**BRAIN TUMOR DETECTION USING VGG-16 TRANSFER LEARNING**

**A MINI PROJECT REPORT**

*Submitted by*

|  |  |
| --- | --- |
| **NAVEEN KUMAR A**  **UDHAYAPRAKASH T**  **VEERALOKESH B** | **830120104305**  **830120104314**  **830120104315** |

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

****

## **GOVERNMENT COLLEGE OF ENGINEERING SRIRANGAM**

## **ANNA UNIVERSITY: CHENNAI 600 025**

##### 

##### **MAY 2023**

# **ANNA UNIVERSITY: CHENNAI 600 025**

# **BONAFIDE CERTIFICATE**

Certified that this project report **“BRAIN TUMOR DETECTION USING VGG-16 TRANSFER LEARNING”** is the bonafide work of **“NAVEEN KUMAR A (830120104305), UDHAYAPRAKASH T (830120104314),** **VEERALOKESH B (830120104315)”** who carried out the project work under my supervision.

|  |  |
| --- | --- |
| **SIGNATURE** | **SIGNATURE** |
| **Prof. P. VANITHA MUTHU, M.Tech.,**  **HEAD OF THE DEPARTMENT**  Dept of Computer Science & Engineering,  Government College of Engg. Srirangam,  Tiruchirappalli-620012. | **Mr. H. SHEIK MOHIDEEN, M.E.,**  **SUPERVISOR**  Assistant Professor,  Dept of Computer Science & Engineering,  Government College of Engg. Srirangam,  Tiruchirappalli-620012. |

Submitted for the Mini Project (CS6811) viva-voce examination held

on

**INTERNAL EXAMINER EXTERNAL EXAMINER**

**ACKNOWLEDGEMENT**

We first of all thank GOD ALMIGHTY for giving us a golden opportunity to express our individual and technical skills in the field of Computer Science and Engineering.

We express our gratitude to our beloved Principal,

**Dr. V. M. SHANTHI, M.E., Ph.D.,** for giving us a chance to complete our higher education in one of the reputed government institutions running under her magnificent leadership.

We are extremely thankful to our Head of the Department,

**Prof. P. VANITHA MUTHU, M.Tech.,** Associate Professor,Department of Computer Science and Engineering, for her support, motivation, inspiration and tireless encouragement.

We are immensely pleased to thank our project guide, **Mr. H. SHEIK MOHIDEEN, M.E.,** Assistant Professor, Department of Computer Science and Engineering for his valuable comments and suggestions.

We also express our thanks to our Project Coordinator,

**Dr. S. ANNIE JOICE, M.E., M.B.A., Ph.D.,** Assistant Professor, Department of Computer Science and Engineering, for ideas given and for her constant inspiration throughout our project period.

It is a great opportunity to express our sincere thanks to our parents, friends and all the people who have contributed to the successful completion of our project work through their support, encouragement and guidance.

**NAVEEN KUMAR A**

**UDHAYAPRAKASH T**

**VEERALOKESH B**

**ABSTRACT**

A brain tumor is a distorted tissue wherein cells replicate rapidly and indefinitely, with no control over tumor growth. Deep learning has been argued to have the potential to overcome the challenges associated with detecting and intervening in brain tumors. It is well established that the segmentation method can be used to remove abnormal tumor regions from the brain, as this is one of the advanced technological classification and detection tools. In the case of brain tumors, early disease detection can be achieved effectively using reliable advanced A.I. and Neural Network classification algorithms. This study aimed to critically analyze the proposed literature solutions, use the Visual Geometry Group (VGG 16) for discovering brain tumors, implement a convolutional neural network (CNN) model framework, and set parameters to train the model for this challenge. VGG is used as one of the highest-performing CNN models because of its simplicity. Furthermore, the study developed an effective approach to detect brain tumors using MRI to aid in making quick, efficient, and precise decisions. Faster CNN used the VGG 16 architecture as a primary network to generate convolutional feature maps, then classified these to yield tumor region suggestions. The prediction accuracy was used to assess performance. Our suggested methodology was evaluated on a dataset for brain tumor diagnosis using MR images comprising 253 MRI brain images, with 155 showing tumors. Our approach could identify brain tumors in MR images. In the testing data, the algorithm outperformed the current conventional approaches for detecting brain tumors (Precision = 96%, 98.15%, 98.41% and F1-score = 91.78%, 92.6% and 91.29% respectively) and achieved an excellent accuracy of CNN 96%, VGG 16 98.5% and Ensemble Model 98.14%. The study also presents future recommendations regarding the proposed research work.

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **CHAPTER NO** | **TITLE** | **PAGE NO** |
|  | **ABSTRACT** | iv |
|  | **LIST OF FIGURES** | ix |
|  | **LIST OF TABLES** | x |
|  | **LIST OF ABBREVIATIONS** | xi |
| **1** | **INTRODUCTION** | 1 |
|  | * 1. OVERVIEW OF BRAIN AND BRAIN TUMOR | 1 |
|  | * 1. MAGNETIC RESONANCE IMAGING (MRI**)** | 2 |
|  | * 1. PROBLEM STATEMENT | 4 |
|  | * 1. MOTIVATION | 5 |
| **2** | **LITERATURE SURVEY** | 7 |
|  | 2.1 IMAGE ANALYSIS FOR MRI BASED BRAIN TUMOR DETECTION AND FEATURE EXTRACTION USING BIOLOGICALLY INSPIRED BWT AND SVM | 7 |
|  | 2.2 A GENETIC ALGORITHM BASED FEATURE SELECTION FOR CLASSIFICATION OF BRAIN MRI SCAN IMAGES USING RANDOM FOREST CLASSIFIER | 8 |
|  | 2.3 MRI BRAIN IMAGE SEGMENTATION BY USING A DEEP SPECTRUM IMAGE  TRANSLATION NETWORK | 9 |
| **3** | **SYSTEM ANALYSIS** | 10 |
|  | 3.1 EXISTING SYSTEM | 10 |
|  | 3.2 PROPOSED SYSTEM | 12 |
| **4** | **SYSTEM REQUIREMENTS** | 14 |
|  | 4.1 HARDWARE REQUIREMENTS | 14 |
|  | 4.2 SOFTWARE REQUIREMENTS | 14 |
| **5** | **SYSTEM DESIGN** | 15 |
|  | 5.1 SYSTEM ARCHITECTURE | 15 |
|  | 5.2 DATA FLOW DIAGRAM | 17 |
|  | 5.3 UML DIAGRAMS | 19 |
|  | 5.3.1 Use Case Diagram | 19 |
|  | 5.3.2 Class Diagram | 20 |
|  | 5.3.3 Sequence Diagram | 21 |
|  | 5.3.4 Component Diagram | 22 |
|  | 5.3.5 Deployment Diagram | 23 |
| **6** | **SYSTEM IMPLEMENTATION** | 24 |
|  | 6.1 TECHNOLOGY STACK | 24 |
|  | 6.1.1 Python | 24 |
|  | 6.1.2 TensorFlow and Keras | 24 |
|  | 6.1.3 NumPy | 24 |
|  | 6.1.4 OpenCV | 25 |
|  | 6.1.5 Flask | 25 |
|  | 6.1.6 HTML, CSS, JavaScript | 25 |
|  | 6.2 FEATURES OF PROPOSED SYSTEM | 26 |
| **7** | **CHALLENGES AND APPLICATIONS** | 28 |
| **8** | **CONCLUSION AND FUTURE WORK** | 30 |
|  | **APPENDICES** | 32 |
|  | **APPENDIX 1 SCREEN SHOT** | 32 |
|  | **APPENDIX 2 SOURCE CODE** | 38 |
|  | **REFERENCES** | 53 |

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO.** | **TITLE** | **PAGE NO** |
| 1.1 | Basic Structure of human brain | 2 |
| 1.2 | T1, T2 and Flair image | 3 |
| 1.3 | Graph of TE and TR | 3 |
| 3.1.1 | SVM working model | 10 |
| 3.2.1 | Working of VGG16 model | 12 |
| 3.2.2 | VGG16 layered architecture | 13 |
| 5.1.1 | System architecture | 16 |
| 5.2.1 | Data Flow Diagram level 0 | 17 |
| 5.2.2 | Data Flow Diagram level 1 | 18 |
| 5.3.1 | Use case Diagram | 19 |
| 5.3.2 | Class diagram | 20 |
| 5.3.3 | Sequence diagram | 21 |
| 5.3.4 | Component diagram | 22 |
| 5.3.5 | Deployment diagram | 23 |
| A.1.1 | Home Screen | 32 |
| A.1.2 | Selecting Image to Detect Tumor | 33 |
| A.1.3 | Identifies No Tumor in Brain | 34 |
| A.1.4 | Identifies Tumor in Brain | 35 |
| A.1.5 | Training Dataset MRI scan images | 36 |
| A.1.6 | Training Dataset image heights | 37 |
| A.1.7 | Training Dataset image widths | 37 |
| A.1.8 | Training VGG -16 Layers | 38 |
| A.1.9 | Epoch Vs Cross Entropy loss | 39 |
| A.1.10 | Epoch Vs Accuracy | 39 |
| A.1.11 | Process of Training with Data | 40 |
| A.1.12 | Predicting Tumor in MRI with accuracy | 41 |

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **TABLE NO.** | **TITLE** | **PAGE**  **NO** |
| 1.2.1 | TR and TE time | 4 |

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| MRI  TR  TE  VGG 16  ReLU  LRN  SVM  CNN  HTML  CSS  OpenCV  NumPy  EDA | Magnetic Resonance Imaging  Time Repetition  Echo Time  Visual Geometry Group  Rectified Linear Unit  Local Response Normalization  Support Vector Machine  Convolutional Neural Network  Hyper Text Transfer Protocol  Cascading Style Sheets  Open-Source Computer Vision  Numerical Python  Exploratory data analysis |

**CHAPTER 1**

**INTRODUCTION**

The brain is a critical organ, and brain tumors occur when cells grow uncontrollably, disrupting brain function. Detecting tumors manually through MRI images is time-consuming and often inaccurate. Early detection is crucial due to the seriousness of brain cancer. This project aims to use computer-based procedures, specifically Convolutional Neural Networks, for accurate brain tumor detection and classification. Image processing techniques, such as segmentation, enhancement, and feature extraction, are employed to analyze MRI images. The process involves four stages: image pre-processing, segmentation, feature extraction, and classification. By combining image processing and neural network techniques, the system enhances the performance of detecting and classifying brain tumors in MRI images.

**1.1 OVERVIEW OF BRAIN AND BRAIN TUMOR**

The human brain is the central component of the nervous system and is situated within the skull. Its primary function is to regulate and coordinate the various functions of the body. Additionally, the brain enables humans to perceive and adapt to different environmental conditions. It serves as the organ responsible for initiating actions and facilitating the expression of thoughts and emotions. Understanding the structure of the brain is essential for grasping its fundamental aspects.

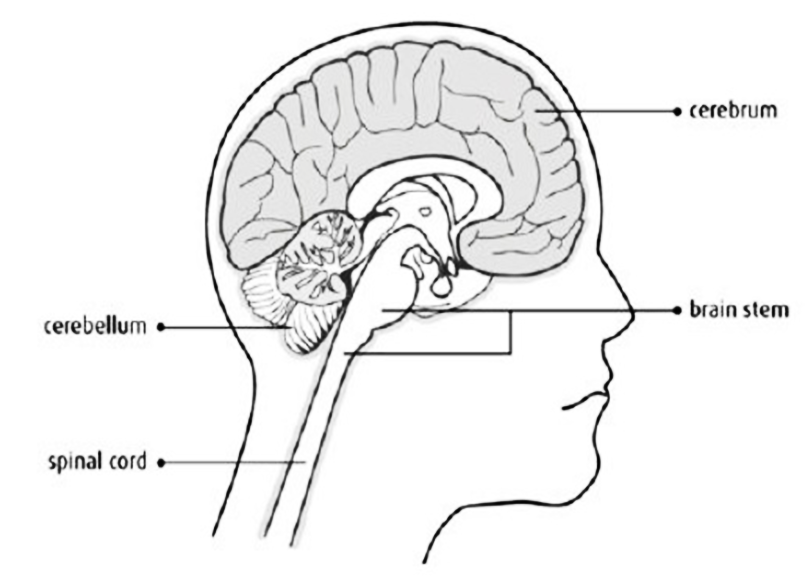


Fig 1.1.1: Basic Structure of human brain

Brain tumors are categorized into primary (benign) and secondary (malignant) types. Primary tumors, such as gliomas, grow slowly in the brain from non-neuronal cells called astrocytes. While they are less aggressive, they exert pressure on the brain, affecting its function. Secondary tumors, on the other hand, are highly malignant and spread quickly from other parts of the body. They originate from metastatic cancer cells, often originating in organs like the lungs, kidneys, or bladder.

**1.2 MAGNETIC RESONANCE IMAGING (MRI)**

Raymond V. Damadian invented the first magnetic image in 1969. The first MRI images for the human body were developed in 1977, revolutionizing medical imaging. MRI provides detailed visualization of the brain and different body tissues. It offers superior image quality compared to X-ray and CT scans. MRI is a valuable technique for detecting brain tumors, using various imaging methods like T1 weighted, T2 weighted, and FLAIR weighted images.

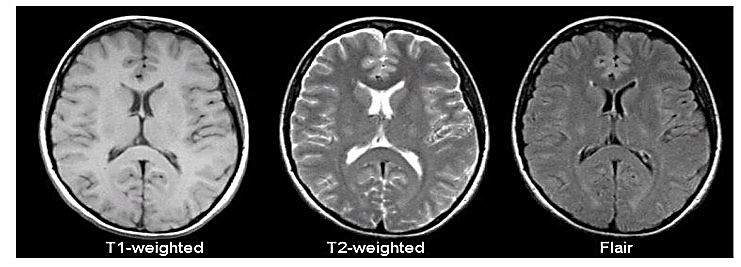


Fig 1.2.1: T1, T2 and Flair image

The common MRI sequences are T1 weighted and T2 weighted. T1 weighted shows only fat as bright, while T2 weighted shows fat and water as bright. T1 weighted has a short repetition time (TR), while T2 weighted has long TE and TR. TE and TR are pulse sequence parameters measured in milliseconds. TE represents the time from the RF pulse center to the echo center, and TR is the time between repeating pulse and echo series.

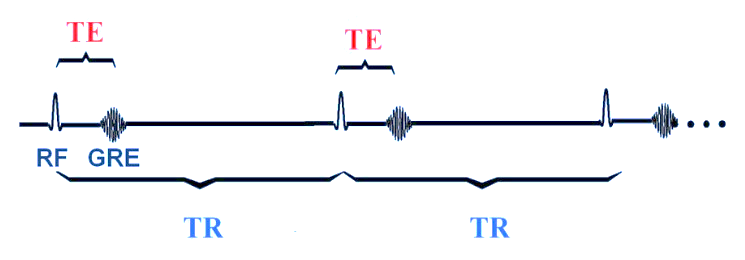


Fig 1.2.2: Graph of TE and TR

The third commonly used sequence in the FLAIR. The Flair sequence is almost same as T2-weighted image. The only difference is TE and TR time are very long. Their approximate TR and TE times are shown in table.

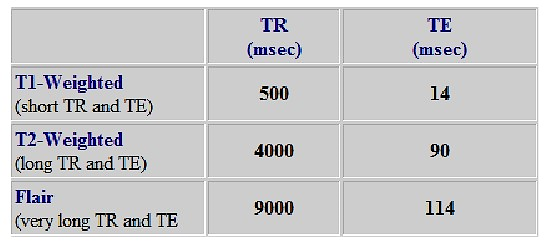


Table 1.2.1: TR and TE time

**1.3 PROBLEM STATEMENT**

Brain tumors are a significant health concern and early detection plays a crucial role in effective treatment and patient outcomes. Traditional diagnostic methods rely on manual interpretation by radiologists, which can be time-consuming and subject to human error. The advancement of deep learning techniques, particularly convolutional neural networks (CNNs), has shown great promise in automating the detection process and achieving high accuracy.

**1.4 MOTIVATION**

* **Improved Diagnostic Accuracy**

Deep learning techniques like the VGG16 architecture enhance brain tumor detection by improving diagnostic accuracy, reducing misdiagnosis, and enabling early intervention.

* **Time-Efficiency**

Implementing an automated brain tumor detection system using VGG16 streamlines the diagnostic process, providing faster results and reducing the burden on healthcare professionals. Quick and reliable detection enables medical experts to prioritize treatment planning and patient care.

* **Scalability and Accessibility**

To meet the growing demand for medical imaging, an automated brain tumor detection system can address the shortage of skilled radiologists in remote or under-served areas. By leveraging the VGG16 architecture's reliability and performance, this scalable solution can provide accessible diagnostic support to healthcare facilities with limited resources.

* **Potential for Early Intervention**

By providing customers with an easy-to-use and convenient online platform to purchase parts and accessories, the automobile workshop can improve the overall customer experience. This can lead to increased customer satisfaction and loyalty.

* **Competitive Advantage**

Early detection of brain tumors is vital for successful treatment. Accurate identification at an early stage allows for prompt interventions and improved patient outcomes. Utilizing VGG16 can aid in early tumor identification, enhancing survival rates and prognosis.

* **Research and Development**

Brain tumor detection using deep learning models like VGG16 drives research and development in medical imaging analysis. It advances AI in healthcare and inspires further studies on optimizing CNN architectures, exploring new datasets, and developing sophisticated tumor detection models.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 TITLE: IMAGE ANALYSIS FOR MRI BASED BRAIN TUMOR DETECTION AND FEATURE EXTRACTION USING BIOLOGICALLY INSPIRED BWT AND SVM**

**AUTHOR: NILESH BHASKARRAO BAHADURE, ARUN KUMAR RAY, AND HAR PAL THETHI**

**YEAR: 2017**

**OVERVIEW:**

In this paper using MRI images of the brain, we segmented brain tissues  
into normal tissues such as white matter, gray matter, cerebrospinal fluid  
(background), and tumor-infected tissues. We used pre-processing to improve the  
signal-to-noise ratio and to eliminate the effect of unwanted noise. We can use the  
skull stripping algorithm it is based on threshold technique for improve the skull  
stripping performance.

**ADVANTAGE**

* The article provides insights on utilizing SVM classifiers for brain tumor detection, offering a potential framework for accurate classification.

**DISADVANTAGE**

* The article lacks comprehensive validation and may require further research to ensure the reliability and generalizability of the proposed approach.

**2.2 TITLE: A GENETIC ALGORITHM BASED FEATURE SELECTION FOR CLASSIFICATION OF BRAIN MRI SCAN IMAGES USING**

**RANDOM FOREST CLASSIFIER**

**AUTHOR: DR. S. MARY JOANS AND J. SANDHIYA**

**YEAR: 2017**

**OVERVIEW:**

This article presents a Genetic Algorithm-based feature selection process for identifying informative features from MRI images. These selected features are utilized by a Random Forest classifier, resulting in accurate brain tumor classification. The study showcases successful evaluations and results, emphasizing the effectiveness of this approach in achieving high classification accuracy.

**ADVANTAGE**

* Genetic Algorithm-based feature selection enhances brain tumor classification accuracy by identifying informative features, leading to more precise diagnoses and improved treatment planning.

**DISADVANTAGE**

* The Genetic Algorithm-based feature selection process can be computationally intensive, requiring substantial computational resources and time to optimize feature selection.

**2.3 TITLE: MRI BRAIN IMAGE SEGMENTATION BY USING A DEEP SPECTRUM IMAGE TRANSLATION NETWORK**

**AUTHOR: SRINIVASARAO GAJULA, V RAJESH**

**YEAR: 2021**

**OVERVIEW:**

This article applies VGG-19 classifiers for accurate brain tumor detection. Using deep convolutional neural networks, VGG-19 models are trained on brain tumor datasets to distinguish tumor and non-tumor regions effectively. The deep layers capture complex patterns and spatial information, ensuring reliable detection.

**ADVANTAGE**

* Accurate brain tumor detection with VGG-19 classifiers due to deep architecture and feature extraction.

**DISADVANTAGE**

* It has more layers and need resource-intensive training with large dataset.

**CHAPTER 3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

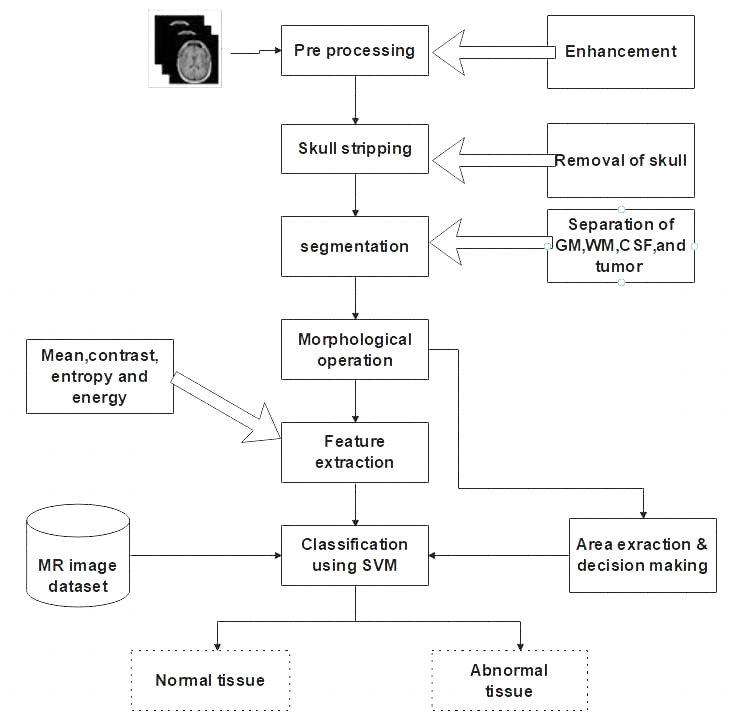
****

Fig 3.1.1: SVM working model

In the first stage, computer-based procedures utilizing Artificial Neural Network Algorithm and Support Vector Machine (SVM) are employed for tumor detection and classification in MRI images. The second stage involves image processing techniques such as histogram equalization, segmentation, enhancement, morphological operations, and feature extraction for brain tumor detection in cancer-affected MRI images. This work presents an automatic method to increase accuracy and reduce diagnosis time by utilizing SVM and image processing techniques.

**Image Pre-processing:** MRI scanned images are pre-processed using a high-pass filter to remove noise.

**Segmentation:** Region growing technique is used for image segmentation by selecting seed points.

**Morphological operations:** Dilation and erosion operations extract boundary areas in binary images.

**Feature Extraction:** Edge detection and higher-level feature extraction (shape, texture, colour and contrast) are performed.

**Connected Component Labelling:** Connected components are identified and assigned unique region labels.

**Tumor Identification:** A knowledge base is created from brain MRI datasets, and SVM-based classification is used for tumor identification.

**3.2 PROPOSED SYSTEM**

The proposed system includes five modules: Dataset, Pre-processing, Data Splitting, CNN Model Training, and Classification. It involves collecting MRI images, preprocessing them, and splitting the data into training (80%) and testing (20%) sets. The deep neural network is trained using the CNN Model Training module, and the Classification module determines tumor presence. This system enables effective brain tumor detection using deep learning techniques.

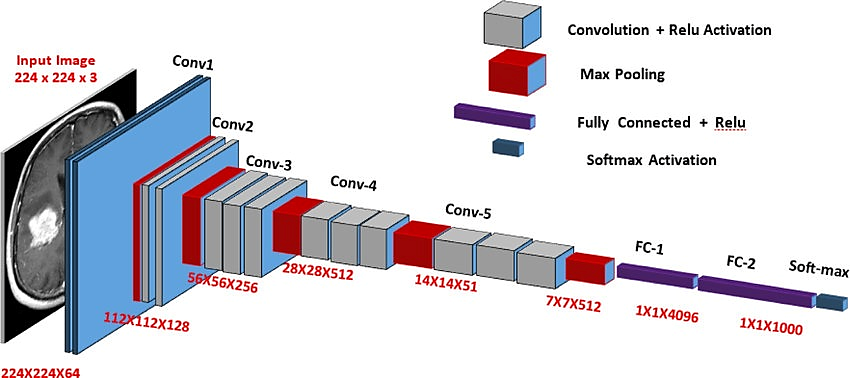
****

Fig 3.2.1: Working of VGG16 model

Transfer learning is a knowledge-sharing method that reduces data size, training time, and computational costs in deep learning. It allows the transfer of learning from a pre-trained model to a new one. It has been applied in tumor classification, software defect prediction, activity recognition, and sentiment classification. This study compares the performance of the proposed Deep CNN model with the popular transfer learning approach, VGG16.

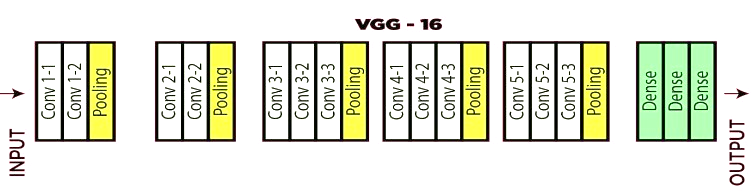
****

Fig 3.2.2: VGG16 layered architecture.

VGG16 is a convolutional neural network with fixed input size of 224x224 RGB image. It uses small 3x3 filters to capture spatial information and 1x1 convolution filters for linear transformations of input channels. The stride is 1 pixel and spatial padding is 1 pixel for 3x3 convolution layers. Max-pooling is performed with a 2x2 window and stride of 2. Three fully connected layers follow the convolutional layers, with the first two having 4096 channels each and the third performing 1000-way classification. ReLU is used as the activation function in hidden layers. Notably, the networks do not include Local Response Normalization (LRN) due to its limited impact on performance and increased resource requirements.

**CHAPTER 4**

**SYSTEM REQUIREMENTS**

**4.1 HARDWARE REQUIREMENTS**

* RAM : 4 GB RAM and more
* Processor : Intel 8th gen i5 or higher
* Hard Disk : 8GB and more

**4.2 SOFTWARE REQUIREMENTS**

* Operating System : Windows 10
* IDE : Jupiter notebook, Visual Studio Code
* FRONTEND : Html, CSS, JavaScript, Bootstrap
* BACKEND : Python, Flask
* LIBRARIES : TensorFlow, Keras, NumPy, OpenCV

**CHAPTER 5**

**SYSTEM DESIGN**

**5.1 SYSTEM ARCHITECTURE**

System architecture plays a critical role in ensuring the efficiency and effectiveness of a computer-based system. It enables proper allocation of resources, defines the system's boundaries, and determines the overall system behavior. The architecture helps in identifying the key components and their interconnections, facilitating system analysis, design, and integration. It allows for modularity, enabling the system to be developed and maintained in a structured manner. Additionally, the system architecture supports scalability, allowing the system to handle increased workloads and adapt to changing requirements. It also aids in system security by implementing appropriate access controls and data protection measures. Furthermore, the architecture documentation serves as a valuable reference for system stakeholders, including developers, designers, testers, and maintainers. Overall, a well-designed system architecture is essential for building robust, reliable, and flexible computer-based systems.

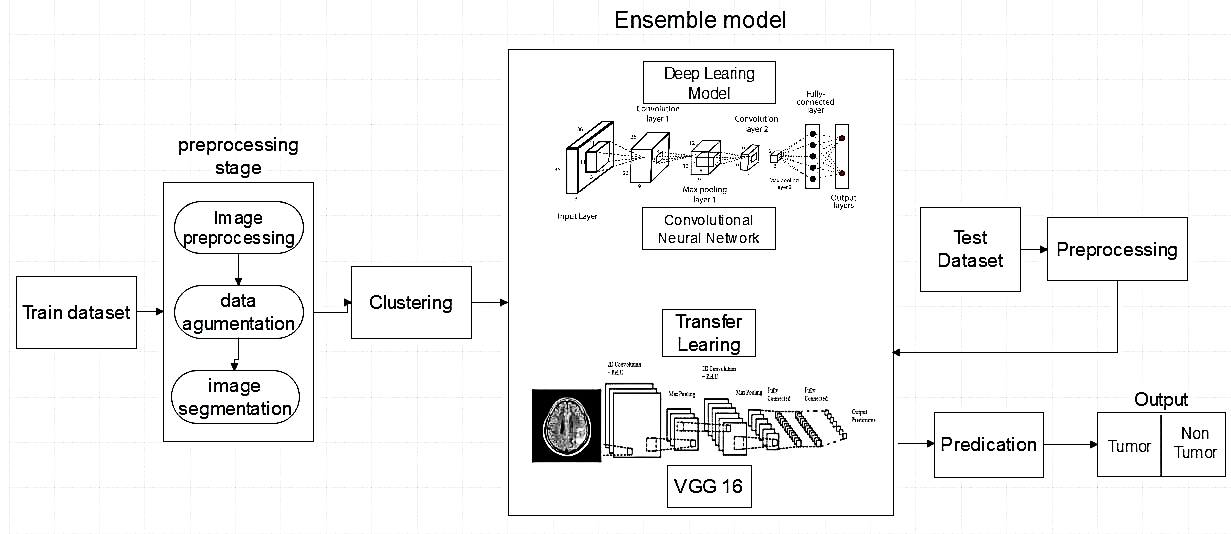
****

Fig 5.1.1: System architecture diagram

The system architecture for VGG16 and CNN-based brain tumor detection consists of input MRI images that are pre-processed and fed into the VGG16 architecture. The VGG16 architecture acts as a feature extractor, capturing important patterns and information from the images. The extracted features are then passed through a classification layer to determine whether the images contain a brain tumor or not. The system is trained using labeled data and fine-tuning techniques, and its performance is evaluated using metrics such as accuracy, precision, recall, and F1 score. By leveraging the deep learning capabilities of VGG16 and CNN architectures, this system enables accurate and effective detection of brain tumors in MRI images.

**5.2 DATA FLOW DIAGRAM**

A data flow diagram (DFD) is a graphical representation of the flow of data through a system. It shows how data is input, processed, and output by a system and how it interacts with external entities. DFDs consist of four main components: entities, processes, data flows, and data stores. Entities are external actors that interact with the system, while processes represent the actions taken on the data.

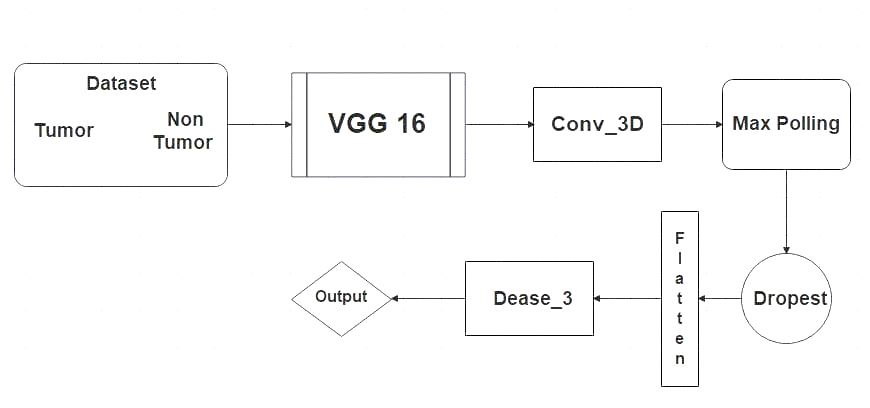


Fig 5.2.1: Data flow diagram level 0

Data flows indicate the movement of data between the entities, processes, and data stores, and data stores represent the repositories where data is stored. DFDs are useful in analyzing and designing systems, as they provide a clear and concise representation of the system's data flow, helping stakeholders to understand the system's behavior and identify areas for improvement.

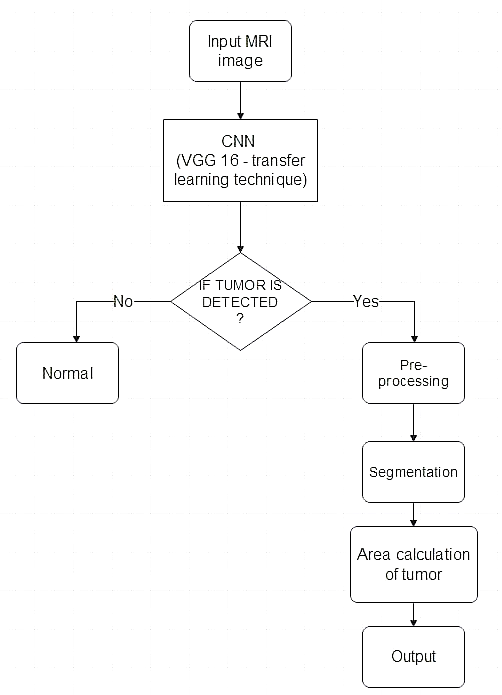


Fig 5.2.2: Data flow diagram level 1

**5.3 UML DIAGRAMS**

**5.3.1 Use Case Diagram**

Diagram

Description automatically generated

Fig 5.3.1: Use case diagram

A use case diagram is a type of diagram in the Unified Modeling Language (UML) that represents the interactions between actors and a system. It is used to describe the functionality of a system and its interactions with outside entities. The diagram consists of actors, use cases, and relationships between them. Actors are external entities, such as users or systems, that interact with the system, while use cases represent the functionality of the system. Relationships between actors and use cases indicate the interactions between them. Use case diagrams are helpful in identifying system requirements and validating them against user needs and can be used in software development and other fields.

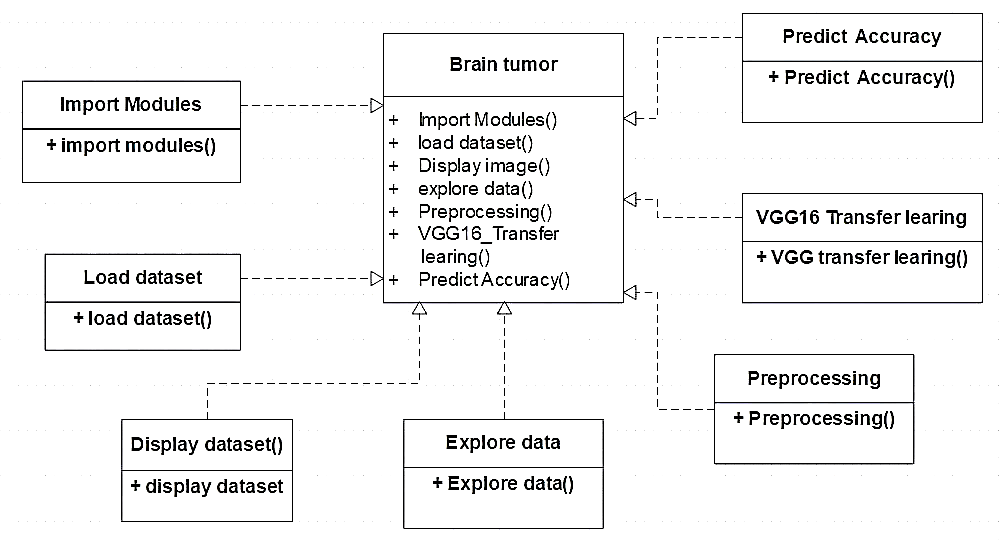
**5.3.2 Class Diagram**

Fig 5.3.2: Class diagram

A class diagram in UML represents system structure with classes, attributes, methods, and relationships. Classes are blueprints for objects with shared attributes and behaviors. Attributes represent state, methods represent behavior, and relationships include inheritance, composition, and association. Class diagrams aid design, analysis, and team collaboration.

**5.3.3 Sequence Diagram**

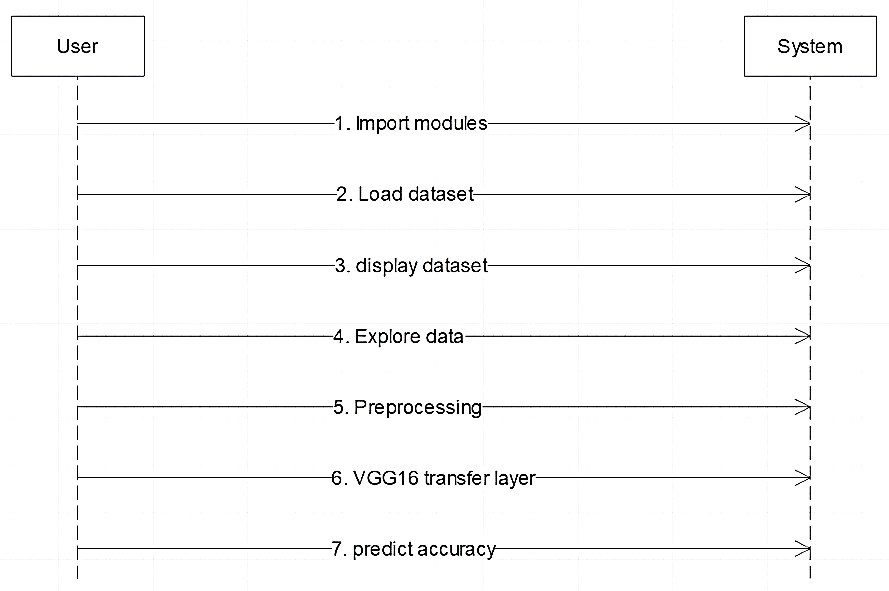


Fig 5.3.3: Sequence diagram

A sequence diagram in UML depicts object/component interactions over time. It shows message flow, lifelines, and activation periods. Lifelines represent object/component lifespan, while activation periods indicate activity intervals. Sequence diagrams aid system design and analysis, providing a visual representation of behavior and facilitating understanding of event sequences. They are widely used in software development for modeling system behavior and validating design against requirements.

**5.3.4 Component Diagram**

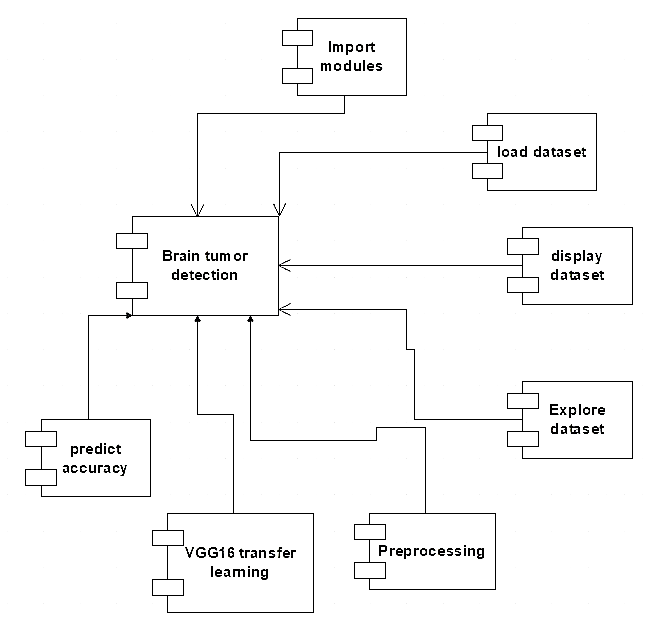


Fig 5.3.4: Component diagram

A component diagram in UML illustrates the structural relationships between system components. Components are modular, reusable parts with defined interfaces. The diagram showcases component connections, interactions, and dependencies. Components can be deployed across platforms and systems. These diagrams aid system design and analysis, visually representing components and their interactions to enhance architectural understanding and identify potential improvements.

**5.3.5 Deployment Diagram**

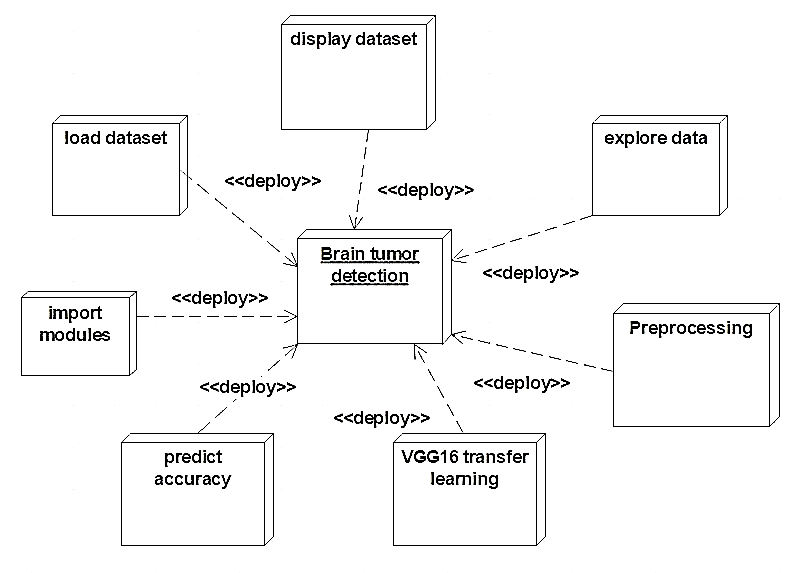


Fig 5.3.5: Deployment diagram

A UML deployment diagram visually represents the physical distribution and interaction of system components across nodes, illustrating component deployment and communication. It provides a visual representation of how system components are deployed and interact within a distributed environment.

**CHAPTER 6**

**SYSTEM IMPLEMENTATION**

**6.1 TECHNOLOGY STACK**

**6.1.1 Python**

Python is a widely used programming language for machine learning and deep learning projects. It offers a rich ecosystem of libraries and frameworks that make it suitable for developing the brain tumor detection project.

**6.1.2 TensorFlow and Keras**

TensorFlow and Keras are essential libraries for deep learning in Python. TensorFlow provides a comprehensive framework for building and training neural networks, while Keras offers a user-friendly interface for model development. These libraries are widely used in various applications, including image recognition, due to their extensive capabilities and community support.

**6.1.3 NumPy**

NumPy is essential for image recognition tasks, enabling efficient handling and manipulation of image data, facilitating operations like resizing, cropping, and normalization. It also supports feature extraction and processing of image arrays, making it a fundamental library in image recognition algorithms.

**6.1.4 OpenCV**

OpenCV is powerful library for image preprocessing, manipulation, and feature extraction. It is utilized for tasks such as image resizing, histogram equalization, contour detection, and image enhancement, enhancing the accuracy and effectiveness of the tumor detection process.

**6.1.5 Flask**

Flask provides the necessary framework for handling user requests, rendering HTML templates, and managing the communication between the front-end and back-end components of the system. It allows for the development of a dynamic and interactive web application where users can upload their MRI images, trigger the tumor detection process, and view the results.

**6.1.6 HTML, CSS and JavaScript**

Together, HTML, CSS, and JS form the core technologies for building and styling web pages, as well as creating interactive and engaging user experiences on the web.

**6.2 FEATURES OF PROPOSED SYSTEM**

**6.2.1 Data pre-processing**

It includes pre-processing techniques such as image resizing, noise removal, and normalization to enhance the quality of input MRI images.

**6.2.2 Model architecture**

It may utilize the VGG16 convolutional neural network (CNN) architecture or other deep learning models for feature extraction and classification of brain tumor images.

**6.2.3 Transfer learning**

It incorporates transfer learning, where a pre-trained model (such as VGG16) is used as a starting point and fine-tuned on the specific brain tumor detection task.

**6.2.4 Evaluation metrics**

It may use evaluation metrics like accuracy, precision, recall, and F1-score to assess the performance of the model in correctly detecting brain tumors.

**6.2.5 User interface**

It might have a user-friendly interface that allows users to upload MRI images, process them using the trained model, and visualize the results.

**6.2.6 Deployment**

It provides options for deploying the trained model, such as integrating it into a web application or creating a standalone software tool.

**CHAPTER 7**

**CHALLENGES AND APPLICATION**

The use of VGG16 CNN with transfer learning for brain tumor detection faces several challenges. Firstly, acquiring a large and diverse dataset of labeled brain MRI images with tumor and non-tumor samples can be difficult. The availability of such datasets can significantly impact the model's performance and generalization ability. Secondly, fine-tuning the pre-trained VGG16 model requires careful consideration of hyper parameters and regularization techniques to prevent over fitting and achieve optimal performance. Additionally, selecting the appropriate layers to freeze and layers to train during transfer learning is crucial to balance the model's ability to capture tumor-specific features while retaining the pre-trained knowledge. Lastly, interpreting the learned features and understanding the decision-making process of the model can be challenging, particularly when dealing with complex and high-dimensional data like medical images. Addressing these challenges requires a comprehensive understanding of both deep learning techniques and the domain of brain tumor detection.

This project includes the following applications:

* Early Detection: This brain tumor detection systems can help in the early detection of tumors, improving treatment outcomes and Early identification is vital for timely intervention and effective medical care.
* Automated Diagnosis: It can analyze brain MRI scans and accurately classify them as tumor-positive or tumor-negative. This automation speeds up the diagnostic process, reducing the burden on radiologists and improving efficiency.
* Treatment Planning: It can assist in treatment planning by providing insights into tumor characteristics, size, and location. This information helps doctors in designing personalized treatment strategies, including surgery, radiation therapy, or chemotherapy.
* Image Segmentation: VGG 16 algorithm can segment brain MRI images, precisely delineating tumor boundaries. Accurate segmentation aids in surgical planning, as it allows surgeons to identify the tumor's exact location and minimize damage to healthy brain tissue.
* Prognosis and Outcome Prediction: Models trained on large datasets can analyze various clinical and imaging features to predict patient prognosis and treatment outcomes. This information can help doctors in making informed decisions and counseling patients about their potential outcomes.

**CHAPER 8**

**CONCLUSION AND FUTURE WORK**

The "Brain tumor Detection" system offers an advanced solution for automatic brain tumor detection using deep learning techniques. By leveraging the VGG16 convolutional neural network and transfer learning, accurate classification of brain MRI images is achieved. It includes crucial components like data preprocessing, model training, and evaluation, ensuring a comprehensive pipeline for tumor detection. Popular libraries such as TensorFlow and Keras are utilized to enhance the efficiency and effectiveness of the implementation. This will contribute significantly to the field of medical image analysis and paves the way for future advancements in automated brain tumor detection systems.

The future enhancement for the project includes the following features:

* Augmented Data**:** Expand the dataset by applying transformations like rotation, scaling, and flipping to increase diversity and improve model performance.
* Multi-Class Classification: Extend the project to handle different tumor subtypes, enabling detailed classification and personalized treatment planning.
* Model Interpretability: Implement visualization techniques to understand the model's decision-making process and increase transparency.
* Web Interface: Develop a user-friendly web interface for uploading MRI images, visualizing results, and accessing additional tumor information.
* Performance Optimization: Explore model compression, quantization, and hardware acceleration techniques to optimize inference speed and resource utilization.
* Continuous Training: Incorporate mechanisms for periodically updating the model with new data to adapt and improve performance over time.

**APPENDICES**

**APPENDIX 1**

**SCREENSHOTS**

**Graphical user interface, text

Description automatically generated**

Fig A.1.1. Home Screen

**A screenshot of a computer

Description automatically generated with medium confidence**

Fig A.1.2. Selecting Image to Detect Tumor

Graphical user interface

Description automatically generated

Fig A.1.3. Identifies No Tumor in Brain

Text

Description automatically generated with medium confidence

Fig A.1.4. Identifies Tumor in Brain

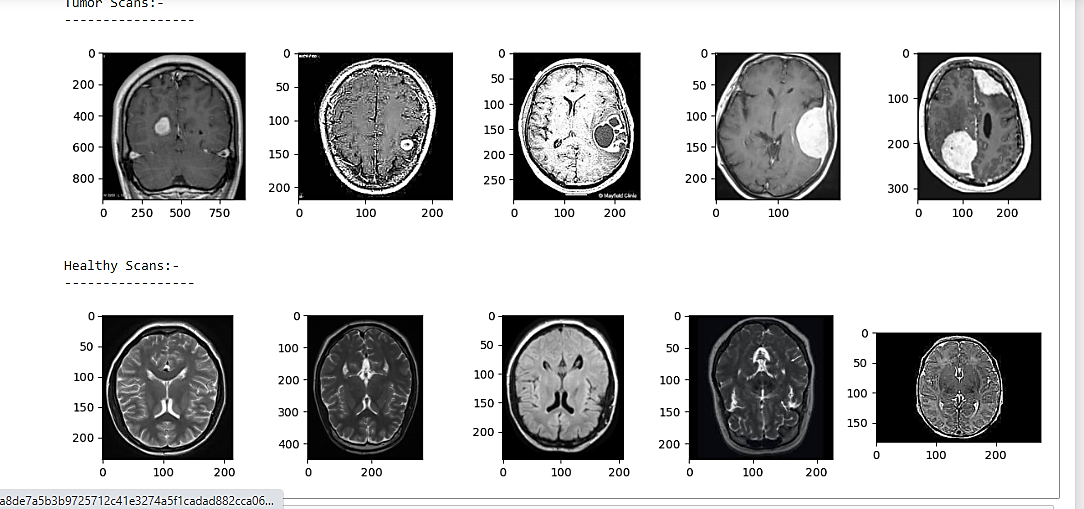


Fig A.1.5. Training Dataset MRI scan images

Chart, histogram

Description automatically generated

Fig A.1.6. Training Dataset image heights

Chart, histogram

Description automatically generated

Fig A.1.7. Training Dataset image widths

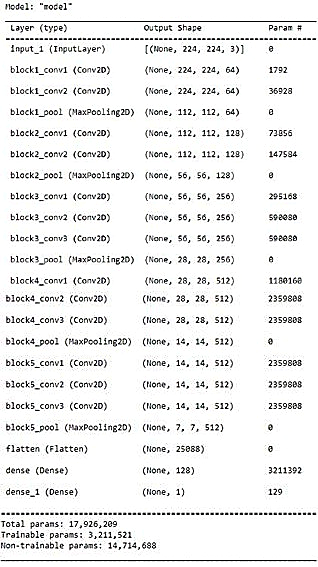
****

Fig A.1.8. Training VGG -16 Layers

Chart, histogram

Description automatically generated

Fig A.1.9. Epoch Vs Cross Entropy loss

Chart, line chart

Description automatically generated

Fig A.1.10. Epoch Vs Accuracy

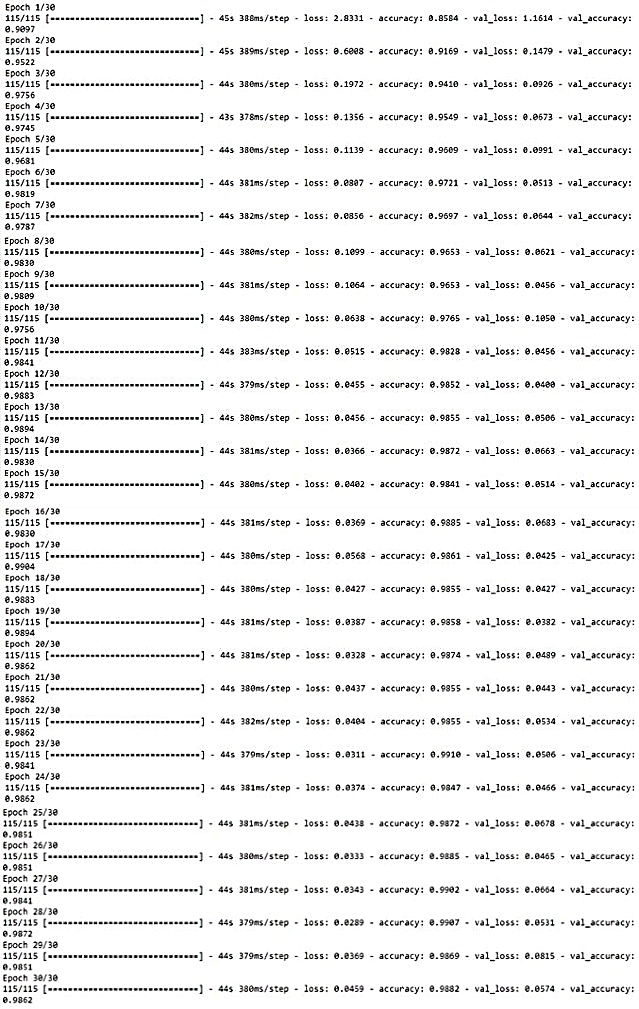
****

Fig A.1.7. Process of Training with Data

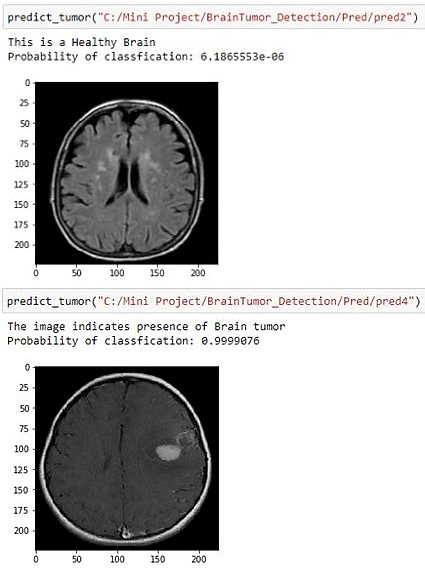
****

Fig A.1.12. Predicting Tumor in MRI with accuracy

**APPENDIX 2**

**SOURCE CODE**

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>Detecting Tumor</title>

<link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css" integrity="sha384-ggOyR0iXCbMQv3Xipma34MD+dH/1fQ784/j6cY/iJTQUOhcWr7x9JvoRxT2MZw1T" crossorigin="anonymous">

<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>

</head>

<body style="color: white;background-color: black">

<h1 class="text-center">Brain Tumor Detection</h1>

<div style="margin-top: 10%">

<form class="text-center" action="/predict" method="post" enctype="multipart/form-data">

<ul style="list-style-type: none;">

<li style="margin-left: 5%;"><input id="chooseimg" class="text-center" type="file" name="imagefile" class="form-control"></li>

<li><input id="detecttum" type="submit" class="btn mt-3" value="Detect Tumor" style="border: 1px solid white;color: black;background-color: yellow"></li>

</ul>

</form>

</div>

<br>

{% if prediction %}

<p class="text-center" id="predtext">{{prediction}}</p>

{% endif %}

<br>

{% if imageloc %}

<center style="margin-left: 3%;"><img id="tumorimg" src="static/{{imageloc}}" width="220" height="230" style="border: 5px solid white"></center>

{% endif %}

</body>

<script>

$("#chooseimg").on('click',function(){

$("#predtext").hide();

$("#tumorimg").hide();

});

$("#detecttum").on('click',function(){

$( window ).load(function() {

$("#predtext").show();

$("#tumorimg").show();

});

});

</script>

</html>

**App.py**

import numpy as np

from tensorflow.keras.preprocessing.image import load\_img,img\_to\_array

from tensorflow.keras.models import load\_model

from tensorflow.keras.applications.vgg16 import preprocess\_input

from flask import Flask,redirect, url\_for, request, render\_template

import flask

app = Flask(\_\_name\_\_)

best = load\_model("save\_model4.h5")

#Adding image pre-processing function

def predict\_tumor(img\_path):

print("entered predict tumor")

# load the image

img = load\_img(img\_path, target\_size=(224, 224)) #(224,224,3)

# convert to array

img = img\_to\_array(img) #(224,224,3)

# add batch size as a dimension

img = np.expand\_dims(img, axis=0) #(1,224,224,3)

if best.predict(img)[0][0]>0.45: # just a threshold

return "This MRI scan indicates presence of Brain tumor"

else:

return "This is a Healthy Brain"

@app.route('/', methods=['GET'])

def welcome():

# Main page

return render\_template('index.html')

@app.route('/predict', methods=['POST'])

def predict():

if request.method == "POST":

imagefile=request.files['imagefile']

if imagefile:

image\_path = "./static/" + imagefile.filename

imagefile.save(image\_path)

return render\_template('index.html',prediction=predict\_tumor(image\_path),imageloc=imagefile.filename)

return render\_template('index.html',prediction=predict\_tumor(image\_path),imageloc = None)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(port=8080)

**Brain\_tumor.ipynb**

**#**Importing Libraries

import warnings

warnings.filterwarnings("ignore")

import matplotlib.pyplot as plt

from matplotlib.image import imread

import os

import numpy as np

import pandas as pd

import random

import shutil

import tensorflow as tf

from tensorflow.keras.preprocessing import image

from tensorflow.keras.preprocessing.image import ImageDataGenerator,load\_img,img\_to\_array

from tensorflow.keras.models import Sequential,Model,load\_model

from tensorflow.keras.layers import Conv2D,MaxPooling2D,Flatten,Dense,Dropout

import math

import seaborn as sns

from tensorflow.keras.callbacks import Callback,LearningRateScheduler

from tensorflow.keras.applications.vgg16 import VGG16,preprocess\_input

from prettytable import PrettyTable

import cv2

from tqdm import tqdm

# **#**Some Initial investigation of the Images

tumor\_folder = r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor\Brain Tumor Dataset\tumor"

healthy\_folder = r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor\Brain Tumor Dataset\healthy"

tumor\_listdir = os.listdir(tumor\_folder)

healthy\_listdir = os.listdir(healthy\_folder)

print(f"No. of tumor images={len(tumor\_listdir)}")

print(f"No. of healthy images={len(healthy\_listdir)}")

tumor\_img\_extensions = list(set([img.split(".")[1] for img in tumor\_listdir]))

healthy\_img\_extensions = list(set([img.split(".")[1] for img in healthy\_listdir]))

diff\_extensions = list(set(tumor\_img\_extensions+healthy\_img\_extensions))

print(f"Different extensions of images are: {diff\_extensions}")

print()

print("-------------------------------------------------------------------------")

print()

# let's look at some tumorous and healthy MRI photos

def plot\_photos(folder):

images = os.listdir(folder)

plt.subplots(figsize=(15, 5))

# plot any 5 images

for i in range(1,6):

# define subplot

plt.subplot(2,5,i)

choice = random.randint(0,len(images)-1)

chosenImg = images[choice]

# define file path

filename = folder + "/" + chosenImg

# load image pixels

image = imread(filename)

# plot raw pixel data

plt.imshow(image)

# show the figure

plt.show()

print("Tumor Scans:-")

print("-----------------")

plot\_photos(tumor\_folder)

print()

print("Healthy Scans:-")

print("-----------------")

plot\_photos(healthy\_folder)

# renaming properly since this is an entirely separate dataset

os.rename(r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\no",r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\healthy")

os.rename(r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\yes",r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\tumor")

test\_tumor\_listdir = os.listdir(r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\tumor")

test\_healthy\_listdir = os.listdir(r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset\healthy")

print(f"No. of tumor images in test dataset={len(test\_tumor\_listdir)}")

print(f"No. of healthy images in test dataset={len(test\_healthy\_listdir)}")

# **#**Creating Standard directories

# moving all images (of both- tumor and healthy) together into a single temporary folder to further perform train-test split

# move tumor images first

shutil.copytree(tumor\_folder, 'temp', shutil.move)

# move healthy images now

for file in healthy\_listdir:

src = healthy\_folder+"/"+file

shutil.copy(src,"temp")

# sanity check

print("Total No. of images in this temp folder=",len(os.listdir("temp")))

#creating directories

parent = "tumor\_vs\_healthy/"

child\_dirs = ["train/","validation/"]

for i in child\_dirs:

#create subdirectories for the 2 labels

label\_sub\_dirs = ["tumor/","healthy"]

for j in label\_sub\_dirs:

newdir = parent + i + j

os.makedirs(newdir,exist\_ok=True)

# seed random number generator

random.seed(1) # to get possible similar set of random numbers each time

# defining 80-20 ratio of train-validation split

val\_ratio=0.20

# copy the images from temp folder into respective subdirectories for train and validation

src\_directory = "temp"

for file in os.listdir(src\_directory):

s = src\_directory + "/" + file

dst\_dir = "train/"

if random.random() <val\_ratio:

dst\_dir = "validation/"

if file.startswith('y'):

d = parent + dst\_dir + "tumor/" + file

shutil.copyfile(s,d)

eliffile.startswith('no') or file.startswith('No'):

d = parent + dst\_dir + "healthy/" + file

shutil.copyfile(s,d)

#print the no of images stored in folders

print("Sanity for no. of images in each folder:-")

print("train")

print("healthy-->",len(os.listdir("tumor\_vs\_healthy/train/healthy")))

print("tumor-->",len(os.listdir("tumor\_vs\_healthy/train/tumor")))

print("validation")

print("healthy-->",len(os.listdir("tumor\_vs\_healthy/validation/healthy")))

print("tumor-->",len(os.listdir("tumor\_vs\_healthy/validation/tumor")))

# **#**Basic EDA

# a simple EDA to get to know the height & width which have maximum no. of images

# using temp folder since all images are present here

parent = "C:/Mini Project/BrainTumor\_Detection/temp/"

all\_heights = []

all\_widths = []

for i in os.listdir(parent):

filename = parent + i

image = imread(filename)

all\_heights.append(image.shape[0])

all\_widths.append(image.shape[1])

#pdf of all image heights

sns.displot(data=all\_heights,kind="kde",height=4,aspect=3)

plt.title("pdf of all image heights")

plt.xlabel("Height")

#pdf of all image widths

sns.displot(data=all\_widths,kind="kde",height=4,aspect=3)

plt.title("pdf of all image widths")

plt.xlabel("Width")

# **#**Preprocessing

BATCH\_SIZE=32

TARGET\_SIZE=(224,224) # since VGG16 expects this size of images

# scaling the pixel values to the range 0-1 and augmenting:-

# (note: we'll augment the train data only. Not the validation/test data)

# preprocessing\_function will stay common in all 3 sets

# creating the train data generator

datagen = ImageDataGenerator(width\_shift\_range=0.1,height\_shift\_range=0.1,

zoom\_range=0.2,shear\_range=0.2,horizontal\_flip=True,

preprocessing\_function=preprocess\_input)

# preparing train iterator

path1 = r"C:\Mini Project\BrainTumor\_Detection\tumor\_vs\_healthy\train"

train\_it = datagen.flow\_from\_directory(path1,class\_mode="binary",batch\_size=BATCH\_SIZE,target\_size=TARGET\_SIZE)

# creating the data generator for validation & test sets

datagen = ImageDataGenerator(preprocessing\_function=preprocess\_input)

# preparing validation & test iterators

path2 = r"C:\Mini Project\BrainTumor\_Detection\tumor\_vs\_healthy\validation"

validation\_it = datagen.flow\_from\_directory(path2,class\_mode="binary",batch\_size=BATCH\_SIZE,target\_size=TARGET\_SIZE)

path3 = r"C:\Mini Project\BrainTumor\_Detection\Brain Tumor Dataset"

test\_it = datagen.flow\_from\_directory(path3,class\_mode="binary",batch\_size=BATCH\_SIZE,target\_size=TARGET\_SIZE)

print("Our data pre-processing is now done!")

print()

print("Checking the Binary Labels:-")

print(train\_it.class\_indices,validation\_it.class\_indices,test\_it.class\_indices)

# just a sanity for deciding No. of batches or steps\_per\_epoch:-

print("No. of batches or steps\_per\_epoch for train set should be-->","2403/32=",2403/32,

"i.e.",math.ceil(3659/32), ",which is simply", "len(train\_it3)=",len(train\_it3))

print("No. of batches or steps\_per\_epoch for validation set should be-->","597/32=",597/32,

"i.e.",math.ceil(941/32), ",which is simply", "len(validation\_it3)=",len(validation\_it3))

# **#**Defining Model Architectures & Evaluation

### Re-usable functions &Keras Callbacks

# function for plotting diagnostic curves --> a re-usable function

def summarize\_diagnostics(history):

plt.subplot(1,2,1)

plt.title("Epoch vs Binary Cross Entropy Loss")

plt.plot(history.history['loss'],color="blue",label="train")

plt.plot(history.history['val\_loss'],color="orange",label="test")

plt.xlabel("Epoch")

plt.legend()

plt.subplot(1,2,2)

plt.title("Epoch vs Accuracy")

plt.plot(history.history['accuracy'],color="blue",label="train")

plt.plot(history.history['val\_accuracy'],color="orange",label="test")

plt.xlabel("Epoch")

plt.legend()

plt.show()

#To schedule the Learning Rate

def lf\_schedule(epoch,lr):

if epoch%10 ==0:

lr=lr - 0.05\*lr #-->i.e. 0.95\*lr (5% decay)

print(f'New learning rate for epoch={epoch} is {lr}')

return lr

else:

return lr

callback\_lrs = LearningRateScheduler( schedule=lf\_schedule,verbose=1 )

# **#**Transfer Learning for Feature Extraction via VGG16 + Data Augmentation

# tf.keras.backend.clear\_session()

# defining 1st model architecture

# loading the base-model

model=VGG16(weights='imagenet',include\_top=False,input\_shape=(224,224,3))

# mark the loaded layers as not-trainable

for layer in model4.layers:

layer.trainable=False

# adding new classifier layers

flat = Flatten()(model.layers[-1].output)

classifier = Dense(units=128,activation='relu',kernel\_initializer="he\_uniform")(flat)

output = Dense(units=1,activation="sigmoid")(classifier)

# define the model

model = Model(inputs=model4.inputs,outputs=output)

# compile the model

model.compile(optimizer="adam",loss="binary\_crossentropy",metrics=["accuracy"])

model.summary()

#process of training data

history = model.fit(train\_it3,steps\_per\_epoch=len(train\_it3),

validation\_data=validation\_it3,validation\_steps=len(validation\_it3),

epochs=30)

# plot accuracy and loss

print("Scanning the diagnostics of the model performance:-")

plt.subplots(figsize=(20, 5))

summarize\_diagnostics(history)

#---------

loss,acc = model4.evaluate(test\_it3,steps=len(test\_it3))

print("Evaluation Accuracy on test data-set=",np.round(acc\*100,3),"%")

**#**Sample Predictions

def predict\_tumor(img\_path):

# load the image

img = load\_img(img\_path, target\_size=(224, 224)) #(224,224,3)

plt.imshow(img)

# convert to array

img = img\_to\_array(img) #(224,224,3)

# add batch size as a dimension

img = np.expand\_dims(img, axis=0) #(1,224,224,3)

best = load\_model("/content/drive/MyDrive/Data\_Science\_Portfolio/BrainTumor\_Detection/save\_model4.h5")

if best.predict(img)[0][0]>0.4:

print("The image indicates presence of Brain tumor")

else:

print("This is a Healthy Brain")

print("Probability of classfication:",best.predict(img)[0][0])

#-----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Pred/pred1")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Pred/pred2")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Pred/pred3")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Pred/pred4")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Brain Tumor Dataset/healthy/no 97.jpg")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Brain Tumor Dataset/tumor/Y187.jpg")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Brain Tumor Dataset/tumor/Y154.jpg")

#----

predict\_tumor("C:/Mini Project/BrainTumor\_Detection/Brain Tumor Dataset/healthy/no 37.jpg")

**REFERENCES**

[1] Anitha. V.;Murugavalli.S. Brain tumor classification using two-tier classifier with adaptive segmentation technique.IET Comput. Vis.2016,10,9–11

[2] A Customized VGG19 Network with Concatenation of Deep and Handcrafted Features for Brain Tumor Detection by Venkatesan Rajinikanth, Alex Noel Joseph Raj, Krishnan Palani Thanaraj and Ganesh R. Naik

[3] Beers. A.; Chang. K.; Brown.J.; Sartor.E.; Mammen.C.P.; Gerstner,E.; Rosen. B.; Kalpathy-Cramer. J. Sequential 3D U-nets for biologically informed brain tumor segmentation. arXiv 2017, 1709, 02967.

[4] Brain Tumor Analysis Using Deep Learning and VGG-16 Ensembling Learning Approaches by Ayesha Younis, Li Qiang, Charles Okanda Nyatega , Mohammed Jajere Adamu, Halima Bello Kawuwa

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

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