ABSTRACT

The detection, segmentation, and extraction from Magnetic Resonance Imaging (MRI) images of contaminated tumor areas are significant concerns however, a repetitive and extensive task executed by radiologists or clinical experts relies on their expertise. Image processing concepts can imagine the various anatomical structure of the human organ. Detection of human brain abnormal structures by basic imaging techniques is challenging. In this paper, a Fully Automatic Heterogeneous Segmentation using Support Vector Machine (FAHS-SVM) has been proposed for brain tumor segmentation based on deep learning techniques. The present work proposes the separation of the whole cerebral venous system into MRI imaging with the addition of a new, fully automatic algorithm based on structural, morphological, and relaxometry details. The segmenting function is distinguished by a high level of uniformity between anatomy and the neighboring brain tissue. ELM is a type of learning algorithm consisting of one or more layers of hidden nodes. Such networks are used in various areas, including regression and classification. In brain MRI images, the probabilistic neural network classification system has been utilized for training and checking the accuracy of tumor detection in images. The numerical results show almost 98.51% accuracy in detecting abnormal and normal tissue from brain Magnetic Resonance images that demonstrate the efficiency of the system suggested.

1. INTRODUCTION

Gliomas are the most prevalent and lethal brain tumors, classified into Low-Grade (LGG) and High-Grade Gliomas (HGG), with HGG being more aggressive and typically leading to a survival period of only 14 months post-diagnosis, even with treatments like surgery, chemotherapy, and radiotherapy. MRI plays a crucial role in glioma assessment due to its ability to capture detailed and complementary sequences. Accurate segmentation of gliomas and their sub-regions is essential for diagnosis, treatment planning, and follow-up, but manual segmentation is time-consuming and prone to inter- and intra-rater variability. Automatic segmentation is challenging due to tumor variability in shape, size, and location, as well as MRI intensity inconsistencies and tumor-induced deformation of normal brain tissue. Traditional approaches involve probabilistic models, Markov Random Fields (MRF), and probabilistic atlases, while machine learning methods such as Support Vector Machines (SVM) and Random Forests (RF) have proven effective by incorporating features like texture, symmetry, and intensity. Deep learning methods, particularly Convolutional Neural Networks (CNNs), have shown superior performance by automatically learning hierarchical features from raw data, overcoming the need for handcrafted features. Techniques like 3D filters, multi-pathway architectures, cascaded networks, and subject-specific training further enhance segmentation accuracy. Inspired by VGG-style CNNs, recent work proposes using small 3×3 kernels to build deeper networks with fewer parameters and greater non-linearity, reducing overfitting while maintaining large receptive fields. Pre-processing with intensity normalization (e.g., Nyúl's method) addresses variability from multi-site MRI scanners, while data augmentation techniques improve model robustness. These advanced CNN-based methods significantly improve glioma segmentation, aiding clinical decisions and improving patient outcomes.

1.1 MOTIVATION

Gliomas are aggressive brain tumors with high mortality, requiring precise segmentation for effective treatment. Manual segmentation is time-consuming and inconsistent, leading to the need for automated methods. Traditional machine learning methods like Random Forests have shown success, but recent advancements in deep learning—especially Convolutional Neural Networks (CNNs)—have significantly improved segmentation accuracy by learning complex patterns from raw MRI data. The use of small convolutional kernels, data augmentation, and intensity normalization techniques enhances model performance and robustness, making CNN-based models promising tools for clinical brain tumor analysis.

- **Develop an Accurate Glioma Segmentation Model:** Create a deep learning-based framework capable of accurately identifying and segmenting glioma regions and sub-regions in brain MRI scans.
- **Improve Clinical Workflow:** Reduce the reliance on manual segmentation by radiologists, which is time-consuming and error-prone, thereby enhancing efficiency and consistency in diagnosis and treatment planning.
- **Handle Data Variability:** Address challenges related to intensity inhomogeneity and scanner differences using pre-processing techniques like intensity normalization.
- **Utilize Deep Learning for Feature Extraction:** Employ Convolutional Neural Networks (CNNs) to automatically learn hierarchical and complex features from raw MRI data without the need for handcrafted feature engineering.
- **Enhance Model Robustness and Generalization:** Implement data augmentation strategies to deal with limited datasets and improve the model's ability to generalize across diverse cases and tumor presentations.
- **Incorporate Efficient Network Design:** Use small 3×3 convolutional kernels to design deeper yet more efficient CNN architectures that reduce overfitting and increase segmentation accuracy.

- **Compare with Traditional Techniques:** Evaluate the performance of the deep learning model against traditional machine learning methods (e.g., Random Forests) to highlight improvements in segmentation precision.
- **Facilitate Real-world Application:** Ensure the proposed system can be adapted for clinical use, supporting radiologists with reliable and reproducible results in tumor analysis and monitoring.

1.2 PROBLEM STATEMENT

Brain tumors, particularly gliomas, are among the most life-threatening neurological disorders, with high mortality rates and complex treatment requirements. Accurate and timely identification of these tumors is critical for effective treatment planning. However, manual interpretation of MRI scans is time-consuming, prone to inter-observer variability, and often inconsistent due to the tumors irregular shapes, locations, and appearances across patients. Additionally, variability in MRI image quality caused by different scanners and acquisition protocols makes diagnosis even more challenging. Traditional segmentation and classification methods lack the adaptability and precision needed for reliable results. Therefore, there is a pressing need for an intelligent, automated system that can accurately detect and classify brain tumors in MRI images using advanced deep learning techniques—enhancing diagnostic accuracy, reducing workload for medical professionals, and supporting timely clinical decisions.

1.3 SCOPE AND OBJECTIVE

The aim of tumor detection is to specify the tumor location, namely active tissue or necrotic tumor tissue. This is carried out by detecting the abnormal region comparing with the normal healthy tissues.dtheuseofsmall3 3kernelstoobtain deeper CNNs. With smaller kernels we can stack more convolutional layers, while having the same receptive field of bigger kernels. For instance, two 3*3

cascaded convolutional layers have the same effective receptive field of one 5*5 layer, but fewer weights. At the same time, it has the advantages of applying more non-linearity and being less prone to overfitting because small kernels have fewer weights than bigger kernels as a pre-processing step that aims to address data heterogeneity caused by multi-site multi-scanner acquisitions of MRI images. The large spatial and structural variability in brain tumors are also an important concern that we study using two kinds of data augmentation.

Goals:

- **Accurate Tumors Localization**: Detect and delineate tumor regions such as active and necrotic tissue by comparing MRI scans with normal brain tissue structures.
- **Design Efficient CNN Architectures**: Use small 3×3 convolutional kernels to construct deep CNNs, enabling increased non-linearity, reduced overfitting, and efficient feature extraction with fewer parameters.
- **Normalize Multi-Site MRI Data**: Apply intensity normalization techniques to handle variability in MRI images caused by different scanners and acquisition settings.
- **Enhance Model Robustness**: Employ diverse data augmentation methods to manage the high spatial and structural variability of brain tumors and improve model generalization.
- **Automate Tumors Classification Pipeline**: Develop a scalable deep learning framework that automates tumor detection and classification to support clinicians in diagnosis and treatment planning.

The goal of this project is to reduce manual tumor segmentation by using Convolutional Neural Networks (CNNs), a type of artificial neural network designed for image analysis. CNNs help automate the detection and classification of brain tumors, addressing challenges like variability in tumor shape, size, and location. To improve accuracy, the system also uses neighborhood information from voxels and probabilistic models like Markov Random Fields (MRF) for smoother and more reliable segmentation.

2. LITERATURE SURVEY

S. Bauer – “A Survey of MRI-Based Medical Image Analysis for Brain Tumor Studies”. This study provides an overview of how MRI images are used for brain tumor analysis. It focuses on the segmentation process—separating the tumor and its parts from the surrounding brain tissue—and discusses the challenges caused by the tumor changing the shape and structure of brain anatomy. The paper compares different methods and highlights those that work well with standard MRI scans used in hospitals.

E. G. Van Meir – “Exciting New Advances in Neuro-Oncology: The Avenue to a Cure for Malignant Glioma”. This paper discusses the latest progress in treating malignant gliomas, which are the deadliest form of brain tumors. It talks about the use of a chemotherapy drug called temozolomide and improvements in surgery using special imaging techniques like fluorescence-guided surgery. The author is hopeful that in the near future, new combinations of drugs will better target these tumors, reducing their deadly effects and giving hope to patients and doctors.

M. Prastawa – “A Brain Tumor Segmentation Framework Based on Outlier Detection”. This research introduces a method that not only segments the tumor but also detects the surrounding edema (swelling), which is essential for treatment planning. Unlike many other techniques, this method does not rely solely on the contrast enhancement seen in T1-weighted MRI scans, making it more flexible and reliable.

C. H. Lee – “Segmenting Brain Tumors Using Pseudo-Conditional Random Fields”. This work proposes a segmentation technique using pseudo-conditional random fields (PCRFs). These are faster and more efficient than traditional methods while maintaining high accuracy. The method considers not just the pixel itself but also the information from nearby pixels, which helps make better predictions about tumor regions.

R. Meier – “Patient-Specific Semi-Supervised Learning for Postoperative Brain Tumor Segmentation”. This paper presents a semi-supervised learning method that uses both pre-surgery and post-surgery images of the same patient to improve the accuracy of tumor segmentation. This personalized approach makes the segmentation more reliable, especially when dealing with changes in the brain after surgery.

D. Zikic – “Segmentation of Brain Tumor Tissues with Convolutional Neural Networks”. The authors explore using Convolutional Neural Networks (CNNs) to segment brain tumors directly from MRI images. They feed the network small image patches with multi-channel intensity data. Only basic pre-processing is used to adjust for different MRI scanners, showing that CNNs can effectively handle real-world data variability and perform accurate tumor segmentation.

2.1 EXISTING SYSTEM

There are many image processing method, for example, histogram equalization, picture segmentation, image enhancement, morphological operation, feature choice and obtaining the features, and order.

A wide range of image processing techniques was used in segmenting brain tumor tissues. Some researchers used basic approaches for segmentation such as thresholding segmentation techniques. Another basic approach is edge detection which was used to detect the change of luminance intensity around the tumor to segment the tumor region. Region growing algorithm was also used for detecting and segmenting brain tumor in MRI images.

DISADVANTAGES

1. **Increased Complexity:** Using various filters, Fourier transforms, and discrete transforms increases the computational complexity of the processing system.
2. **High Equipment Cost:** The specialized equipment required for advanced image processing and diagnosis is expensive.

3. **Need for Skilled Personnel:** The presence of trained medical personnel is necessary to interpret the results correctly and make accurate decisions.
4. **Limited Equipment Availability:** Diagnosis and processing can be performed only with particular, often highly specialized, equipment.
5. **Time-Consuming Process:** Some segmentation methods (like region growing or manual thresholding) can be slow and require manual intervention.
6. **Sensitivity to Image Quality:** Techniques like edge detection and thresholding are sensitive to noise and variations in MRI image quality, potentially leading to inaccurate segmentation.
7. **Low Generalization:** Basic methods like thresholding or edge detection might not perform well across different datasets or varying tumor shapes, sizes, and intensities.

2.2 PROPOSED SYSTEM

Fully Automatic Heterogeneous Segmentation using Support vector machine (FAHS-SVM) for brain tumor detection and segmentation. Figure 2 describes the proposed FAHS-SVM method architecture. Magnetic Resonance Imaging is a diagnostic tool for human anatomy study and testing. Increased tumor vascularity leads to preferential uptake of the contrasting agent and can be utilized better to view the tumors of the normal tissue around them. When the contrast injections are performed repeatedly, the dynamic nature of contrast uptake can be tested, which can increase the distinction between malignant or benign diseases.

Implementation:

- To propose Fully Automated Heterogeneous Segmentation using Support Vector Machine (FAHSSVM) for brain tumor detection and segmentation.
- To design an Extreme learning machine algorithm for the classification and feature extraction of MRI images.
- The experimental results show high accuracy in detecting brain tumors with the help of datasets.

2.3 FUNCTIONAL REQUIREMENTS

1) **Image Uploading:** The system must allow users (doctors, lab staff) to upload MRI images in accepted formats (like JPEG, PNG, or DICOM).
2) **Preprocessing:** The system should automatically preprocess uploaded MRI images (e.g., resize, normalize, denoise) to make them ready for analysis.
3) **Tumor Detection:** The system must detect the presence of a tumor in the MRI image by analyzing abnormalities compared to healthy brain tissues.
4) **Tumor Localization:** The system should identify and highlight the exact location of the tumor (active tissue or necrotic tissue) within the MRI image.
5) **Tumor Classification:** After detection, the system must classify the tumor into categories such as Low-Grade Glioma (LGG) or High-Grade Glioma (HGG).
6) **Result Display:** The system should clearly display the diagnosis result to the user with probabilities or confidence scores.
7) **Report Generation:** The system must generate a simple diagnostic report containing tumor detection results, classification details, and tumor location visualization.
8) **Data Storage:** All uploaded images, processed results, and reports should be saved securely in the database for future reference.
9) **User Interface:** The system must provide a clean, simple, and user-friendly interface to upload images, view results, and download reports.
10) **Security and Privacy:** The system must ensure that patient data is encrypted and access-controlled, maintaining data privacy according to medical standards.
11) **Performance:** The system should provide quick processing and response times, aiming for minimal delay between upload and result.
12) **Model Updating:** The system must allow for future updates to the deep learning model to improve accuracy with new data.

2.4 NON-FUNCTIONAL REQUIREMENTS

Describe user-visible aspects of the system that are not directly related with the functional behavior of the system. Non-Functional requirements include quantitative constraints, such as response time (i.e. how fast the system reacts to user commands.) or accuracy (.e. how precise are the systems numerical answers.).

- Portability
- Reliability
- Usability
- Time Constraints
- Error messages
- Actions which cannot be undone should ask for confirmation
- Responsive design should be implemented
- Space Constraints
- Performance
- Standards
- Ethics
- Interoperability
- Security
- Privacy
- Scalability

2.5 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

- ECONOMICAL FEASIBILITY
- TECHNICAL FEASIBILITY
- SOCIAL FEASIBILITY

2.5.1 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

2.5.2 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

2.5.3 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.6 SOFTWARE REQUIREMENTS

- Programming Language / Platform : Python
- IDE : PyCharm
- Web Framework : Flask
- UI Technologies : HTML, CSS, JS
- Database : SQLite

2.7 HARDWARE REQUIREMENTS

- Processor : Intel i3 and above
- RAM : 4GB and Higher
- Hard Disk : 500GB: Minimum

3. IMPLEMENTATION

3.1 IMPLEMENTATION TECHNIQUES

❖ **Introduction of Python**

Python is an interpreted, object-oriented, high-level programming language known for its dynamic semantics. It features built-in data structures, dynamic typing, and binding, making it ideal for Rapid Application Development and as a scripting language to connect components. Its simple and readable syntax reduces maintenance costs. Python supports modules and packages, promoting code reuse and modular programming. The interpreter and standard libraries are freely available across all major platforms. Python's lack of a compilation step makes the edit-test-debug cycle very fast. Debugging is easy because errors raise exceptions instead of causing crashes, and Python offers a built-in debugger for inspecting variables and stepping through code. The debugger itself is written in Python, showing the language's introspective capabilities. Additionally, adding simple print statements is often enough for quick and effective debugging.

❖ **Introduction to PyCharm**

Many programmers nowadays opt for Python to build software applications with the concise, clean, and readable code base. They can even accelerate custom software application development by taking advantage of a number of integrated development environments (IDEs) for Python.

JetBrains has developed PyCharm as a cross-platform IDE for Python. In addition to supporting versions 2.x and 3.x of Python, PyCharm is also compatible with Windows, Linux, and macOS. At the same time, the tools and features provided by PyCharm help programmers to write a variety of software applications in Python quickly and efficiently. The developers can even customize the PyCharm UI according to their specific needs and preferences. Also, they can extend the IDE by choosing from over 50 plug-ins to meet complex project requirements. PyCharm is one of the most widely used IDEs

for Python programming language. At present, the Python IDE is being used by large enterprises like Twitter, Pinterest, HP, Symantec, and Groupon.

❖ Introduction of Flask

Flask is a lightweight, easy-to-use Python web framework that helps you build web applications quickly. In a deep learning project, Flask acts like a bridge between your trained model and users**.** It allows you to create a web interface where users can upload images, texts, or any input, and get the predictions from your deep learning model.

Why Flask in Deep Learning Projects?

- **Model Deployment**: Flask helps you deploy your trained deep learning model easily so that it can be accessed online.
- **User Interaction**: Users can interact with your model through a simple web page (upload image, click predict).
- **API Creation**: Flask can be used to create an API (Application Programming Interface) so that other applications (like mobile apps) can access your model.
- **Lightweight**: It doesn't add unnecessary complexity good for small to medium-sized ML/DL projects.
- **Flexible**: You can control everything: page design (HTML/CSS), how the model loads, and how predictions are returned.

Basic Workflow of Flask in Deep Learning

1. **Train Your Model**: First, train your deep learning model (for example, brain tumor detection using CNN).
2. **Save the Model**: Save the trained model using libraries like pickle, joblib, or .h5 format (Keras model).
3. **Create a Flask App**:

   - Load the trained model.
   - Set up routes (e.g., homepage, upload page).
   - Take user input (like an MRI image).

- Process the input and send it to the model.
- Return and display the prediction result.

4. **Run the Server**: Flask will create a local server where you can test your app on your browser (localhost).
5. **Deploy Online** (optional): You can deploy the Flask app to cloud platforms like Heroku**,** AWS**,** Azure, etc., to make it available to anyone.

SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

SOFTWARE DESIGN REQUIREMENTS:

In designing the software following principles are followed:

1. **Modularity and partitioning**: software is designed such that, each system should consists of hierarchy of modules and serve to partition into separate function.

2. **Coupling:** modules should have little dependence on other modules of a system.

3. **Cohesion:** modules should carry out in a single processing function.

4. **Shared use:** avoid duplication by allowing a single module be called by other that need the function it provide.

DATA FLOW DIAGRAM

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.
3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional details

Data flow diagrams are used to graphically represent the flow of data in a business information system. DFD describes the processes that are involved in a system to transfer data from the input to the file storage and reports generation.

Data flow diagrams can be divided into logical and physical. The logical data flow diagram describes flow of data through a system to perform certain functionality of a business. The physical data flow diagram describes the implementation of the logical data flow.

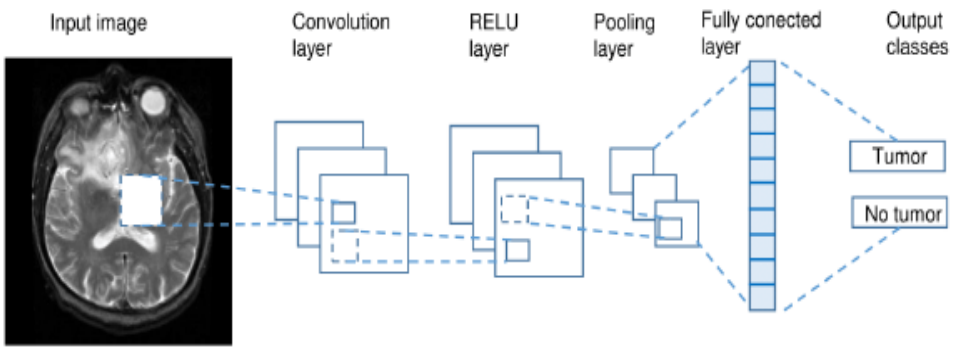

FIG: 3.1.1 Data flow diagram

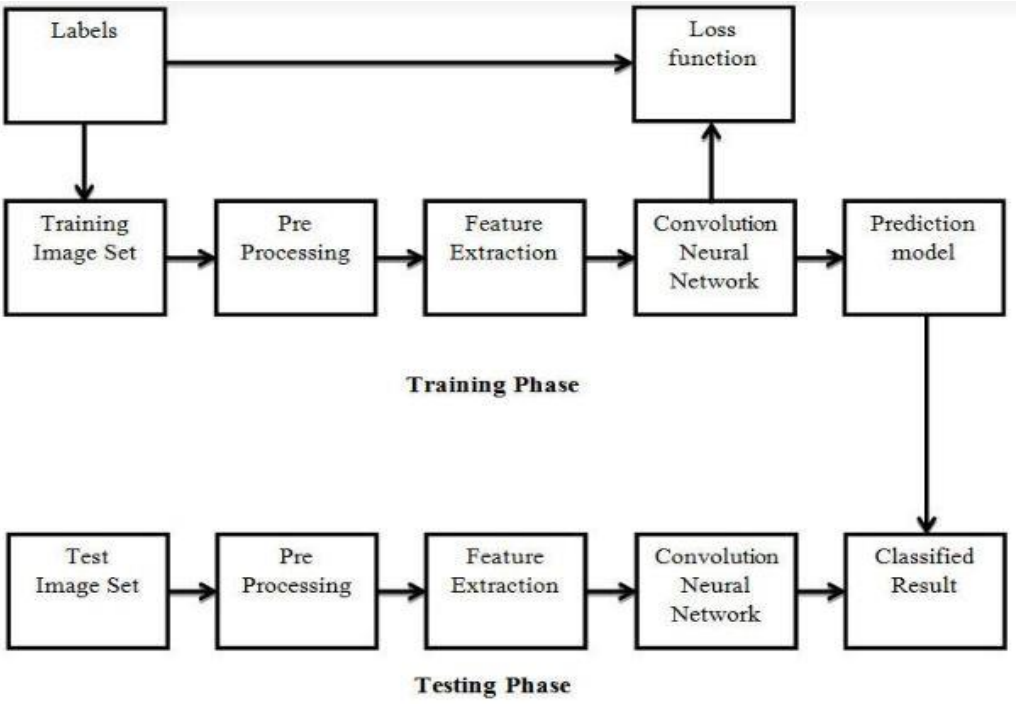

FIG: 3.1.2 Flow Chart

Software Development Life Cycle:

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.

Software Development Life Cycle (SDLC) is the process of building or updating software in a structured way. It includes these main steps:

- **Requirement Analysis and Design**: First, we gather and understand what the system should do. Then, we plan how it will work — what programs are needed, how they connect, what the user interface will look like, and what data is required. A good design is important to avoid problems later.
- **Implementation**: In this step, the planned design is turned into real code. Programming languages like Python 3.6 and tools like Anaconda Cloud are used. Developers write and test small parts of the program using compilers, interpreters, and debuggers.

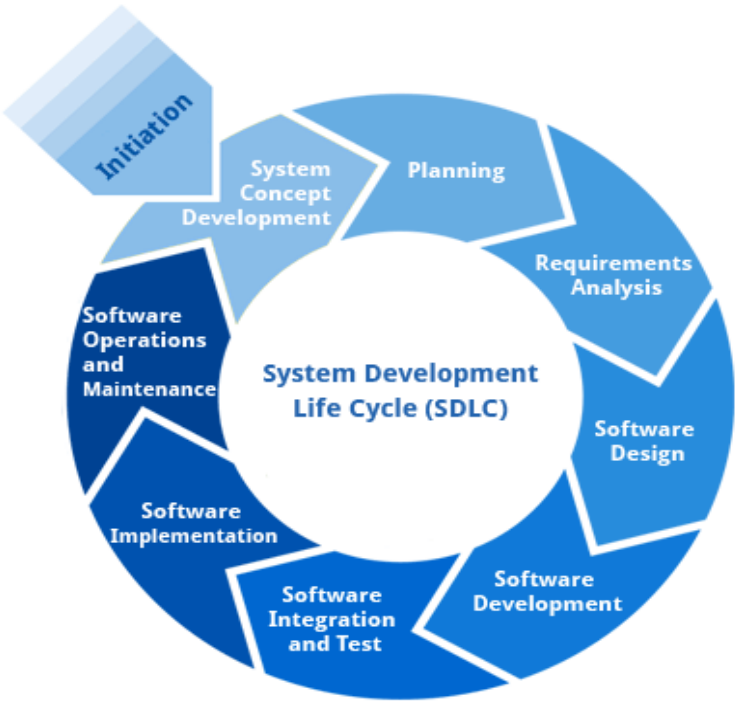

FIG 3.1.3 Software development life cycle

- **Testing**: After coding, the system is tested carefully. First, each part is tested alone, then all parts are tested together. It checks if the system works correctly, handles large data, and meets user needs.
- **Maintenance**: Once the software is used by customers, it may need updates or fixes. Maintenance ensures the system stays updated and works smoothly even after changes.

SDLC METHDOLOGIES

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process. SPIRAL MODEL was defined by Barry Boehm in his 1988 article, "A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models. As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project. The following diagram shows how a spiral model acts like:

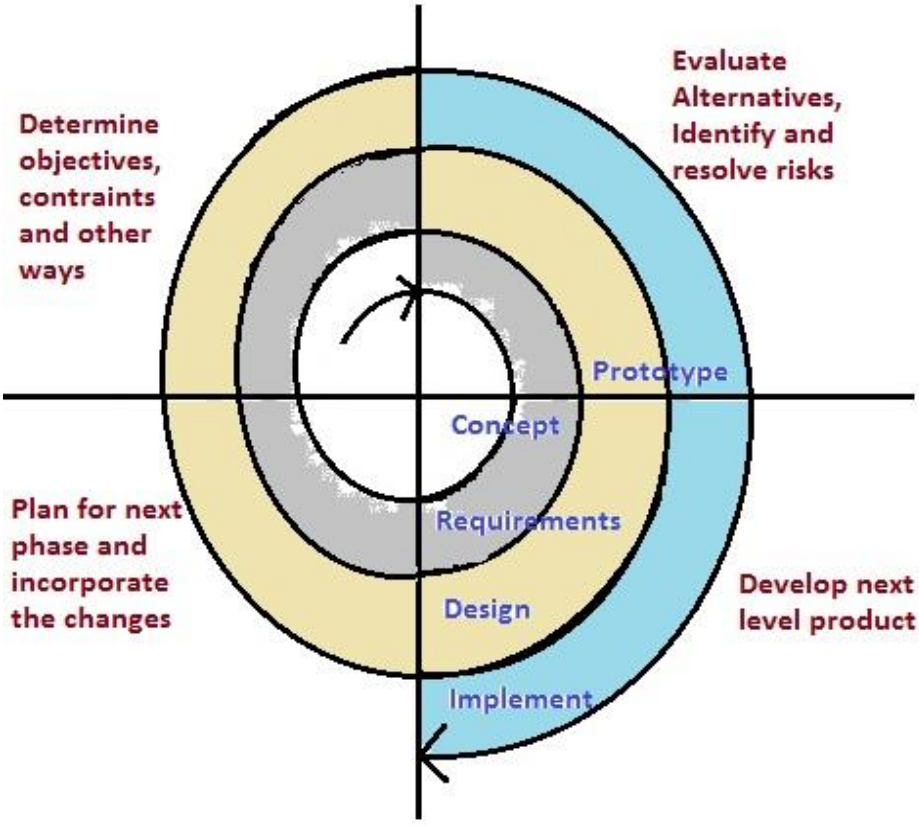

FIG 3.1.4 Spiral Model

The steps for Spiral Model can be generalized as follows:

- The new system requirements are defined in as much details as possible.
- This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
- A preliminary design is created for the new system.
- A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
- A second prototype is evolved by a fourfold procedure:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning a designing the second prototype.
4. Constructing and testing the second prototype.

- At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involve development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer's judgment, result in a less-than-satisfactory final product.
- The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
- The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
- The final system is constructed, based on the refined prototype.
- The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

Machine Learning Process

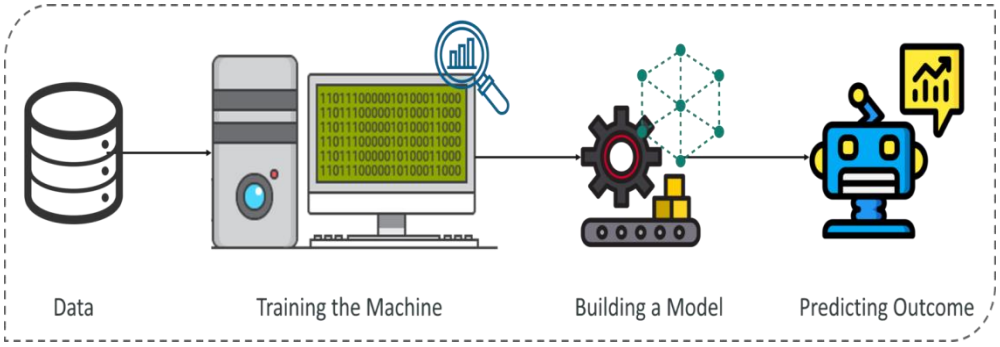

FIG 3.1.5 Machine Learning Process

How does Machine Learning Work?

Machine Learning algorithm is trained using a training data set to create a model. When new input data is introduced to the ML algorithm, it makes a prediction on the basis of the model.

The prediction is evaluated for accuracy and if the accuracy is acceptable, the Machine Learning algorithm is deployed. If the accuracy is not acceptable, the Machine Learning algorithm is trained again and again with an augmented training data set.

The Machine Learning process involves building a Predictive model that can be used to find a solution for a Problem Statement. To understand the Machine Learning process let's assume that you have been given a problem that needs to be solved by using Machine Learning.

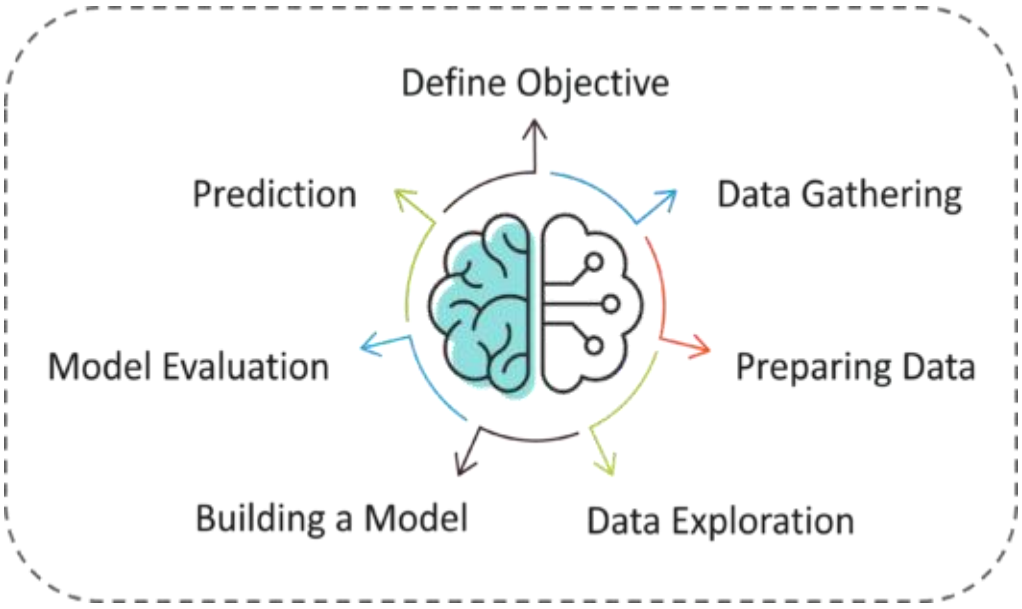

FIG 3.1.6 Working of machine learning

The below steps are followed in a Machine Learning process:

Step 1: Define the objective of the Problem Statement

At this step, we must understand what exactly needs to be predicted. In our case, the objective is to predict the possibility of rain by studying weather conditions. At this stage, it is also essential to take mental notes on what kind of data can be used to solve this problem or the type of approach you must follow to get to the solution.

Step 2: Data Gathering

At this stage, you must be asking questions such as,

- What kind of data is needed to solve this problem?
- Is the data available?
- How can I get the data?

Once you know the types of data that is required, you must understand how you can derive this data. Data collection can be done manually or by web scraping. However, if you're a beginner and you're just looking to learn Machine

Learning you don't have to worry about getting the data. There are 1000s of data resources on the web, you can just download the data set and get going.

Coming back to the problem at hand, the data needed for weather forecasting includes measures such as humidity level, temperature, pressure, locality, whether or not you live in a hill station, etc. Such data must be collected and stored for analysis.

Step 3: Data Preparation

The data you collected is almost never in the right format. You will encounter a lot of inconsistencies in the data set such as missing values, redundant variables, duplicate values, etc. Removing such inconsistencies is very essential because they might lead to wrongful computations and predictions. Therefore, at this stage, you scan the data set for any inconsistencies and you fix them then and there.

Step 4: Exploratory Data Analysis

Grab your detective glasses because this stage is all about diving deep into data and finding all the hidden data mysteries. EDA or Exploratory Data Analysis is the brainstorming stage of Machine Learning. Data Exploration involves understanding the patterns and trends in the data. At this stage, all the useful insights are drawn and correlations between the variables are understood.

For example, in the case of predicting rainfall, we know that there is a strong possibility of rain if the temperature has fallen low. Such correlations must be understood and mapped at this stage.

Step 5: Building a Machine Learning Model

All the insights and patterns derived during Data Exploration are used to build the Machine Learning Model. This stage always begins by splitting the data set into two parts, training data, and testing data. The training data will be used to build and analyze the model. The logic of the model is based on the Machine Learning Algorithm that is being implemented.

Choosing the right algorithm depends on the type of problem you're trying to solve, the data set and the level of complexity of the problem. In the upcoming sections, we will discuss the different types of problems that can be solved by using Machine Learning.

Step 6: Model Evaluation & Optimization

After building a model by using the training data set, it is finally time to put the model to a test. The testing data set is used to check the efficiency of the model and how accurately it can predict the outcome. Once the accuracy is calculated, any further improvements in the model can be implemented at this stage. Methods like parameter tuning and cross-validation can be used to improve the performance of the model.

Step 7: Predictions

Once the model is evaluated and improved, it is finally used to make predictions. The final output can be a Categorical variable (eg. True or False) or it can be a Continuous Quantity (eg. the predicted value of a stock).

In our case, for predicting the occurrence of rainfall, the output will be a categorical variable.

4. MODULE DESCRIPTION

4.1 Preprocessing

The image is loaded as the .mat format and the image is in the form of MRI. MRI image is in either RGB or gray scale. The image is resized and converted to grey if the image was in color. After that DWT (Discrete Wavelet Transform) will be applied.

4.2 Feature Extraction

In this part, we extract features of images to use in classification part. For feature extraction and feature selection, we use convolution neural network (CNN). As previously described to classification the images in the first step, CNN extracts the features of the dataset and perform the classification based on these features. In KE-CNN method we use CNN for feature extraction. Our configuration for CNN is the same model used in the previous method for convolutional neural networks. In this configuration, we have four convolution layers, four maxpooling layers with 1 fully connected layer.

4.3 Image Segmentation

It is a labelling process for each pixel in a medical image data set for indicating the type of tissue or the structure of anatomy. The labels that result from this approach have a vast area of applications in visualization and medical research. The method of dividing an image into sets of pixels which is also called super pixels is the image segmentation. The chief objective of segmentation is to identify the tumor's location. It is along chosen by the pixel powers themselves, yet in addition by the neighboring pixel powers and locations. Thought of these neighboring pixels extraordinarily controls the impact of noise.

4.4 Implementation Methodology

The entire system is divided into three major stages:

A. Preprocessing: MRI images often suffer from bias field distortion, causing tissue intensities to vary across the image. We address this using:

- N4ITK Method for bias field correction
- Nyúl's Intensity Normalization to align intensity scales across different MRI scans

Steps:

1. Learn intensity landmarks from the training set.
2. Normalize original intensities to match these landmarks.
3. Compute the mean and standard deviation across training patches.
4. Normalize patches to have zero mean and unit variance.

This ensures contrast and intensity consistency across subjects.

B. Convolutional Neural Network (CNN): CNN plays a central role in feature extraction and classification:

- **Convolutional Layers:** Apply kernels over the image to generate feature maps.
- **Pooling:** Max pooling layers reduce computational complexity while preserving important features.
- **Regularization:** Dropout is used to prevent overfitting.
- **Data Augmentation:** We rotate training patches by multiples of 90° to expand the dataset.
- **Loss Function:** Categorical Cross-entropy is minimized.
- **Architecture:** Separate CNN architectures for LGG (Low-Grade Glioma) and HGG (High-Grade Glioma), considering their structural variability.
- **Training:** Optimization using Stochastic Gradient Descent with Nesterov's Accelerated Momentum. Learning rate decays after each epoch.

Important CNN design choices include:

- Xavier Initialization for weights
- Leaky ReLU Activation
- Overlapping Pooling (3×3 kernels, 2×2 strides)
- Softmax at the final layer for classification.

C. Post-processing: To eliminate noise and false positives, small segmented clusters below a threshold size are removed, ensuring a cleaner and more accurate segmentation result.

4.5 Development Phases

1. Data Collection: Data must be of high quality and can be sourced from trusted repositories like Kaggle, UCI ML Repository.

2. Data Preparation: Raw collected data is cleaned, structured, and formatted into usable datasets. Example: Images converted into pixel matrices

3. Input: Prepared data is converted into machine-readable formats (e.g., numeric arrays from images, text, audio, video).

4. Processing: ML algorithms are applied to perform learning and model building.

5. Output: Meaningful outputs (reports, graphs, segmented images) are generated.

6. Storage: The trained models, outputs, and metadata are stored for future use.

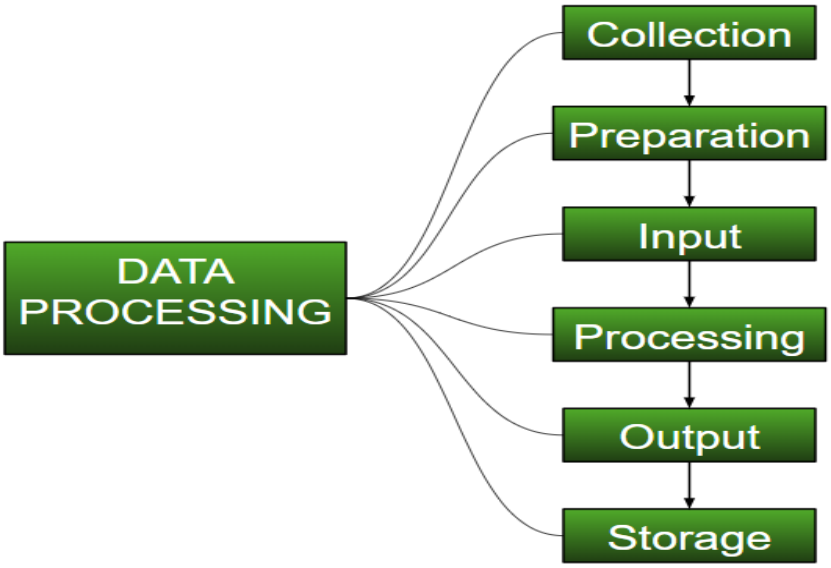

FIG:4.5.1 Development Phases

4.6. Data Preprocessing for Machine learning in Python

• Pre-processing refers to the transformations applied to our data before feeding it to the algorithm.

• Data Preprocessing is a technique that is used to convert the raw data into a clean data set. In other words, whenever the data is gathered from different sources it is collected in raw format which is not feasible for the analysis.

Key Preprocessing Steps:

- **Rescaling:** Normalize feature scales using MinMaxScaler.
- **Binarization:** Apply thresholds using Binarizer.
- **Standardization:** Apply StandardScaler to convert distributions to zero mean and unit variance.

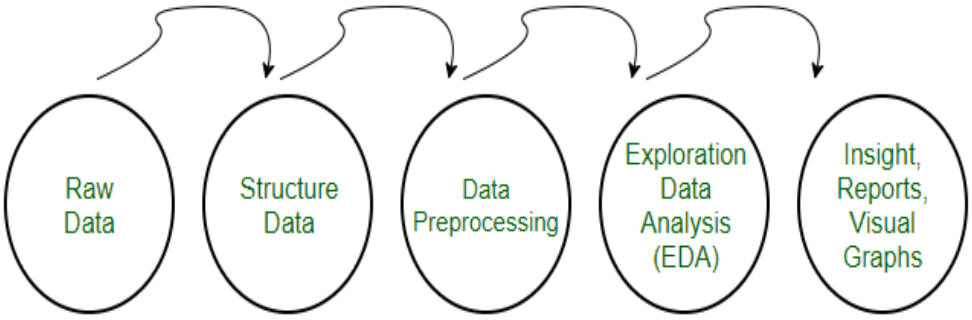

FIG:4.6.1 Data Preprocessing for Machine learning in Python

4.7 Data Cleaning

Data cleaning is a crucial part of machine learning, as it ensures the quality of the dataset, which can significantly impact model performance. Proper data cleaning can lead to better results, even with simpler algorithms.

Key Steps in Data Cleaning:

1. **Removal of Unwanted Observations:**
   - Eliminate duplicate or irrelevant data points.
   - Redundant data can skew results, while irrelevant data doesn't contribute to the problem.

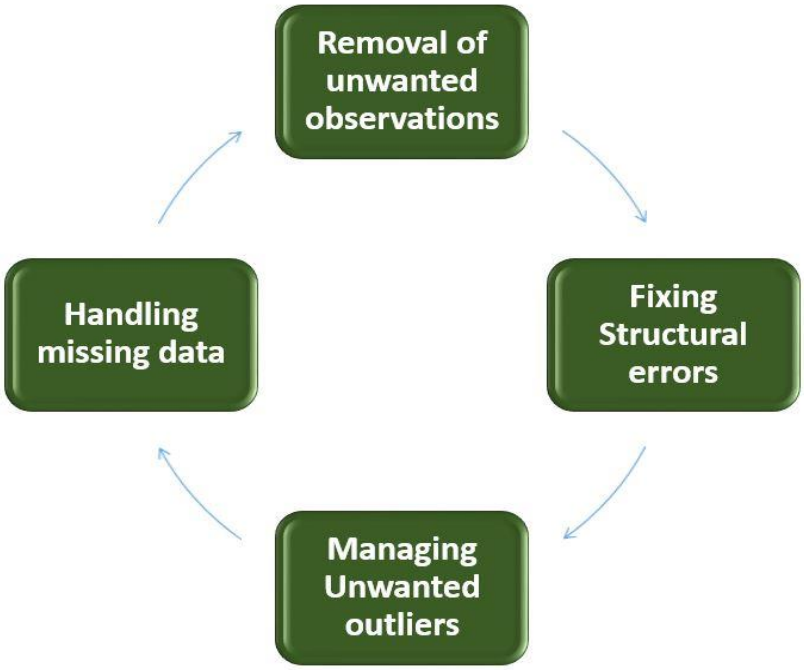

FIG:4.7.1 Data Cleaning

2. **Fixing Structural Errors:**
    - Correct errors like typos, inconsistent capitalization, and mislabeled classes.
    - Structural errors can mislead the model, reducing accuracy.
3. **Managing Outliers:**
    - Outliers can affect model performance, especially in linear regression.
    - Decide whether to remove outliers based on their relevance to the data.
4. **Handling Missing Data:**
    - Dropping missing data may lead to loss of information, while imputing values can introduce inaccuracies.
    - Flagging missing data and allowing the algorithm to estimate missingness can help preserve information.

5. SYSTEM DESIGN

System design is transition from a user oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough.

5.1 SYSTEM ARCHITECTURE

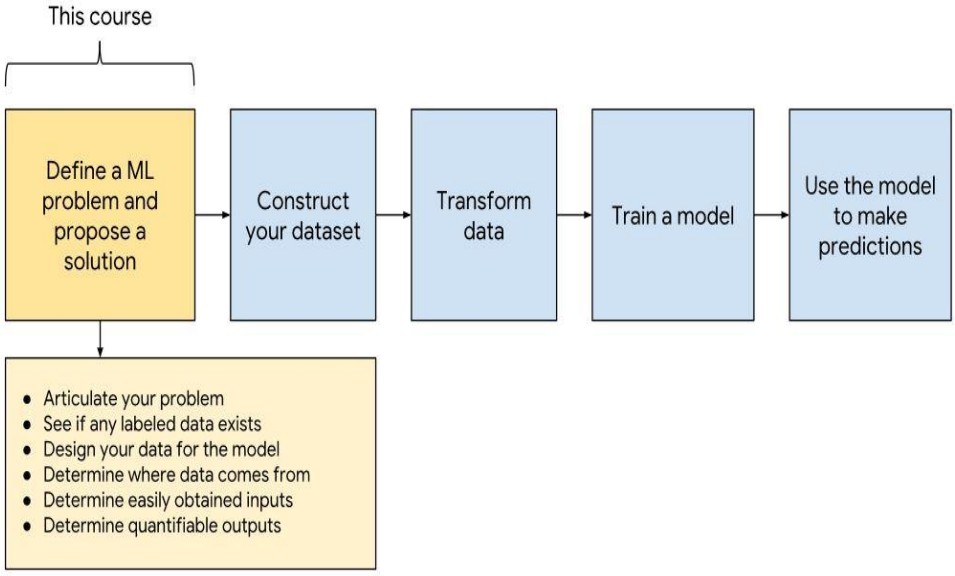

FIG:5.1.1 System architecture for brain tumor detection using CNN

5.2 INTRODUCTION TO UML

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic, semantic, and pragmatic rules. A UML system is represented using five different views that describe the system from a distinctly different perspectives. Each view is defined by a set of the diagram, which is as follows:

1. User Model View: This view represents the system from the users' perspective. The analysis representation describes a usage scenario from the end-user's perspective.

2. Structural Model View: In this model, the data and functionality are arrived from inside the system. This model view models the static structures.

3. Behavioral Model View: It represents the dynamic of behavior as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

4. Implementation Model View: In this view, the structural and behavioral parts of the system are represented as they are to be built.

5. Environmental Model View: In this view, the structural and behavioral aspects of the environment in which the system is to be implemented are represented.

5.3 UML DIAGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.. The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

5.3.1 USECASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

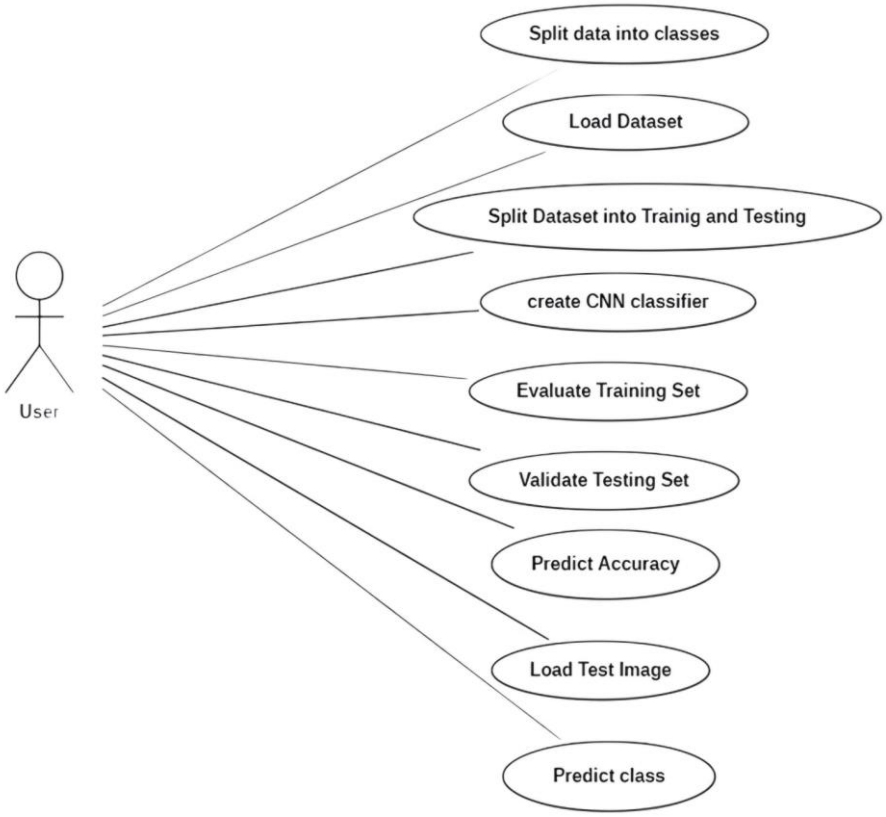

Diagram 5.3.1: USECASE DIAGRAM

5.3.2 SEQUENCE DIAGRAM

Sequence Diagrams Represent the objects participating the interaction horizontally and time vertically. A Use Case is a kind of behavioral classifier that represents a declaration of an offered behavior. Each use case specifies some behavior, possibly including variants that the subject can perform in collaboration with one or more actors. Use cases define the offered behavior of the subject without reference to its internal structure. These behaviors, involving interactions between the actor and the subject, may result in changes to the state of the subject and communications with its environment.

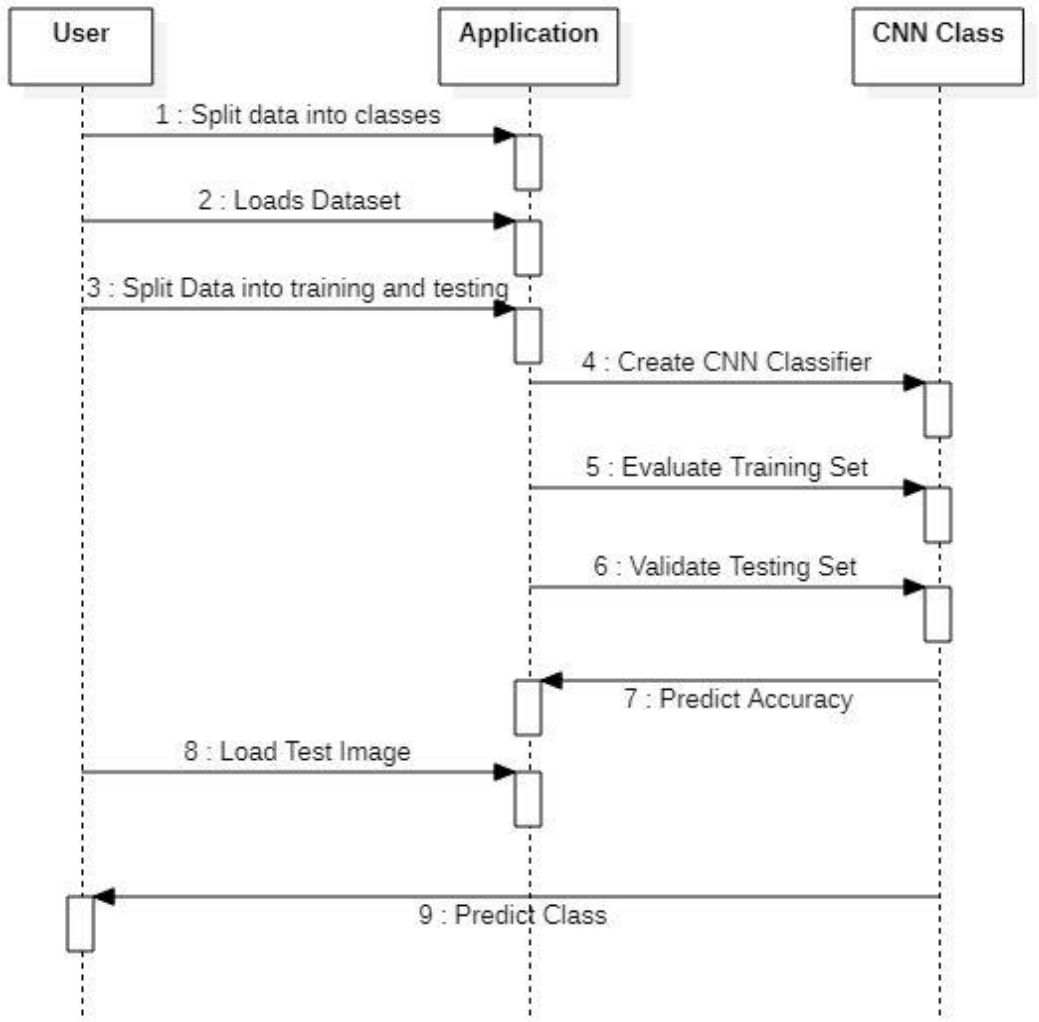

Diagram 5.3.2 SEQUENCE DIAGRAM

5.3.3 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of Workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

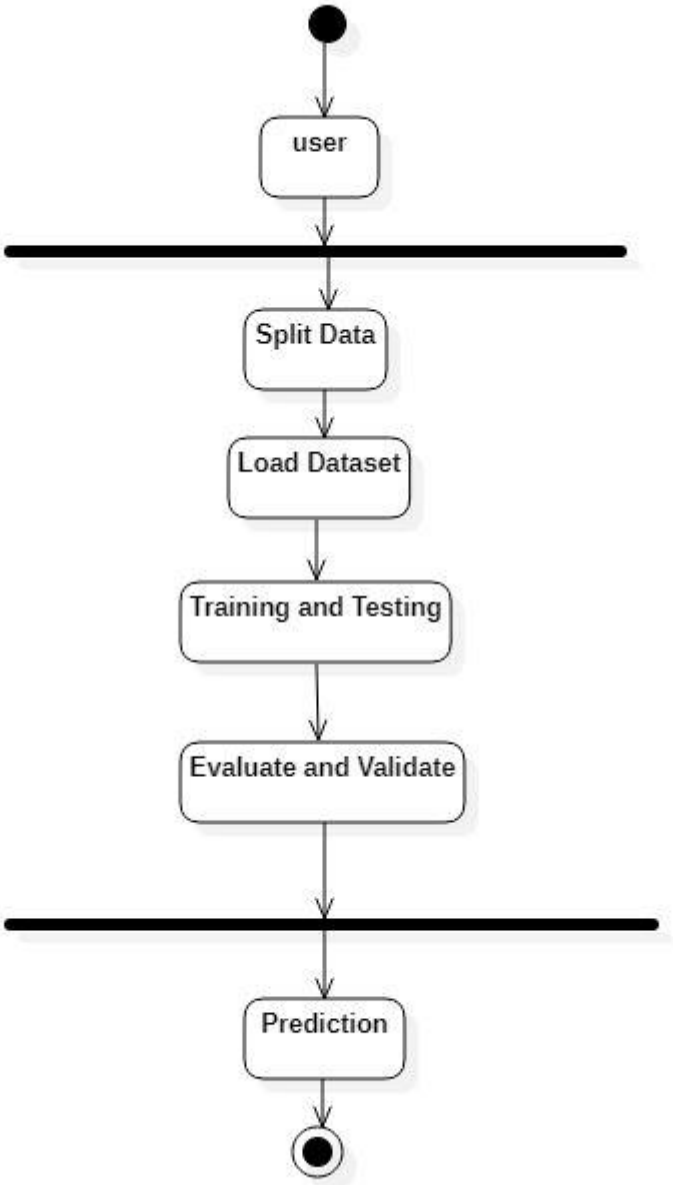

Diagram 5.3.3 ACTIVITY DIAGRAM

5.3.4 STATE CHART DIAGRAM

A state chart diagram is a type of behavioral diagram in UML (Unified Modeling Language) that shows the possible states an object can be in and the transitions between those states.

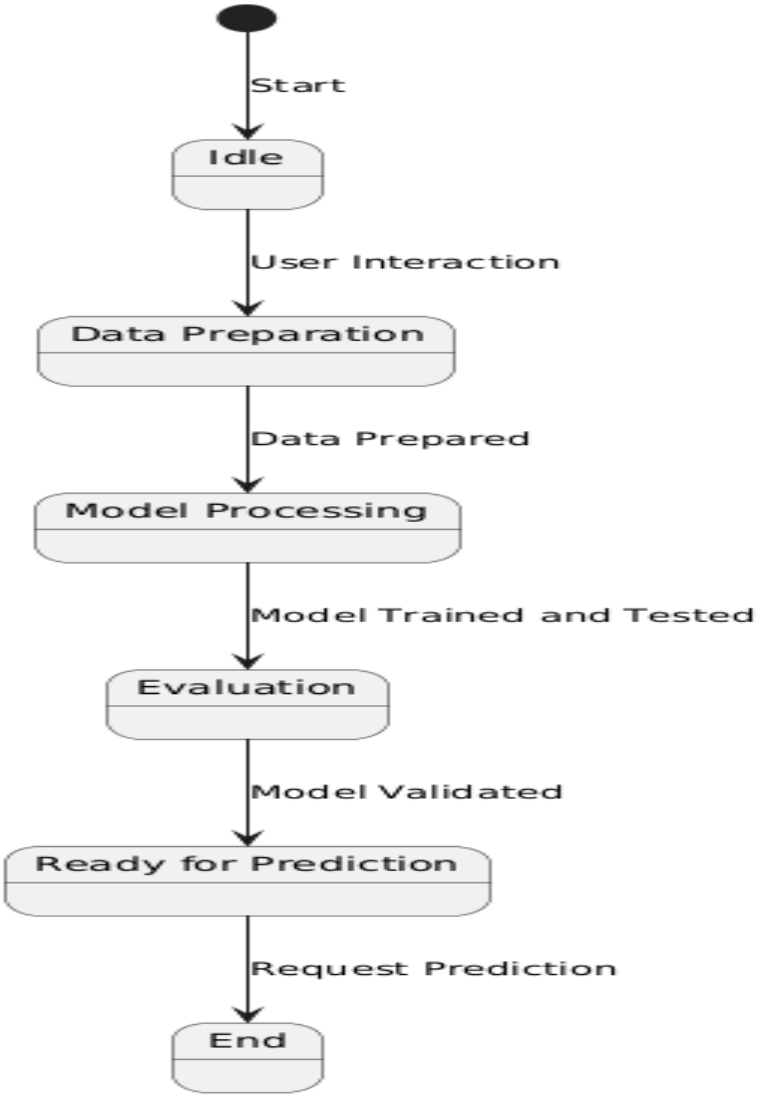

Diagram 5.3.4 STATE CHART DIAGRAM

5.3.5 COMPONENT DIAGRAM

A component diagram is a type of UML (Unified Modeling Language) diagram that shows the relationships between components of a system.

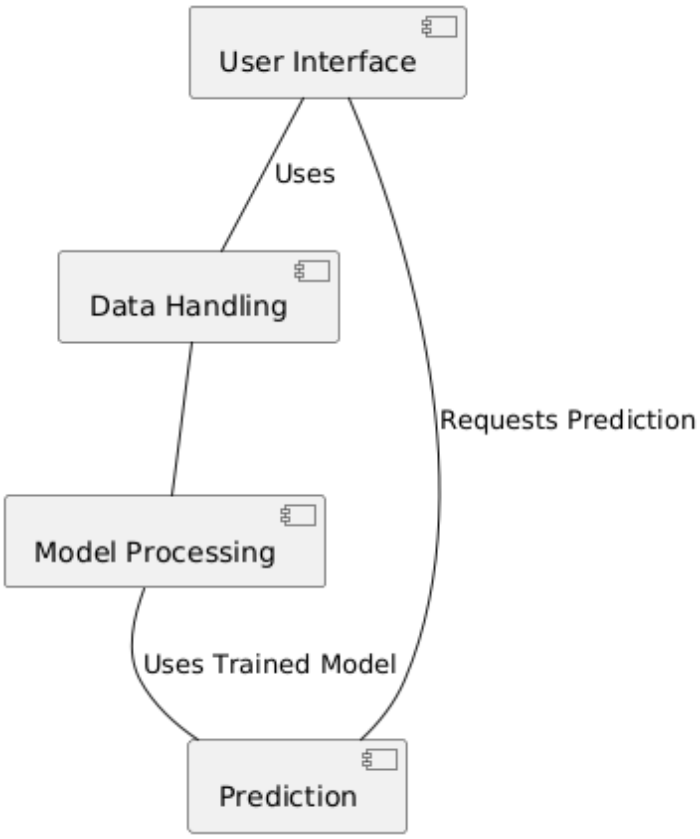

Diagram 5.3.5 COMPONENT DIAGRAM

5.3.6 CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

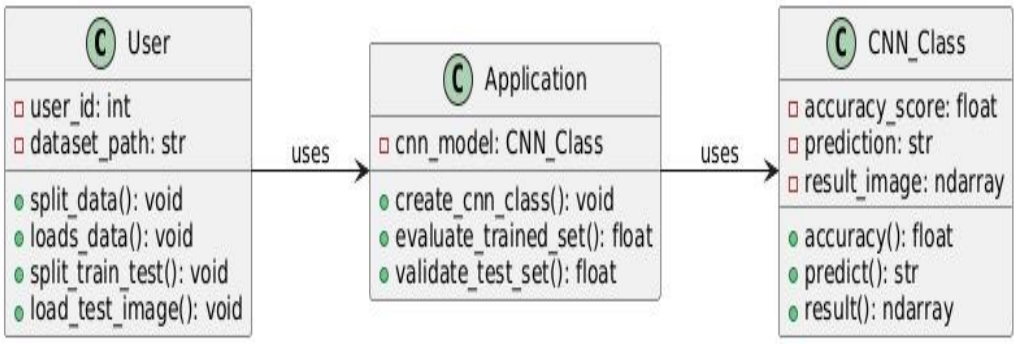

Diagram 5.3.6 CLASS DIAGRAM

5.3.7 DEPLOYMENT DIAGRAM

There may be more steps involved, depending on what specific requirements you have, but below are some of the main steps:

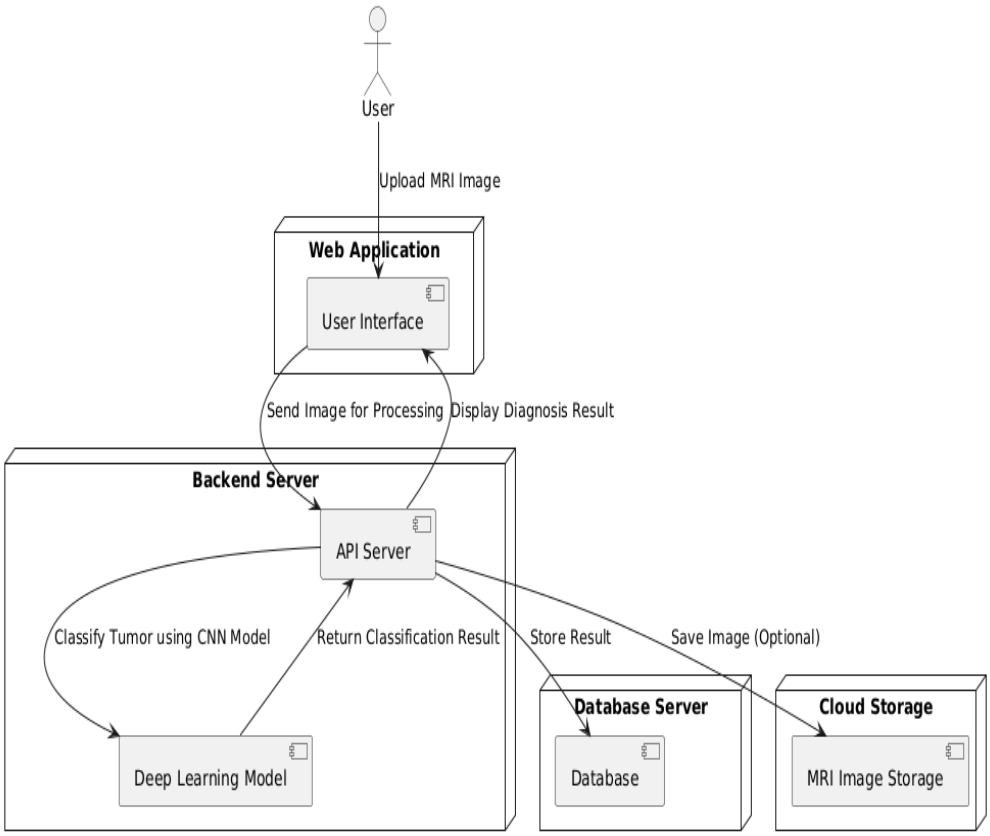

Diagram 5.3.7 DEPLOYMENT DIAGRAM

6. CODING

6.1 SAMPLE CODE

```
from _future_ import division, print_function


from flask import Flask, request, render_template

from werkzeug.utils import secure_filename


from tensorflow.keras.layers import *

from tensorflow.keras.losses import *

from tensorflow.keras.models import *

from tensorflow.keras.metrics import *

from tensorflow.keras.optimizers import *

from tensorflow.keras.applications import *

from tensorflow.keras.preprocessing.image import load_img


import numpy as np

from PIL import Image, ImageEnhance

import random

import os


app = Flask(_name_)
```

```
IMAGE_SIZE = 128

unique_labels=['glioma', 'meningioma', 'notumor', 'pituitary']


def augment_image(image):


    image = Image.fromarray(np.uint8(image))

    image = ImageEnhance.Brightness(image).enhance(random.uniform(0.8,1.2))

    image = ImageEnhance.Contrast(image).enhance(random.uniform(0.8,1.2))

    image = ImageEnhance.Sharpness(image).enhance(random.uniform(0.8,1.2))

    image = np.array(image)/255.0

    return image



def open_images(paths):

    images = []

    for path in paths:

        image = load_img(path, target_size=(IMAGE_SIZE, IMAGE_SIZE))

        image = augment_image(image)

        images.append(image)

    return np.array(images)
```

```
def model_predict(img_path):


    base_model = VGG16(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3),
include_top=False, weights='imagenet')
    # Set all layers to non-trainable
    for layer in base_model.layers:
        layer.trainable = False
    # Set the last vgg block to trainable
    base_model.layers[-2].trainable = True
    base_model.layers[-3].trainable = True
    base_model.layers[-4].trainable = True


    model = Sequential()
    model.add(Input(shape=(IMAGE_SIZE, IMAGE_SIZE, 3)))
    model.add(base_model)
    model.add(Flatten())
    model.add(Dropout(0.3))
    model.add(Dense(128, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(len(unique_labels), activation='softmax'))
    model.compile(optimizer=Adam(learning_rate=0.0001),
                  loss='sparse_categorical_crossentropy',
                  metrics=['sparse_categorical_accuracy'])
```

```
    model.load_weights("model.h5")


    prediction_class = model.predict(open_images([img_path]))
    prediction_class = np.argmax(prediction_class, axis=1)
    print(prediction_class)


    return unique_labels[prediction_class[0]]


@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')



@app.route('/predict/', methods=['POST'])
def predict():
    if request.method == 'POST':
        f = request.files['file']
        print("in post")
        basepath = os.path.dirname(_file_)
        file_path = os.path.join(
            basepath, 'static/images', secure_filename(f.filename))
        print("filepath:",file_path)
```

```
        f.save(file_path)

        print("after save")

        return
render_template('pred.html',pred=model_predict(file_path),f_name=f.filename
)


    return None



if _name_ == '_main_':

    app.run(debug=True)
```

7. TESTING

7.1 TEST CASES

Tested	Test name	Inputs	Expected output	Actual Output	Result
1	Splitting data into classes	data	input taken	successfully splitted	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
2	load dataset	dataset	dataset loaded	successfully loaded	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
3	Splitting dataset into training and testing	dataset	splitted to training and testing	successfully splitted to train data and test data	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
4	load cnn classifier	sklearn module	model created	successfully model created	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
5	training set evaluation	train data	training done	successfully trained	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
6	Validating test dataset	test data	test dataset validated	successfully validated	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
7	getting accuracy	test dataset	accuracy in percentage	successfully got accuracy	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
8	load test image	test image	detecting harmful or not	successfully detected	pass

Tested	Test name	Inputs	Expected output	Actual Output	Result
9	print result	test image	printing malignant or begant	successfully printed	pass

Table – 7. 1 TEST CASES

8. OUTPUT SCREENS

8.1 HOME PAGE

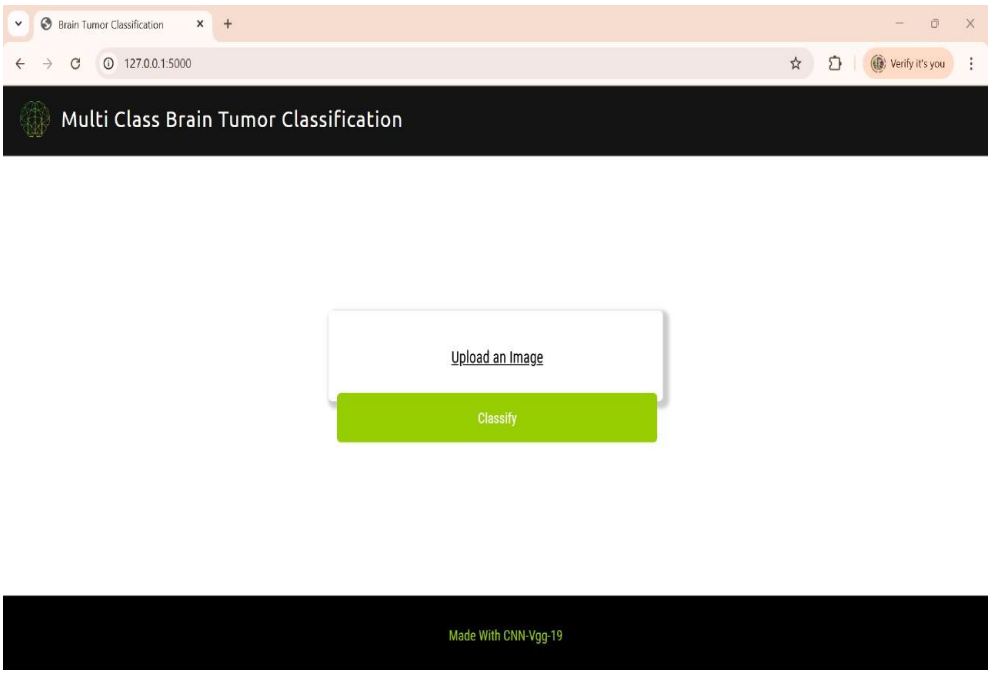

FIG: 8.1 Home Page

8.2 SELECT IMAGE

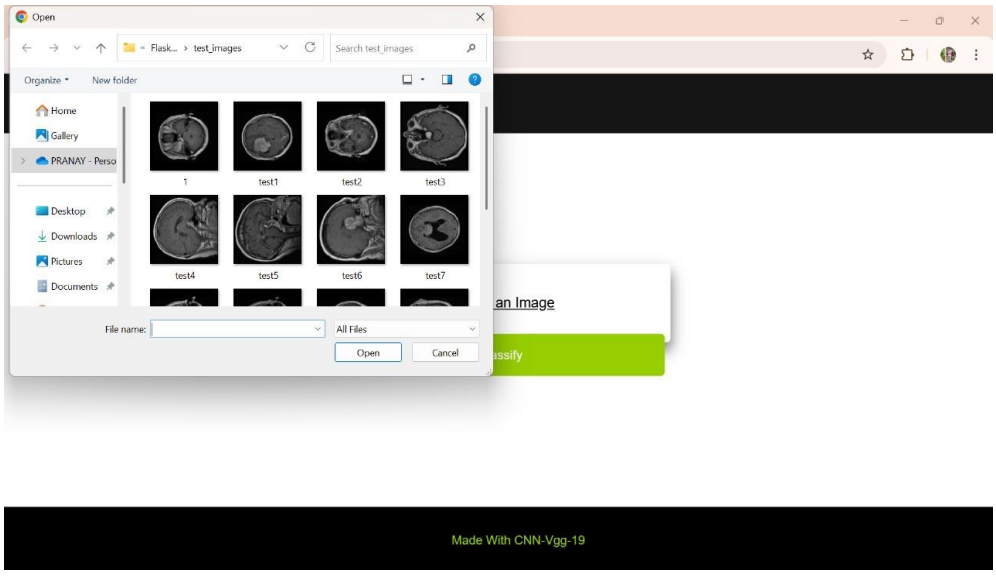

FIG:8.2 Select Image

8.3 TUMOR DETECTED

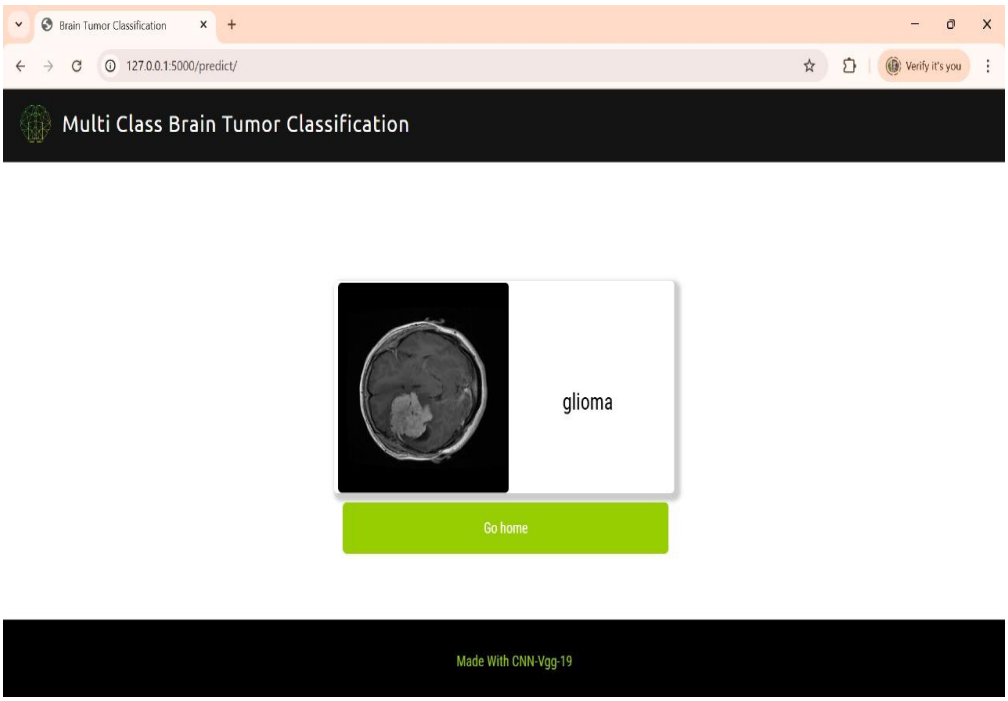

FIG 8.3 TUMOR DETECTED

8.4 NO TUMOR DETECTED

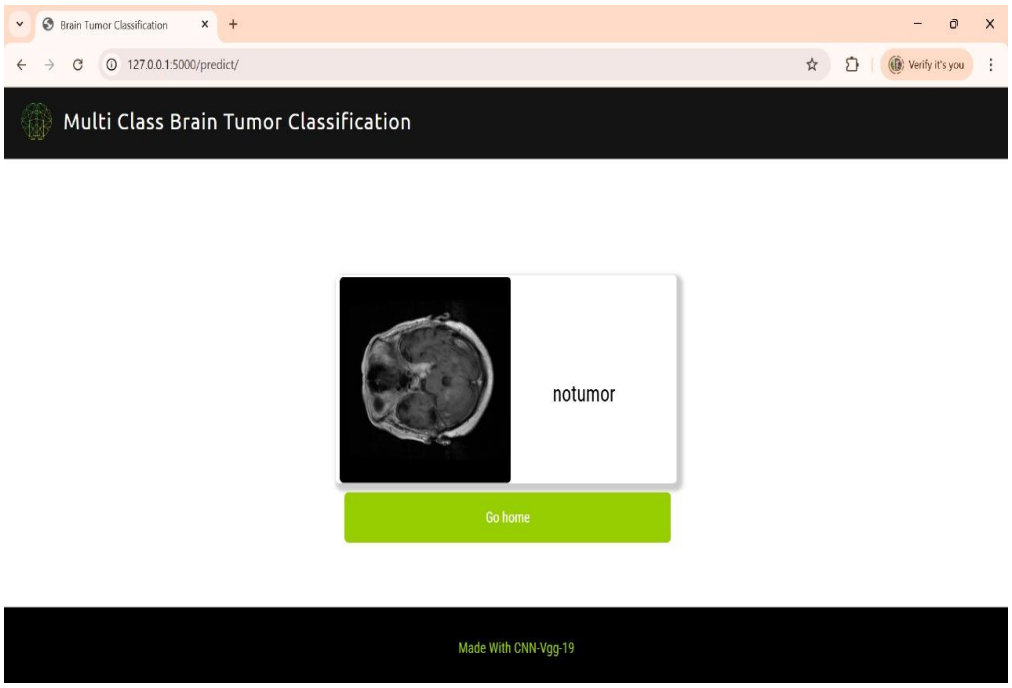

FIG:8.4 NO TUMOR DETECTED

9. CONCLUSION

We propose a novel CNN-based method for the segmentation of brain tumors in MRI images, consisting of three main stages: pre-processing, classification via CNN, and post-processing. Initially, MRI images are normalized by computing the mean intensity value and standard deviation across all training patches extracted for each sequence, and then patches are normalized to have zero mean and unit variance. The CNN architecture effectively relates both local and global features, enabling accurate segmentation. Training and testing speeds are improved through the use of max pooling, maxout, and dropout techniques, which also complement the learning process. Furthermore, reducing features in the fully connected layers not only increases speed but also minimizes overfitting by lowering the number of parameters. The results demonstrate that the proposed method accurately detects enhancing tumors and precisely specifies the tumor region. This accurate segmentation not only aids clinical diagnosis but also contributes to improving the patient's lifetime

10. REFERENCES

[1] S. Bauer et al., “A survey of MRI-based medical image analysis for brain tumor studies,” Phys. Med. Biol., vol. 58, no. 13, pp. 97–129, 2013.

[2] D. N. Louiset al., “The 2007 who classification of tumors of the central nervous system,” Acta Neuropathologica, vol. 114, no. 2, pp. 97–109, 2007.

[3]E.G.VanMeiretal.,“Exciting new advances in neuro-oncology:The avenue to accrue for malignant glioma, ”CA,CancerJ.Clinicians,vol. 60, no. 3, pp. 166–193, 2010.

[4] C.-H. Lee et al., “Segmenting brain tumors using pseudo-conditional random fields,” in Medical Image Computing and Comput.-Assisted Intervention-MICCAI 2008. New York: Springer, 2008, pp. 359–366.

[5] B. Menze et al., “The multimodal brain tumor image segmentation benchmark (BRATS),” IEEE Trans. Med. Imag., vol. 34, no. 10, pp. 1993–2024, Oct. 2015.

[6] N. J. Tustisonet al., “N4ITK: Improved n3 bias correction,” IEEE Trans.Med.Imag.,vol.29,no.6,pp.1310–1320,Jun.2010.

[7] L.G.Nyúl,J.K.Udupa,andX.Zhang,“Newvariantsofamethodof MRI scale standardization,” IEEE Trans. Med. Imag., vol. 19, no. 2, pp. 143–150, Feb. 2000.

[8] M. Prastawa et al., “A brain tumor segmentation framework based on outlier detection,” Med. Image Anal., vol. 8, no. 3, pp. 275–283, 2004.

[9]B.H.Menzeetal.,“A generative model for brain tumor segmentation in multi-modal images,” in Medical Image Computing and Comput.Assisted Intervention-MICCAI2010.NewYork:Springer,2010,pp. 151–159.

[10] A. Gooya et al., “GLISTR: Glioma image segmentation and registration,” IEEE Trans. Med. Imag., vol. 31, no. 10, pp. 1941–1954, Oct. 2012.

[11] D. Kwon et al., “Combining generative models for multifocal glioma segmentation and registration,” in Medical Image Computing andComput.Assisted Intervention-MICCAI 2014. New York: Springer, 2014, pp. 763–770.

[12] B Kundan, Sangaralingam P. Combining Machine Learning and Deep Learning in the Retinopathy Diagnostic Algorithm for Enhanced Detection of DR and DME.J Neonatal Surg [Internet}.2025Apr.2[cited 2025Apr.9];14(5):128-40. Available from: https://www.jneonatalsurg.com/index.php/jns/article/view/2914