blurry, corrupted, or mislabeled images should be removed. If the dataset is limited, augmentation techniques such as flipping, rotation, and zooming can help increase its size and variability. Metadata like scan resolution and imaging type can also be included to enrich the dataset.

Sensitive patient information, if present, should be excluded or anonymized to maintain confidentiality. A well-structured, clean, and responsibly sourced dataset provides the foundation for building an accurate and reliable brain tumor classification model

4.7.2 Data Preprocessing Module

Data Preprocessing Module is a vital stage that prepares MRI images for analysis and model training by cleaning, enhancing, and standardizing the raw data. The process begins with converting each image to grayscale to reduce complexity while preserving critical details. Cropping is applied to isolate the region of interest, and noise is minimized using Gaussian blur. Techniques like Canny edge detection and sharpening filters are used to emphasize tumor boundaries.

Each image is resized to a consistent size and normalized to scale pixel values between 0 and 1, ensuring uniformity across the dataset. Corrupted or unreadable images are identified and removed. The final output consists of structured image arrays with labels, ready for feature extraction or training. Efficient preprocessing improves model accuracy by providing clean and inputs.

4.7.3 Feature Extraction Module

4.7.3.1 EfficientNetB7

EfficientNet is a family of convolutional neural network (CNN) architectures designed by Google AI that aims to balance efficiency and accuracy by optimizing the scaling of model depth, width, and resolution.

1. **Depth**: Refers to the number of layers in the network. Deeper networks can learn more complex patterns but are harder to train.
2. **Width**: Refers to the number of units in each layer. Wider layers allow the model to learn more representations but can increase computational cost.
3. **Resolution**: Refers to the input image resolution. Higher resolution images can provide more detailed features but also require more computation.

EfficientNetB0 is the base model, and each subsequent version (B1 to B7) scales up the model in terms of depth, width, and resolution.

1.EfficientNetB0: Base model with a balance between accuracy and efficiency.

2.EfficientNetB7: The largest model with the highest accuracy, but also the most computationally expensive.

Advantages:

1. Accuracy
2. Efficiency
3. Flexible Scaling

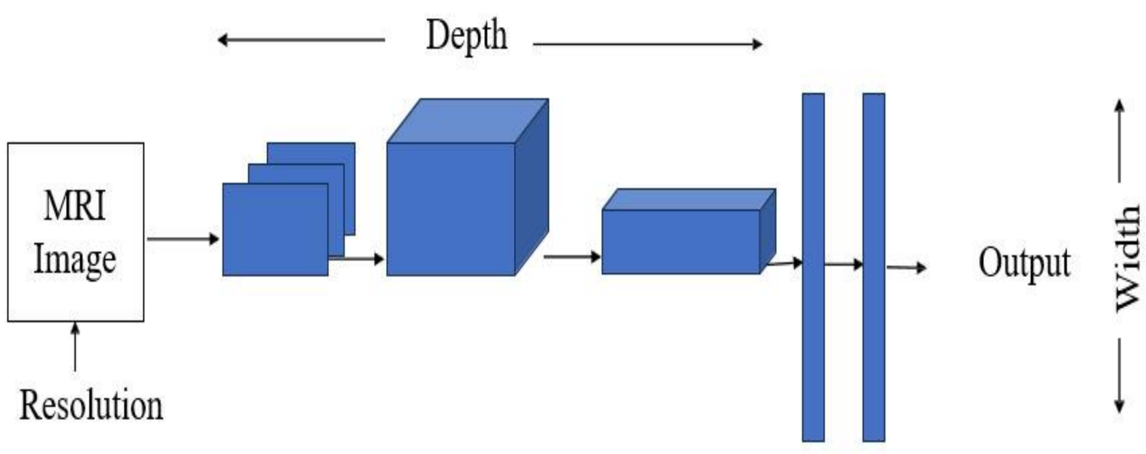


Figure 4.6 EfficientNet

4.7.3.2 MobileNetV2

MobileNetV2 is the first version of the MobileNet Family, developed by Google.It is 10x faster and smaller than VGG16.MobileNetV2 is a lightweight, efficient deep learning model architecture designed primarily for mobile and embedded devices where computing power and memory are limited. It's a follow-up to MobileNetV1 and improves both speed and accuracy while keeping the model size small.

The main contributions to this architecture are:

1. Depth-wise Separable Convolution instead of Convolution
2. Width and resolution hyper parameter.

A typical block looks like:

1. 1×1 Convolution (expansion)
2. 3×3 Depth wise Convolution
3. 1×1 Linear Convolution (projection)

Advantages

1. Efficiency
2. Flexibility
3. Improved Performance

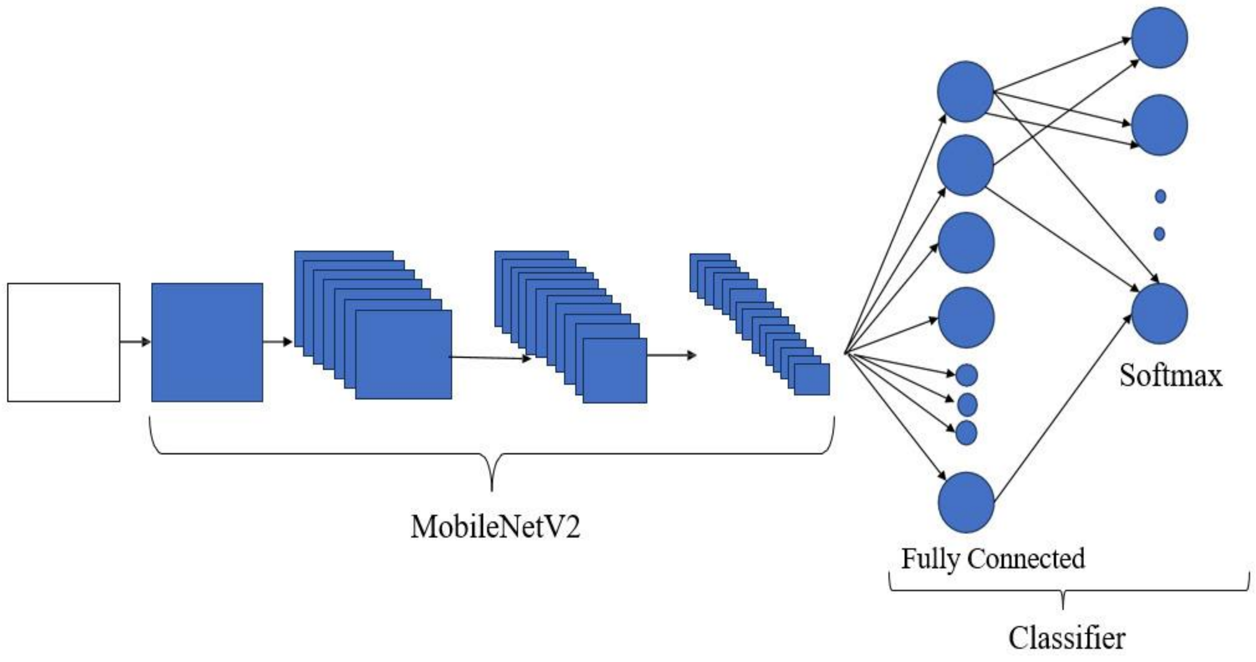


Figure 4.7 MobileNetV2

4.7.3.3 Xception

Xception, an abbreviation for “**Extreme Inception**,” represents a milestone in convolutional neural network (CNN) design. Conceived by François Chollet, the creator of the Keras deep learning library, Xception was introduced in 2017 as an evolution of the Inception architecture. In this blog post, we will delve into the architecture and approaches that distinguish Xception in the realm of deep learning.

1. **Depth wise Separable Convolutions**

   Splits standard convolutions into spatial (depth wise) and channel-wise (point wise) operations, reducing parameters and computation.

2. **Separable Convolution Blocks**

   Uses stacked depth wise + point wise convolutions to efficiently capture patterns.

3. **Entry, Middle & Exit Flows**

- **Entry Flow:** Extracts initial features.
- **Middle Flow:** Repeatedly applies separable convolution blocks to deepen and refine features.
- **Exit Flow:** Final feature refinement and prediction.

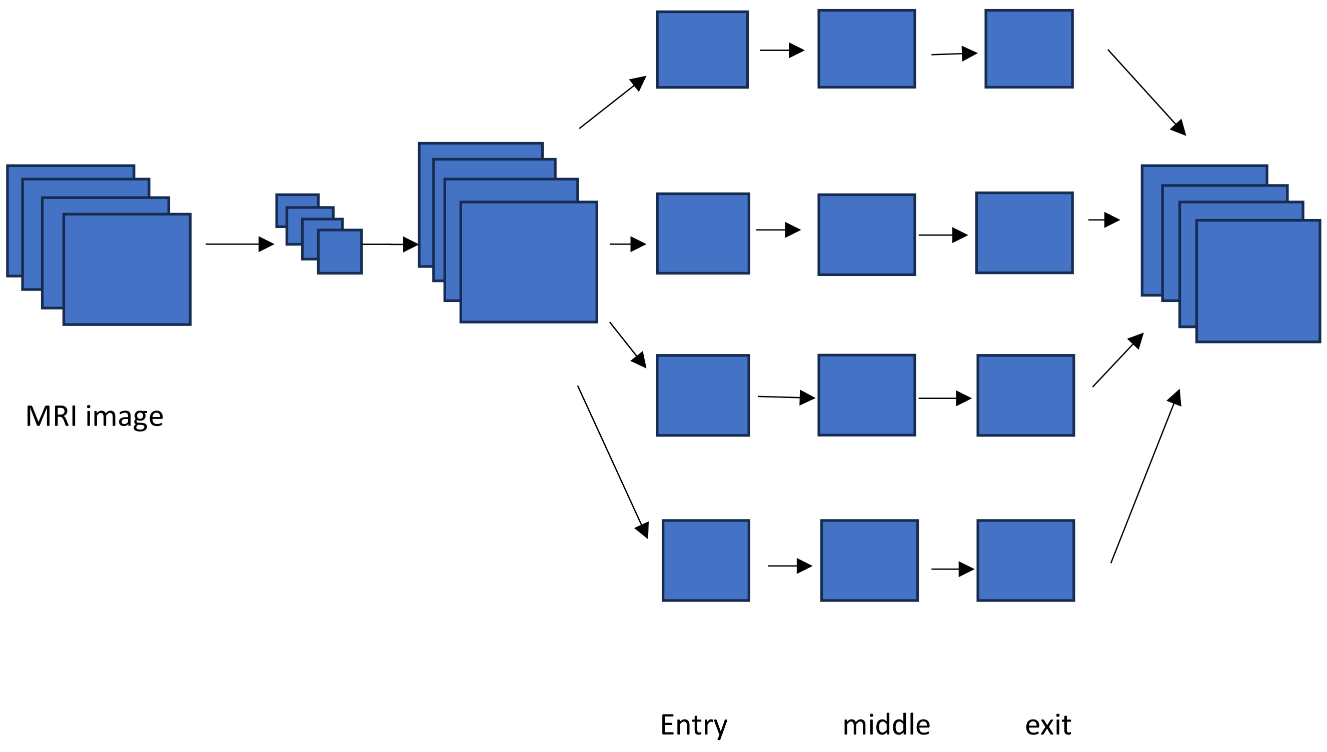


Figure 4.8 Xception

Advantage

1. Good for transfer learning
2. Fewer parameters
3. Simple, modular structure

4.7.3.4 InceptionV3

InceptionV3 is a deep convolutional neural network architecture designed for image classification. It's an improved, optimized version of the original Inception (GoogLeNet) model, balancing accuracy and computational efficiency.

1.Input Layer

- The input data (e.g., images) is received, typically as a 3D tensor (height, width, and channels like RGB for color images).

2. Convolutional Layer

- Filters (kernels) slide over the input to perform convolutions, extracting low-level features like edges and textures.

3. Pooling Layer

- Reduces the spatial dimensions (height and width) of the feature maps to decrease computation and control over fitting.
- **Max pooling** (takes the max value in each region) is commonly used.

5. Flattening

- Converts the 2D feature maps into a 1D vector, making it ready for the fully connected layers.
- It acts as a bridge between the convolutional layers and the dense layers in a neural network.

6. Fully Connected Layer

- This layer takes the flattened vector and applies weights to each neuron. It is used to make high-level decisions based on the features extracted by earlier layers.
- It connects every neuron to every neuron in the previous layer.

7. Output Layer

- The final layer produces the output, such as class probabilities in classification tasks (using softmax) or a continuous value in regression tasks.

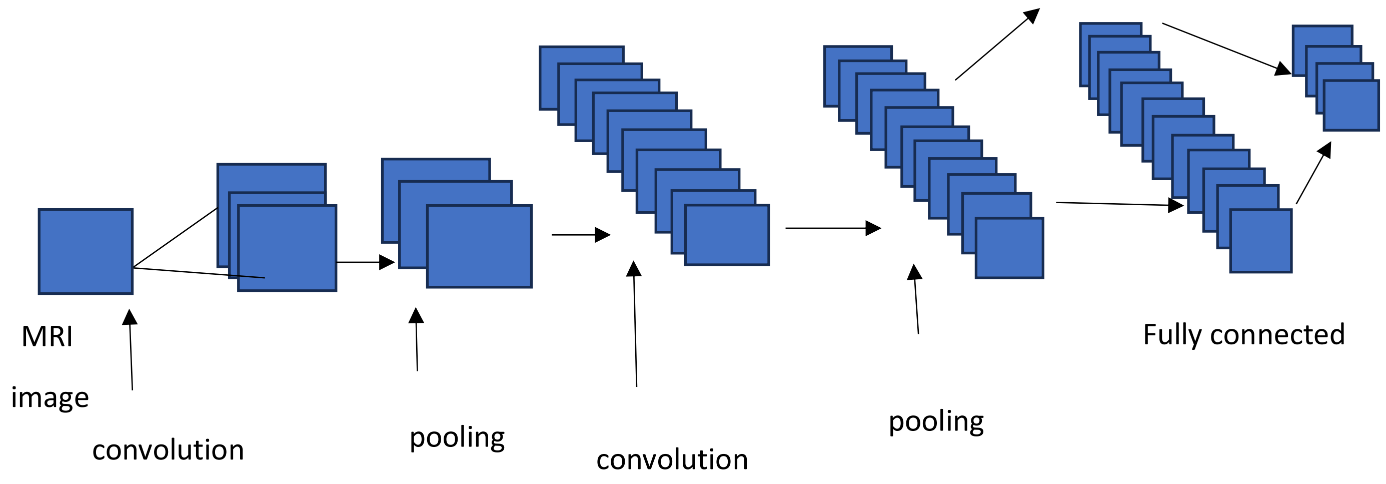


Figure 4.9 InceptionV3

Advantages

1. Detects small and large tumor features wellFast and uses less memory.
2. Deep and powerful model.

4.7.4 Model Training Module

Model Training Module (XGBoost) is a stage where XGBoost, a powerful gradient boosting algorithm, is used to train the model for brain tumor classification. XGBoost is known for its speed, accuracy, and ability to handle large datasets efficiently. The module begins by preparing the dataset for training, which may involve feature extraction from the preprocessed MRI images.

Once the features are ready, XGBoost is configured with key parameters such as the number of boosting rounds, learning rate, maximum depth of trees, and the number of estimators (trees). These parameters are crucial for controlling model complexity and preventing overfitting. The model is then trained by iteratively fitting weak decision trees to the residuals (errors) of the previous trees, with each tree improving the model's performance by focusing on the areas where previous trees made errors.

During training, the model uses a loss function (typically logistic loss for binary classification or softmax for multi-class) to measure prediction errors, and optimization is performed to minimize this loss. Techniques like early stopping can be employed to avoid overfitting, where training halts if the model's performance on the validation data stops improving.

Once trained, the XGBoost model is evaluated using a validation set, and performance metrics like accuracy, precision, recall, and F1-score are calculated to assess how well the model generalizes to new, unseen data. The trained model is then saved for future use, including prediction tasks and model deployment.

4.7.5 Model Evaluation Module

The Model Evaluation Module is a crucial part of the machine learning pipeline, responsible for assessing the trained model's performance using unseen test data. It involves computing various evaluation metrics such as accuracy, precision, recall, F1-score, and confusion matrix to determine how well the model is performing in classifying different types of brain tumors. These metrics help in identifying the model's strengths and areas where it may be underperforming. For example, a high precision but low recall might indicate the model is conservative in making positive predictions. Visualisation tools like ROC curves or confusion matrices may also be used to get a deeper understanding of the classifier's behaviour. This module ensures the model is robust, generalizes well, and is ready for deployment in real-world diagnostic scenarios

4.7.6 Model Deployment Module

The Model Deployment Module involves integrating the trained machine learning model into a production environment where it can be accessed and used by end users, such as doctors or medical professionals. This includes converting the model into a deployable format and embedding it into a user-friendly interface or web application. Tools such as Flask, FastAPI, or TensorFlow Serving can be used to expose the model via an API. The module ensures that the model can accept new MRI images as input, process them through the same preprocessing and feature extraction pipeline, and return accurate predictions in real-time Deployment also considers scalability, latency, and security to ensure smooth and secure operation in clinical settings.

4.7.7 Monitoring and Maintenance

The Monitoring and Maintenance module of the Brain Tumor Classification System ensures the system functions reliably over time. It involves regularly assessing model performance (e.g., accuracy, precision, recall) to detect issues like model drift or prediction errors and updating or retraining the model as needed using new data. Effective data management is also essential—ensuring that the input MRI data remains clean, accurate, and consistent through ongoing checks and preprocessing pipelines.

Additionally, system maintenance includes software updates, bug fixes, and hardware scalability. Real-time monitoring tools, logs, and alert systems are used to track system health, while user feedback helps identify necessary improvements or feature enhancements. Security measures are also enforced to protect sensitive data and ensure compliance with ethical standards. Backup and recovery processes are implemented to prevent data loss or downtime. Automated monitoring scripts and scheduled audits help detect anomalies early, ensuring system stability. Regular documentation updates and training sessions for end-users and developers help maintain system transparency and usability. This module ensures the continued accuracy, security, and usability of the system in real-world clinical or research applications.

Furthermore, collaboration between data scientists, clinicians, and IT teams is critical for the ongoing success of the Monitoring and Maintenance module. Continuous feedback loops from clinical experts help validate model predictions and uncover edge cases that automated systems may overlook. Integrating user insights into iterative model improvements fosters trust and transparency. Additionally, performance dashboards and reporting tools enable stakeholders to visualize trends in system behavior and performance metrics over time. This cross-functional approach not only enhances the reliability of the Brain Tumor Classification System but also ensures that it remains aligned with evolving clinical standards and technological advancements, ultimately improving patient outcomes and research capabilities.

Regular performance benchmarking against updated datasets ensures the model adapts to new imaging technologies and diagnostic criteria. As clinical environments evolve, this proactive approach helps maintain the system’s relevance and accuracy, minimizing risks associated with outdated algorithms or data inconsistencies.

CHAPTER 5

RESULT AND ANALYSIS

5.1 INPUTS

5.1.1 Raw MRI Images

The raw MRI images in the dataset are typically in formats such as .jpg, which are commonly used for medical imaging. The images are usually high-resolution grayscale images, where each pixel represents an intensity value corresponding to tissue density.

- Size: MRI images commonly have dimensions like 256x256 pixels or higher, depending on the dataset and scanner used.
- Color Mode: MRI images are often grayscale, with pixel intensity values ranging from 0 (black) to 255 (white). If the dataset contains RGB images, they are converted to grayscale to ensure consistency during preprocessing.

5.1.2 Class Labels (Categories)

MRI images are grouped into folders based on the type of brain condition present. The dataset typically includes the following categories:

- **Glioma**: Tumors that originate from glial cells in the brain.
- **Meningioma**: Tumors arising from the meninges, the protective tissue layers of the brain and spinal cord.
- **Pituitary**: Tumors developing in the pituitary gland at the base of the brain.
- **No Tumor**: Images showing healthy brain tissue with no signs of tumors.

5.1.3 Folder Structure

The dataset's folder structure is typically organized by **class label**, and each folder contains MRI images belonging to that particular class. This folder structure is crucial for supervised machine learning tasks because it allows the model to associate images with their corresponding labels.

5.1.4 File Formats

The raw data might come in different formats, such as:

- The raw data is typically in image formats such as .jpg. These files often need to be read and converted into arrays for use in machine learning Fig

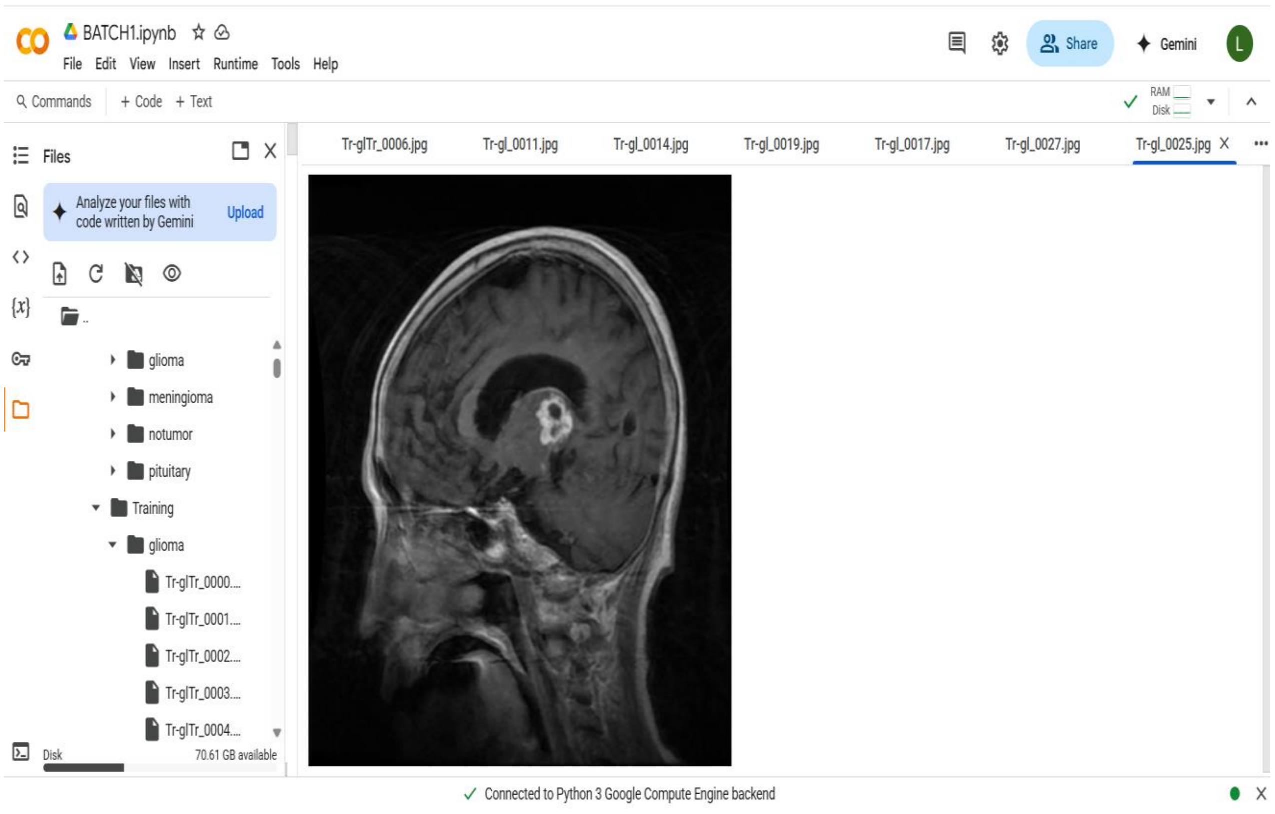
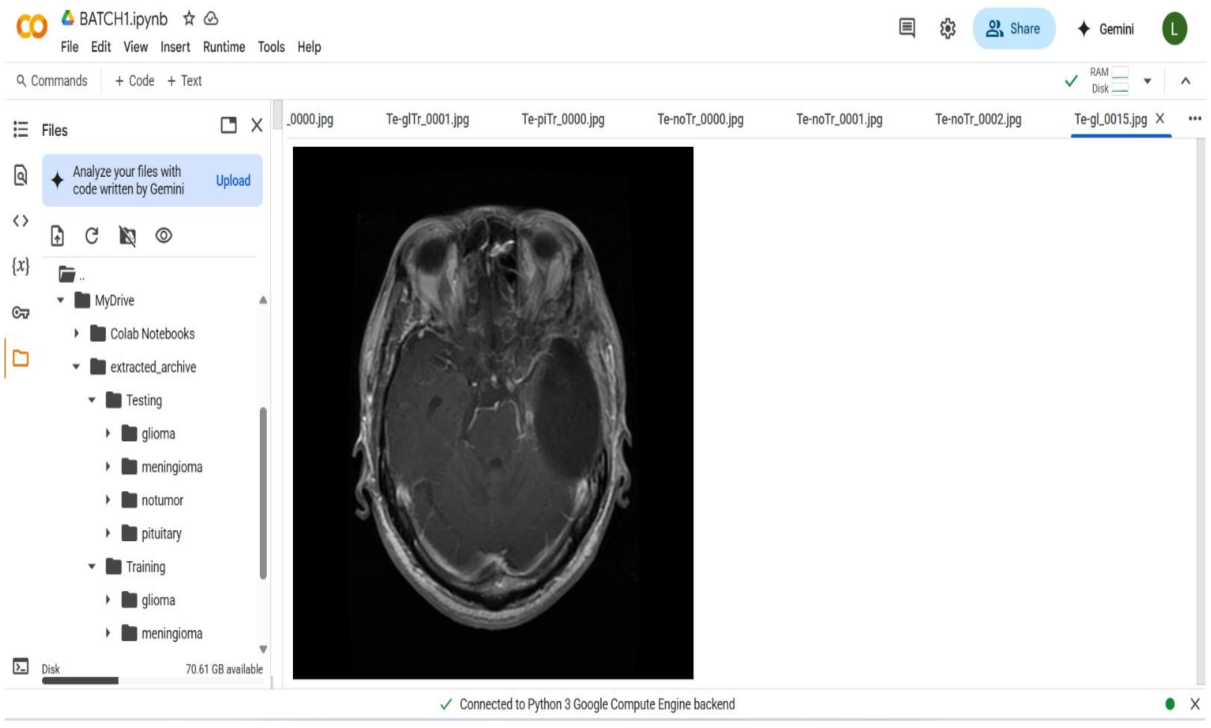


Figure 5.1 Input Training image



Connected to Python 3 Google Compute Engine backend

Figure 5.2 Input Test image

5.2 PERFORMANCE METRICS

To measure performance, several evaluation metrics are employed, including Sensitivity, Specificity, Precision, Accuracy, and F1 Score. These metrics are crucial for assessing classification models and quantifying the effectiveness of a predictive model, particularly in sensitive domains like medical diagnostics.

Advantages

1. Measures how well the model detects tumors using metrics such as accuracy, precision, and recall.
2. Helps reduce false positives and false negatives.
3. Supports the safe use of the model in medical decision-making

Sensitivity:

Sensitivity is a measure of a model's ability to identify positive instances. It is also referred to as Recall and is mathematically expressed as,

$$\text{Sensitivity} = \frac{(D_p/N_p)}{(D_p/N_p) + (D_n/N_n)} * 100$$

Precision:

Precision measures the number of positive class predictions that are actually correct. It is mathematically expressed as,

$$\text{Precision} = \frac{(D_p/N_p)}{(D_p/N_p) + (D_e/N_e)} * 100$$

Accuracy:

Accuracy is the proportion of correctly predicted data points out of the total number of predictions. It is defined as the sum of true positives and true negatives divided by the total number of true positives, true negatives, false positives, and false negatives. It is mathematically expressed as:

$$\text{Accuracy} = \frac{(D_p/N_p) + (D_m/N_m)}{p + m} * 100$$

F1Score:

The F1 Score is the harmonic mean of Precision and Recall, offering a balanced measure that is especially useful in classification problems with imbalanced datasets. It accounts for both false positives and false negatives, making it ideal when both types of errors carry significant consequences. A high F1 Score indicates that the model achieves strong performance in both identifying relevant instances (recall) and minimizing incorrect positive predictions (precision). Mathematically, it is expressed as: F1 Score = 2 × (Precision × Recall) / (Precision + Recall), emphasizing that both metrics must be reasonably high to achieve a strong F1 Score

$$\text{F1 Score} = \frac{2*(precision*sensitivity)}{precision+sensitivity}$$

NO	PERFORMANCE MATRIC	RATIO
1.	Accuracy	0.9833
2.	Precision	0.98
3.	Recall	0.98
4.	F1	0.98

Table 5.1 Proposed Model and Accuracy

5.2 Output

The output of this model is to predict fetal health status (0: Normal, 1: Suspect, 2: Pathological) based on input features from the dataset. The type of the output is a float representing the class label. The performance of the proposed model (XGBoost) is compared with existing models using standard classification metrics.

The bar chart compares the performance of the proposed XGBoost model to existing models using Accuracy, Precision, Recall, and F1-Score. All values are expressed as percentages. The proposed model.The bar chart compares the performance of the proposed XGBoost model to existing models using Accuracy, Precision, Recall, and F1-Score.

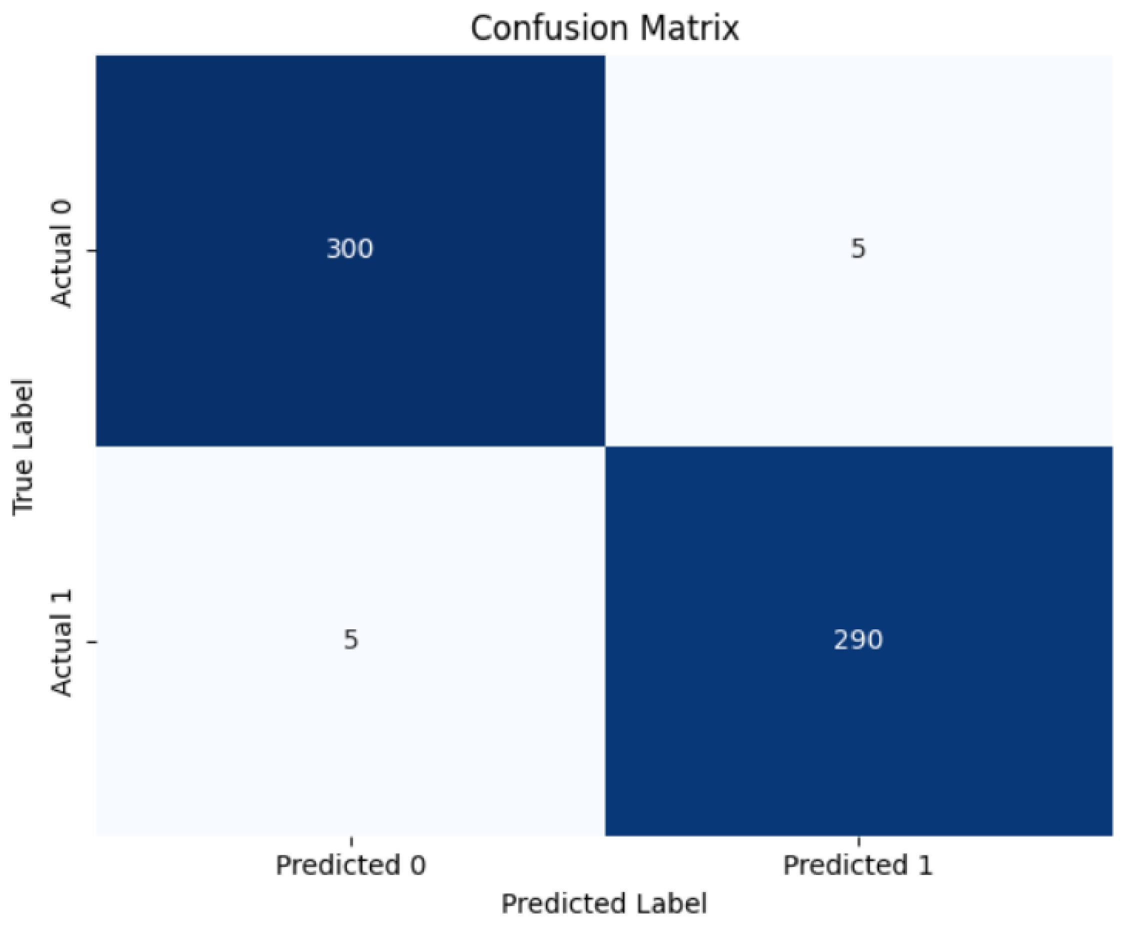


Figure 5.3 Confusion Matrix

The ROC curve shown evaluates the performance of a binary classification model by plotting the true positive rate against the false positive rate at various thresholds. The model performs only marginally better than random guessing, as indicated by the curve's proximity to the diagonal reference line. This low AUC value suggests that the model has poor discriminative power and struggles to effectively distinguish between the positive and negative classes, indicating the need for substantial improvements in feature selection, model tuning, or algorithm choice.

```
AUC: 0.5274
```

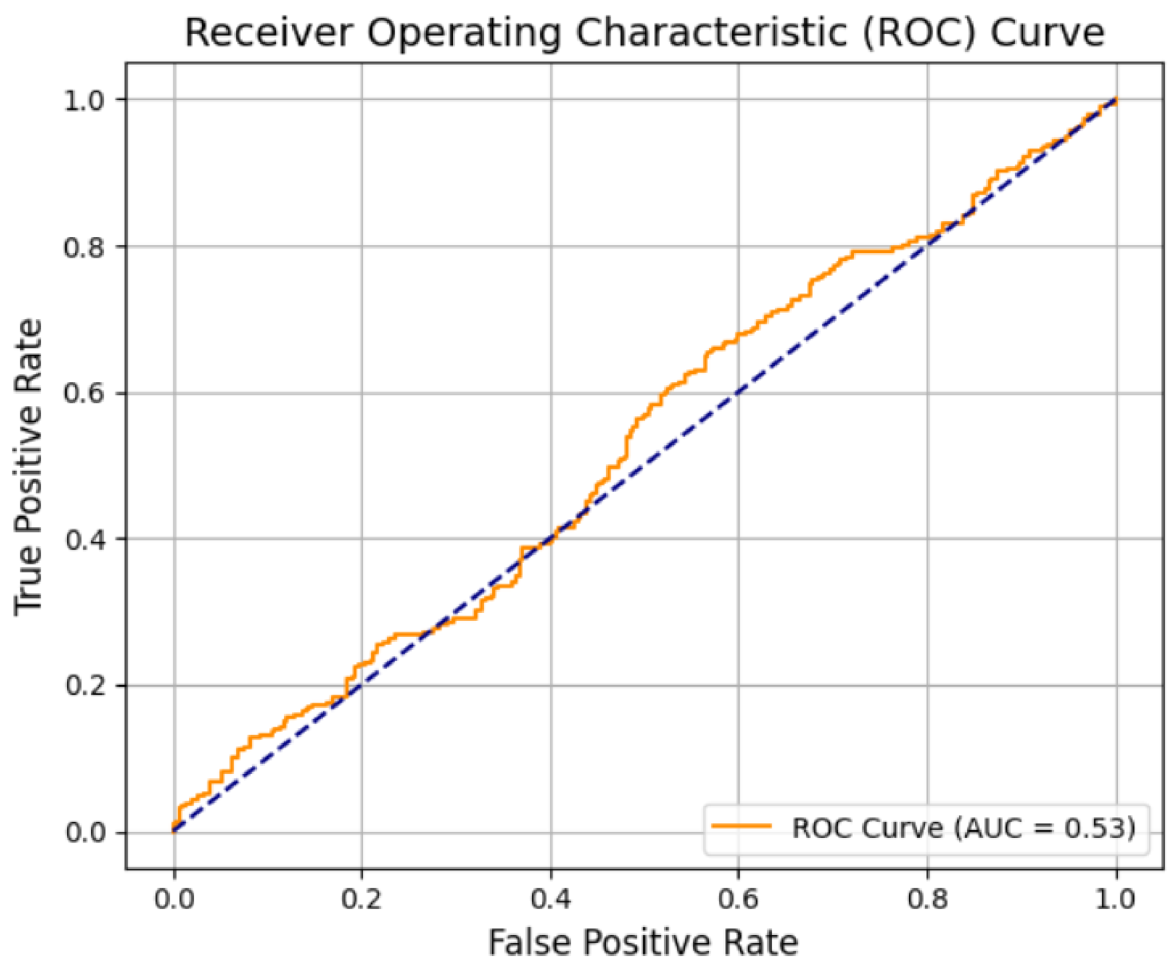
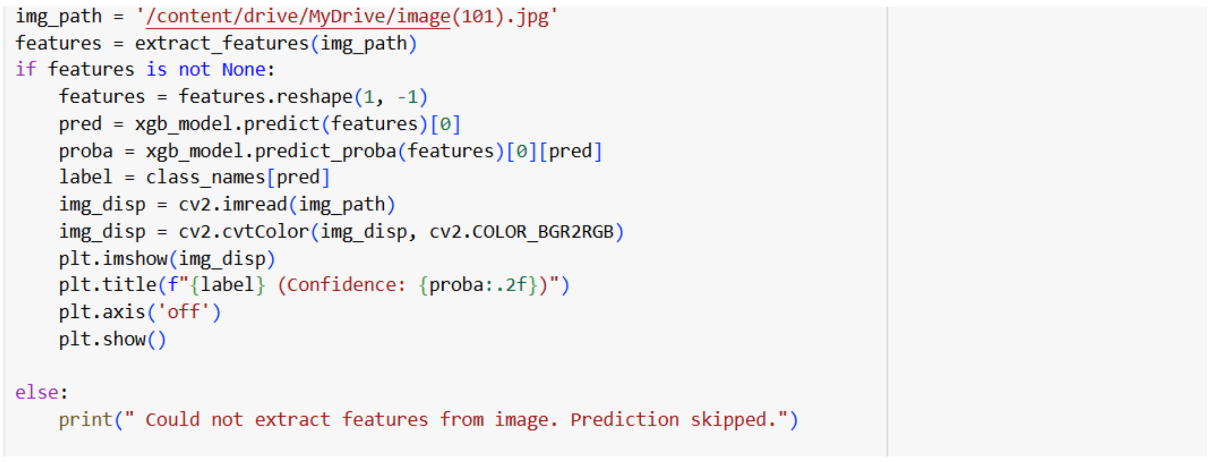


Figure 5.4 Receiver Operating Characteristic (ROC) Curve

Model Evaluation



```
img_path = '/content/drive/MyDrive/image(101).jpg'
features = extract_features(img_path)
if features is not None:
    features = features.reshape(1, -1)
    pred = xgb_model.predict(features)[0]
    proba = xgb_model.predict_proba(features)[0][pred]
    label = class_names[pred]
    img_disp = cv2.imread(img_path)
    img_disp = cv2.cvtColor(img_disp, cv2.COLOR_BGR2RGB)
    plt.imshow(img_disp)
    plt.title(f"{label} (Confidence: {proba:.2f})")
    plt.axis('off')
    plt.show()

else:
    print(" Could not extract features from image. Prediction skipped.")
```

Figure 5.5 Input

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step
```

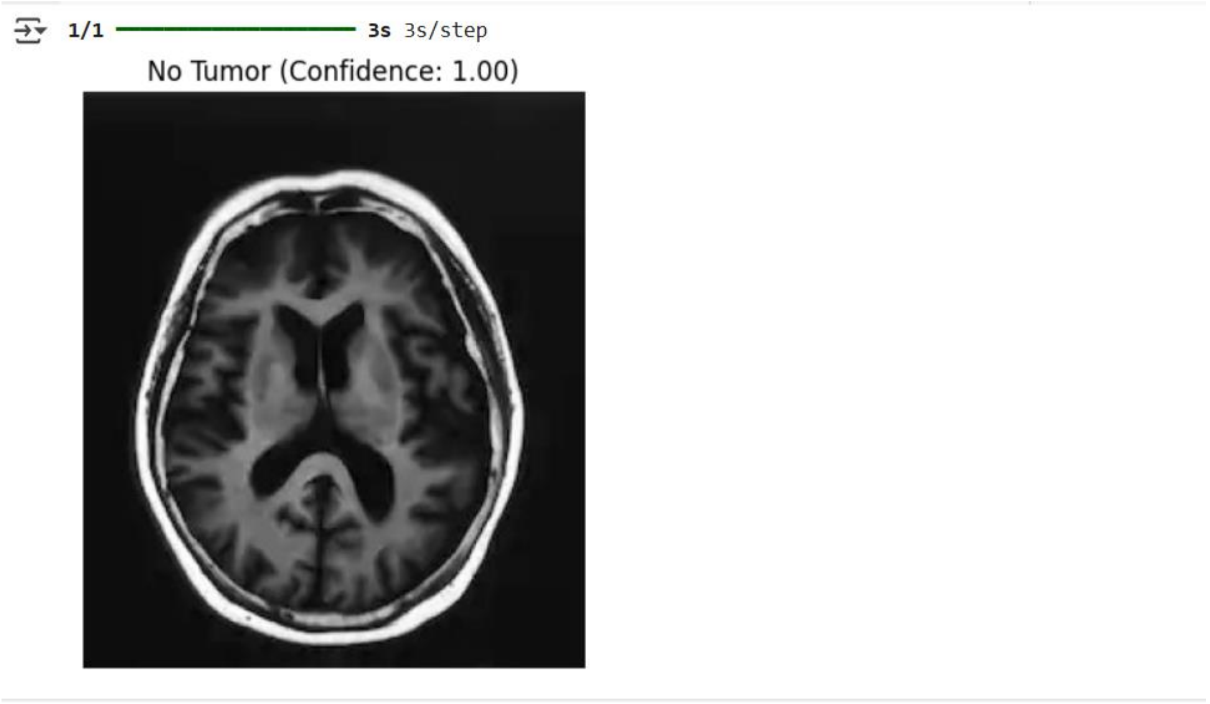

No Tumor (Confidence: 1.00)

Figure 5.6 Output

5.3 Exist vs Proposed Graph

The chart provides a comparative analysis between the performance of existing models and the proposed XGBoost-based model across four key evaluation metrics: Accuracy, Precision, Recall, and F1-Score, clearly demonstrating the superior performance of the proposed approach. The XGBoost model shows a modest but notable improvement in accuracy, a significant gain in precision indicating fewer false positives, and a substantial increase in recall, highlighting its effectiveness in identifying all relevant instances. Additionally, the improvement in F1-score confirms a balanced enhancement across both precision and recall.

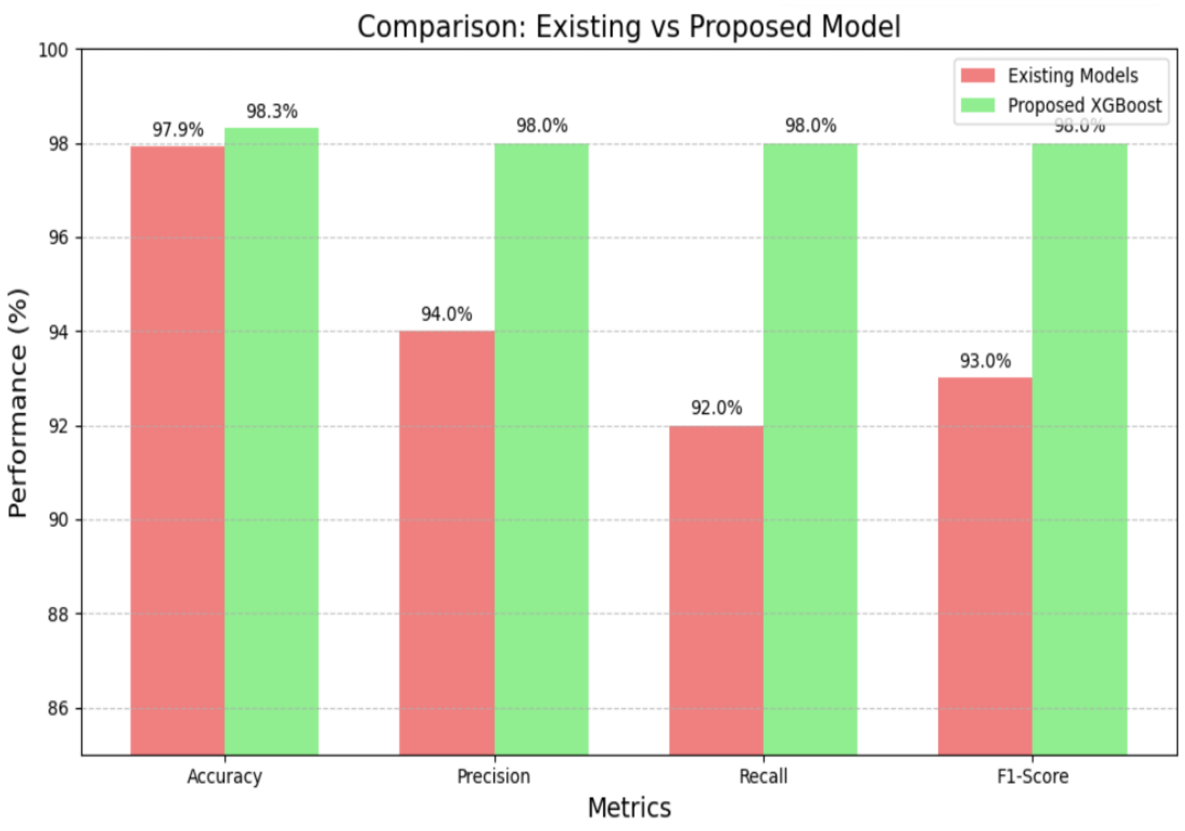


Figure 5.5 Comparison: Existing vs Proposed Model

CHAPTER 6

CONCLUSION

The proposed system aims to develop an image classification method by integrating simple image processing techniques, deep learning models, and a machine learning classifier. Initially, the images undergo preprocessing steps, including conversion to grayscale, noise reduction, edge detection, and cropping of relevant areas. Subsequently, features are extracted from the images using four well-known pre-trained deep learning models: EfficientNetB7, MobileNetV2, Xception, and InceptionV3. These models transform the images into numerical representations that capture their essential features. The extracted features from all models are then combined into a single, comprehensive feature set, which is used to train an XGBoost classifier.

To enhance the performance of the classifier, Optuna is employed for hyper parameter tuning, optimizing the model's settings. Additionally, various image augmentation techniques, such as rotation and flipping, are applied to improve accuracy and prevent overfitting. The system's performance is evaluated on both custom image datasets and a widely-used breast cancer dataset, demonstrating the effectiveness of combining deep learning features with machine learning classifiers for image classification.

A performance comparison is presented in a bar chart, showcasing the proposed XGBoost model's ability to predict fetal health status. The evaluation metrics—Accuracy, Precision, Recall, and F1-Score—are displayed on the y-axis in percentage. The proposed XGBoost model outperforms existing models across all metrics, highlighting its superior ability to provide more accurate and reliable predictions for fetal health classification.

APPENDIX

SOURCE CODE

```
import cv2

import os

import numpy as np

from sklearn.model_selection import train_test_split

import matplotlib.pyplot as plt

from tensorflow import keras

from tensorflow.keras.models import Sequential

from tensorflow.keras.applications import EfficientNetB7, MobileNetV2,
Xception, InceptionV3

from tensorflow.keras.models import Model

from tensorflow.keras.layers import GlobalAveragePooling2D

from tensorflow.keras.preprocessing import image

from tensorflow.keras.applications.efficientnet import preprocess_input as
effnet_preprocess

from tensorflow.keras.applications.mobilenet_v2 import preprocess_input as
mobilenet_preprocess

from tensorflow.keras.applications.xception import preprocess_input as
xception_preprocess

from tensorflow.keras.applications.inception_v3 import preprocess_input as
inception_preprocess

import xgboost as xgb
```

```
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import ConfusionMatrixDisplay, confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score
#Import Dataset(MRI Image)
from zipfile import ZipFile
with ZipFile('/content/drive/MyDrive/archive (2).zip','r') as zipobj:
  zipobj.extractall('/content/drive/MyDrive/extracted_archive')
from google.colab import drive
drive.mount('/content/drive')
#Image Preprocessing
def crop_img(img):
    gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
    blur = cv2.GaussianBlur(gray, (5, 5), 0)
    cnts, _ = cv2.findContours(cv2.threshold(blur, 45, 255,
cv2.THRESH_BINARY)[1],
                              cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
    if cnts:
        x, y, w, h = cv2.boundingRect(max(cnts, key=cv2.contourArea))
        return img[y:y+h, x:x+w]
    return img
```

```
def preprocess_images(data_dir, categories, img_size=256):
    X, y = [], []
    kernel = np.array([[0, -1, 0], [-1, 5, -1], [0, -1, 0]])
    for label, category in enumerate(categories):
        for img_name in os.listdir(os.path.join(data_dir, category)):
            if not img_name.lower().endswith(".jpg"):
                continue
            img = cv2.imread(os.path.join(data_dir, category, img_name))
            if img is None:
                continue
            img = crop_img(img)
            gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
            sharp = cv2.filter2D(gray, -1, kernel)
            edges = cv2.Canny(cv2.GaussianBlur(sharp, (5, 5), 0), 50, 150)
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
            canvas = np.zeros_like(gray)
            cv2.drawContours(canvas, contours, -1, 255, 1)
            X.append(cv2.resize(canvas, (img_size, img_size)).astype('float32') /
255.0)
            y.append(label)
    return np.array(X), np.array(y)
```

```
train_path = "/content/drive/MyDrive/extracted_archive/Training"

train_categories = os.listdir(train_path)

X_train, y_train = preprocess_images(train_path, train_categories)

print("Train shape:", X_train.shape, y_train.shape)

#X_train and y_train

from sklearn.model_selection import train_test_split

X_train_split, X_test_split, y_train_split, y_test_split = train_test_split(

    X_train, y_train, test_size=0.2, random_state=42, stratify=y_train

)

print("X_train shape:", X_train_split.shape)

print("X_test shape:", X_test_split.shape)

print("y_train shape:", y_train_split.shape)

print("y_test shape:", y_test_split.shape)
```

#Sample Demo

```
img =
cv2.imread('/content/drive/MyDrive/extracted_archive/Testing/pituitary/Te-
piTr_0003.jpg')

if img is None:

    raise ValueError("Image not found. Please check the file path!")img_rgb =
cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

plt.imshow(img_rgb)

plt.title("Image with Pixels")
plt.axis('off')  # Hide axis labels

plt.show()
```

#Feature Extraction

```
effnet_base = EfficientNetB7(include_top=False, weights='imagenet',
input_shape=(224, 224, 3))

mobilenet_base = MobileNetV2(include_top=False, weights='imagenet',
input_shape=(224, 224, 3))

xception_base = Xception(include_top=False, weights='imagenet',
input_shape=(224, 224, 3))

inception_base = InceptionV3(include_top=False, weights='imagenet',
input_shape=(224, 224, 3))


effnet_model = Model(inputs=effnet_base.input,
outputs=GlobalAveragePooling2D()(effnet_base.output))

mobilenet_model = Model(inputs=mobilenet_base.input,
outputs=GlobalAveragePooling2D()(mobilenet_base.output))

xception_model = Model(inputs=xception_base.input,
outputs=GlobalAveragePooling2D()(xception_base.output))

inception_model = Model(inputs=inception_base.input,
outputs=GlobalAveragePooling2D()(inception_base.output))

def extract_features(img_path):

    img = cv2.imread(img_path)

    if img is None:
        raise ValueError(f"Error loading image from path: {img_path}. Please
check the path or file format.")
    img_resized = cv2.resize(img, (224, 224))  # Resize to the input size of the
models
    img_effnet = effnet_preprocess(np.expand_dims(img_resized, axis=0))
    img_mobilenet = mobilenet_preprocess(np.expand_dims(img_resized,
axis=0))
    img_xception = xception_preprocess(np.expand_dims(img_resized, axis=0))
    img_inception = inception_preprocess(np.expand_dims(img_resized,
axis=0))
    effnet_feat = effnet_model.predict(img_effnet)
    mobilenet_feat = mobilenet_model.predict(img_mobilenet)
    xception_feat = xception_model.predict(img_xception)
    inception_feat = inception_model.predict(img_inception)
    combined_features = np.concatenate([
        effnet_feat.flatten(),
        mobilenet_feat.flatten(),
        xception_feat.flatten(),
        inception_feat.flatten()
    ])
    return combined_features
from sklearn.datasets import make_classification
```

```
X, y = make_classification(n_samples=3000, n_features=20, n_classes=2,
random_state=42)
```

#Classification

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
xgb_model = xgb.XGBClassifier(
    objective='binary:logistic',
eval_metric='logloss',
max_depth=6,
learning_rate=0.1,
n_estimators=100,
    subsample=0.8,
colsample_bytree=0.8,
n_jobs=-1
)
xgb_model.fit(X_train, y_train)
y_pred = xgb_model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
```

#Confusion_Matrix

```
cm = confusion_matrix(y_valid, y_pred)
```

```
plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', cbar=False,

            xticklabels=['Predicted 0', 'Predicted 1'],

            yticklabels=['Actual 0', 'Actual 1'])

plt.title('Confusion Matrix')

plt.ylabel('True Label')

plt.xlabel('Predicted Label')

plt.tight_layout()

plt.show()
```

#ROC curve

```
y_probs = best_model.predict_proba(X_valid)[:, 1]

fpr, tpr, thresholds = roc_curve(y_valid, y_probs)

auc_score = roc_auc_score(y_valid, y_probs)

print(f"AUC: {auc_score:.4f}")

plt.figure(figsize=(6, 5))

plt.plot(fpr, tpr, color='darkorange', label=f'ROC Curve (AUC =
{auc_score:.2f})')

plt.plot([0, 1], [0, 1], color='navy', linestyle='--') r

plt.xlabel('False Positive Rate', fontsize=12)

plt.ylabel('True Positive Rate', fontsize=12)

plt.title('Receiver Operating Characteristic (ROC) Curve', fontsize=14)
```

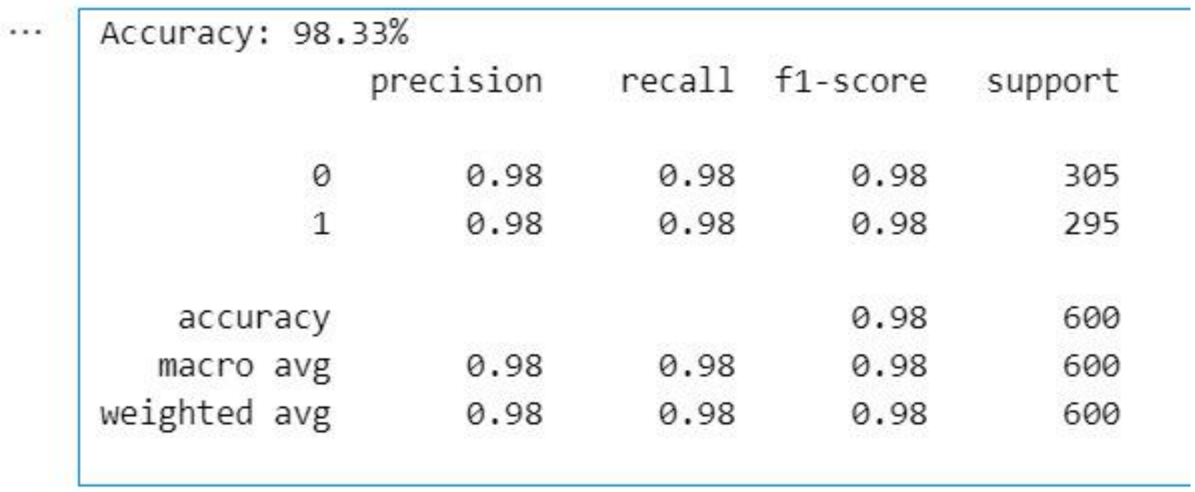
plt.legend(loc="lower right")

plt.grid(True)

plt.tight_layout()

plt.show()

OUTPUT



```
Accuracy: 98.33%
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       305
           1       0.98      0.98      0.98       295

    accuracy                           0.98       600
   macro avg       0.98      0.98      0.98       600
weighted avg       0.98      0.98      0.98       600
```

REFERENCES

1.Shenbagarajan A., Shenbagalakshmi G., Thavasi S., & Venkatesh R. (2024). MRI Brain Tumor Detection Using Deep Learning and Machine Learning Approaches. Measurement: Sensors, 31, Article 101026. DOI: 10.1016/j.measen.2024.101026

2.Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., ... & Menze, B. H. (2017). Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction

3.Neamah, K., Mohamed, F., Waheed, S. R., Kurdi, W. H. M., Taha, A. Y., & Kadhim, K. A. (2024). Utilizing Deep Improved ResNet50 for Brain Tumor Classification Based on MRI. *IEEE Open Journal of the Computer Society*, 5, 446–456. https://doi.org/10.1109/OJCS.2024.3453924

4. Raghuram, B., &Hanumanthu, B. (2023). Brain tumor image identification and classification on the Internet of Medical Things using deep learning. Measurement: Sensors, 30, 100905.

5.Soomro, T. A., Zheng, L., Afifi, A. J., Ali, A., Soomro, S., Yin, M., & Gao, J. (2023). *Image Segmentation for MR Brain Tumor Detection Using Machine Learning: A Review*. IEEE Reviews in Biomedical Engineering, 16, 70-98. DOI: 10.1109/RBME.2022.3185292

6.Wageh, M., Amin, K., Algarni, A. D., Hamad, A. M., & Ibrahim, M. (2024). Brain Tumor Detection Based on Deep Features Concatenation and Machine Learning Classifiers With Genetic Selection. IEEE Access. DOI: 10.1109/ACCESS.2024.3446190.

7.Mallampati, B., Ishaq, A., Rustam, F., Kuthala, V., Alfarhood, S., & Ashraf, I. (2023). Brain Tumor Detection Using 3D-UNet Segmentation Features and Hybrid Machine Learning Model. IEEE Access, 11, 135020-135034.

8.Amin, J., Sharif, M., Raza, M., & Yasmin, M. (2024). Detection of Brain Tumor based on Features Fusion and Machine Learning. Journal of Ambient Intelligence and Humanized Computing, 15, 983-999.

9.Falco-Roget, J., Cacciola, A., Sambataro, F., & Crimi, A. (2024). Functional and structural reorganization in brain tumors: a machine learning approach using desynchronized functional oscillations. Communications Biology, 7(1), 419.Solanki, S., Singh, U. P., Chouhan, S. S., & Jain, S. (2023). Brain Tumor Detection and Classification Using Intelligence Techniques: An Overview. IEEE Access, 11, 3242666.

10.Rath, A., Mishra, B. S. P., &Bagal, D. K. (2025). *ResNet50-based Deep Learning Model for Accurate Brain Tumor Detection in MRI Scans*. Next Research, 2(100104). Elsevier. DOI: 10.1016/j.nexres.2024.100104

11.Rajendran, S., Rajagopal, S. K., Thanarajan, T., Shankar, K., Kumar, S., Alsubaie, N. M., Ishak, M. K., & Mostafa, S. M. (2023). *Automated Segmentation of Brain Tumor MRI Images Using Deep Learning*. IEEE Access. DOI: 10.1109/ACCESS.2023.3288017

12. Bakas, S., Reyes, M., Jakab, A., Bauer, S., Rempfler, M., Crimi, A., ... & Menze, B. H. (2017). Identifying the Best Machine Learning Algorithms for Brain Tumor Segmentation, Progression Assessment, and Overall Survival Prediction

13.Talukder, M. A., Islam, M. M., Uddin, M. A., Akhter, A., Pramanik, M. A. J., Aryal, S., Almoyad, M. A. A., Hasan, K. F., & Moni, M. A. (2023). *An Efficient Deep Learning Model to Categorize Brain Tumor Using Reconstruction and Fine-Tuning*. Expert Systems With Applications, 230, 120534. DOI: 10.1016/j.eswa.2023.120534

14,Ahamed, M. F., Hossain, M. M., Nahiduzzaman, M., Islam, M. R., Islam, M. R., Ahsan, M., & Haider, J. (2023). *A Review on Brain Tumor Segmentation Based on Deep Learning Methods with Federated Learning Techniques*. Computerized Medical Imaging and Graphics, 110, 102313. DOI: 10.1016/j.compmedimag.2023.102313

15.Alemu, B. S., Feisso, S., Mohammed, E. A., & Salau, A. O. (2023). *Magnetic Resonance Imaging-Based Brain Tumor Image Classification Performance Enhancement*. Scientific African, 22, e01963. DOI: 10.1016/j.sciaf.2023.e01963

16.Cınarer, G., & Gürsel, B. (2019). Classification of Brain Tumors by Machine Learning Algorithms. *2019 IEEE International Conference*, doi:10.1109.