e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

INNOVATIVE CNN-BASED METHOD FOR DETECTING BRAIN TUMORS

Koppisetti Gowthami*1, G Aruna Rekha*2, Maroju Deepika*3, Kurapati Srujana*4

*1Post Graduate Student, Department Of Artificial Intelligence, Kakinada Institute Of Engineering & Technology (Approved By AICTE & Affiliated To JNT University Kakinada) Yanam Road, Korangi-533461, E.G. Dist. (A.P), India.

*2Professor, Department Of Artificial Intelligence, Kakinada Institute Of Engineering & Technology (Approved By AICTE & Affiliated To JNT University Kakinada) Yanam Road, Korangi-533461, E.G. Dist. (A.P), India.

*3,4Assistant Professor, Department Of ECE, Kakinada Institute Of Engineering & Technology (Approved By AICTE & Affiliated To JNT University Kakinada) Yanam Road, Korangi-533461, E.G. Dist. (A.P), India.

DOI : https://www.doi.org/10.56726/IRJMETS60069

ABSTRACT

Nowadays, tumors are the second leading cause of cancer, putting many patients at risk. The medical field urgently needs fast, automated, efficient, and reliable techniques to detect tumors, especially brain tumors, as early detection is crucial for effective treatment. Tumors are excess cells growing uncontrollably, depriving healthy cells of nutrients and leading to brain failure. Currently, doctors manually examine MR images to locate brain tumors, a process that is time-consuming and often inaccurate. By using Deep Learning architectures like Convolutional Neural Networks (CNN), we can improve brain tumor detection. This model predicts whether a tumor is present in an image, and if so, classifies it as one of three types: Glioma, Meningioma, or Pituitary tumor. If no tumor is detected, the model returns 'no tumor'. This approach enables doctors to provide proper treatment and save more patients.

Keywords: Tumor Detection, Brain Tumor, Deep Learning, Convolutional Neural Network (CNN), Medical Imaging.

I. INTRODUCTION

1.1 BRAIN TUMOUR DETECTION SYSTEM:

The brain is the most critical organ in the human body, and one common cause of its dysfunction is a brain tumor, which consists of excess cells growing uncontrollably and depriving healthy cells of nutrients, leading to brain failure. Currently, doctors manually examine MRI images to locate brain tumors, a time-consuming and often inaccurate method. Brain cancer is a serious disease causing many deaths, making early detection crucial. This project aims to develop a system that uses computer-based procedures and Convolutional Neural Networks (CNN) to detect and classify brain tumors from MRI images. The process involves image preprocessing, segmentation, max-pooling, feature extraction, and classification. By using these image processing and neural network techniques, the system improves the accuracy and efficiency of detecting and classifying brain tumors.

Overview of brain tumor detection: The main part of the human nervous system is the brain, located in the head and protected by the skull. The brain controls all parts of the human body, allowing individuals to accept and endure various environmental conditions, perform actions, and share thoughts and feelings. This section provides a basic overview of the brain's structure for foundational understanding.

The below diagram shows the human brain, labeled with the cerebellum and spinal cord. The cerebellum helps with coordination, movement, and balance. The spinal cord controls basic body functions like breathing, heart rate, and digestion.

Brain tumors are classified into two main types: primary (benign) and secondary (malignant). Primary brain tumors, such as gliomas, originate from non-neuronal brain cells called astrocytes and grow slowly. Although less aggressive, they can put pressure on the brain, causing it to malfunction. Secondary brain tumors are more aggressive and spread quickly to other tissues. They originate from other parts of the body and contain

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

metastatic cancer cells that spread to areas like the brain and lungs. Secondary brain tumors are often caused by cancers of the lungs, kidneys, and bladder.

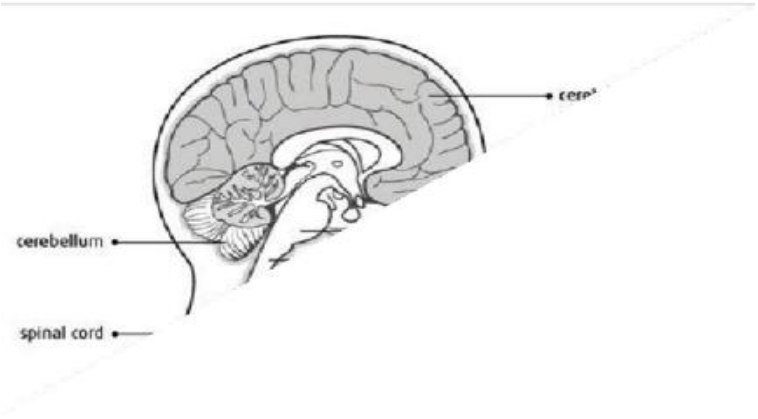

Fig.1: Basic Structure of the Human Brain

Magnetic Resonance Imaging (MRI): Raymond V. Damadian invented the first magnetic image in 1969, and in 1977, the first MRI image for the human body was created, becoming the most accurate technique. MRI allows detailed visualization of the brain's internal structure and various body tissues, providing better-quality images compared to X-rays and CT scans. MRI is particularly effective for detecting brain tumors, with different types of MRI images, such as T1-weighted, T2-weighted, and FLAIR (Fluid Attenuated Inversion Recovery)-weighted, used to map tumor-induced changes.

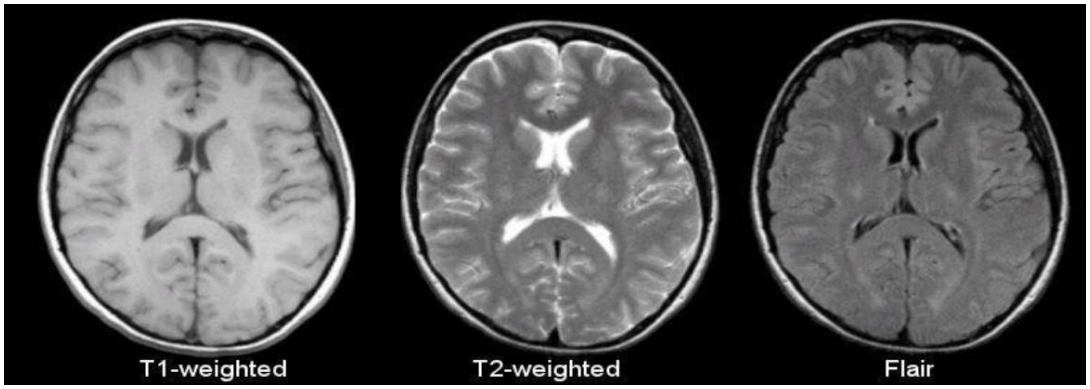

Fig 2: T1, T2 and Flair image

The above figure shows T1-weighted and T2-weighted MRI scans, which highlight different characteristics of the brain. The most common MRI sequences are T1-weighted and T2-weighted. In T1-weighted scans, only fat appears bright, whereas in T2-weighted scans, both fat and water appear bright. T1-weighted scans have a short repetition time (TR), while T2-weighted scans have both a long echo time (TE) and a long repetition time (TR). TE and TR are pulse sequence parameters, with TE standing for time to echo and TR for repetition time, both measured in milliseconds (ms). The echo time (TE) is the duration from the center of the RF pulse to the center of the echo, and TR is the interval between the TE repeating series of pulses and echoes, as shown in the figure.

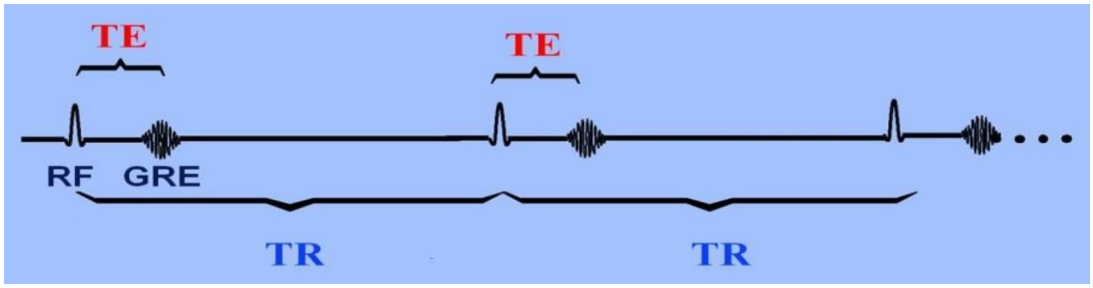

Fig 3: Graph of TE and TR

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

In the previous figure, TR (repetition time) is the interval between applying a pulse sequence in an MRI scan, and TE (echo time) is the time between the initial pulse and receiving an echo signal. The FLAIR sequence, commonly used alongside T1- and T2-weighted sequences, is similar to T2-weighted images but with significantly longer TE and TR times, as shown in the table.

	TR (msec)	TE (msec)
T1-Weighted (short TR and TE)	500	14
T2-Weighted (long TR and TE)	4000	90
Flair (very long TR and TE	9000	114

Fig 4: Table of TR and TE time

1.2 APPLICATION:

The primary goal of this application is tumor identification, aiming to facilitate prompt treatment and protect lives at risk. It benefits both doctors and patients by offering a faster, more accurate, and more efficient alternative to manual identification methods. This application addresses the limitations of manual identification, ensuring timely and effective healthcare interventions**.**

1.3 OBJECTIVE:

The software aims to help doctors identify tumors:

- Save patients' time.
- Provide timely solutions in the early stages.
- Facilitate prompt consultations**.**

1.4 MOTIVATION:

The primary motivation behind brain tumor detection is not only to identify tumors but also to classify their types. This capability is crucial for determining whether a tumor is present and its specific classification, aiding in medical diagnoses and treatments. The project focuses on developing a computer-based system that uses Convolutional Neural Network algorithms to detect tumor regions and classify tumor types from MRI images of various patients**.**

II. LITERATURE SURVEY AND WORKFLOWS

[1] Deepa and Akansha Singh reviewed recent research on brain tumor detection and segmentation, highlighting various techniques employed by researchers using MRI images. The review underscores that automating brain tumor detection and segmentation is a highly active research area, reflecting ongoing efforts to enhance medical imaging technologies.

[2] Devendra Somwanshi, Ashutosh Kumar, Pratima Sharma, and Deepika Joshi explore different entropy functions for tumor segmentation and detection from various MRI images. It examines how threshold values vary based on the specific entropy definition used, influencing the quality and accuracy of the segmented results.

[3] Luxit Kapoor and Sanjeev Thakur surveyed various techniques in Medical Image Processing specifically used to detect brain tumors from MRI images. It lists and provides brief descriptions of these techniques based on current research. Among all the steps involved in tumor detection, segmentation is highlighted as the most crucial process.

[4] Mehdi Nazari, Seyed Mehdi Amin Tabatabaei, and Anousheh Rezaei reviewed the application of Deep Learning (DL) techniques, specifically Convolutional Neural Networks (CNNs), for detecting and segmenting brain tumors in MRI scans. It examines different CNN architectures employed in this context, highlighting their strengths and weaknesses.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

[5] Shervin Emami, Mohammad Amin Jafari, and Elham Rezakhani introduced a novel cascaded CNN architecture designed for brain tumor segmentation. It incorporates uncertainty quantification techniques to enhance the model's robustness and reliability in accurately identifying brain tumors with diverse characteristics.

[6] Sudhakar Deepak, P. Mohamed Ameer explore the application of transfer learning with Deep Convolutional Neural Networks (CNNs) for classifying brain tumors. It investigates the use of pre-trained models such as VGG16 and ResNet for extracting features and classifying brain tumors into different types.

[7] Hao Chen, Qi Dou, Lequan Yu, and Pheng-Ann Heng offer a thorough review of deep learning techniques utilized for segmenting brain tumors in MRI scans. It examines various CNN architectures, including U-Net and its adaptations, aimed at enhancing segmentation accuracy. Additionally, the paper addresses challenges such as limited data availability and class imbalance, proposing potential solutions to mitigate these issues.

[8] Mehmet Seçkin Yaşar and Fatih Özyılmaz examined the effectiveness of transfer learning using pre-trained CNN models such as VGG16 and ResNet for brain tumor classification. It investigates fine-tuning methods to adapt these models to brain tumor datasets and compares their performance against traditional machine-learning approaches.

[9] Shervin Samiei et al. introduced a novel hybrid deep learning architecture designed for brain tumor detection and grading. It integrates a Convolutional Neural Network (CNN) for feature extraction with a Recurrent Neural Network (RNN) to capture spatial dependencies within MRI scans.

OVERVIEW OF EXISTING WORKFLOW:

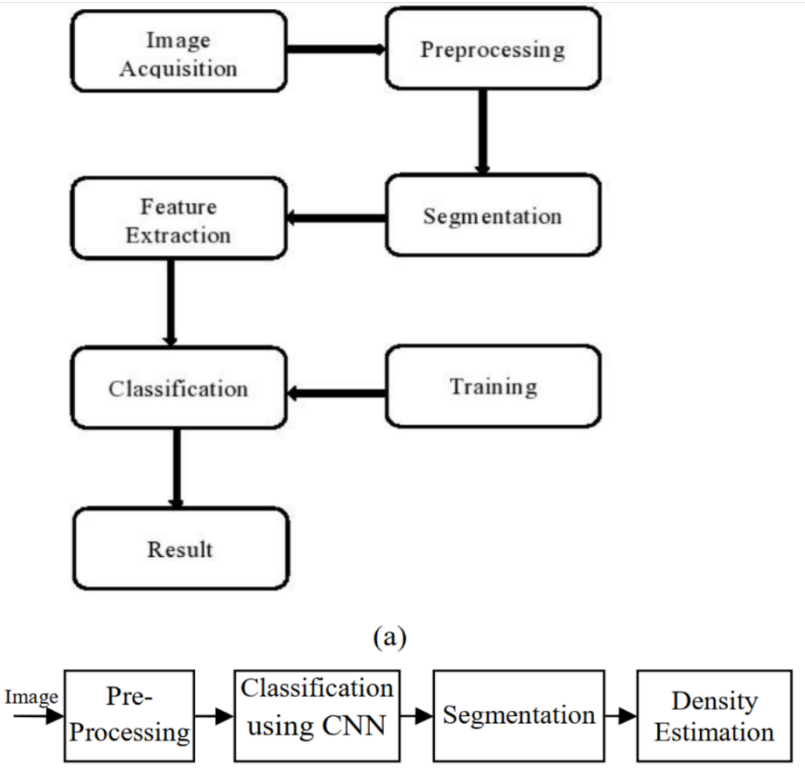

Fig 5: Existing workflow of brain tumor detection.

The above flowchart illustrates the existing workflow for brain tumor detection. It outlines the various stages of the process, beginning from image acquisition to obtaining the final result.

The process of acquiring images for brain tumor detection. Various biomedical imaging techniques such as Computed Tomography (CT) and Magnetic Resonance Imaging (MRI) are utilized, with MRI being particularly favored due to its ability to create detailed images based on water molecule detection. Recent advancements include the development of portable and miniaturized MRI machines, offering simpler alternatives to traditional scanning methods. The study uses an MRI dataset from Kaggle, curated by Navoneel Chakrabarty, which includes 98 normal and 155 abnormal brain images labeled 'yes' for tumors and 'no' for healthy images.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

Augmentation techniques were applied to increase sample diversity, incorporating rotations, shifts, brightness adjustments, and flips. This resulted in a final dataset of 980 normal and 1550 abnormal images used for analysis.

The preprocessing step aims to prepare brain images for subsequent processing. This process is tailored to the specific parameters of the data acquisition device used. If the raw data is in 3D, it is converted to grayscale or 2D. Median filtering is employed to reduce noise, which is beneficial for biomedical images. The dataset includes images with varying resolutions, which are standardized during augmentation through rotation and scaling. Histogram equalization is utilized to improve image quality, and a contrast-limited adaptive histogram equalization algorithm is applied to further enhance the images.

Image Segmentation is image processing, the goal is to partition a digital image into segments, separating a specific region of interest from the background. However, traditional segmentation techniques like thresholding and morphological operations may not effectively delineate tumor regions in brain images, as healthy tissues can have similar intensity levels. Instead, these methods are more suitable for separating the skull from brain images. An OTSU-based thresholding algorithm is employed to create a segmented mask of the skull, while the active contour method outlines the boundary of this region. A second segmentation stage can then be applied specifically to the Region of Interest (ROI) containing the tumor, although it may not yield accurate results for healthy images. This segmented image of the tumor region aids in studying its features and helps in density estimation for further analysis.

Feature extraction involves computing specific characteristics from data to illustrate disease behavior or symptoms. In classification tasks, the selection of these features significantly influences the accuracy and effectiveness of the model. Common features used include asymmetry, diameter, and border irregularity, which are crucial for distinguishing and analyzing disease patterns.

In the training and classification phase of disease detection from brain images, various machine-learning approaches are utilized. Artificial neural networks (ANNs) are effective for classification when features are extracted sequentially. ANNs assume each feature is independent of others. Deep learning techniques, particularly Convolutional Neural Networks (CNNs), are highly effective for classifying tumor images without requiring explicit segmentation. CNNs leverage deep neural networks to automatically learn and classify patterns in brain images, enhancing accuracy in disease detection tasks.

OVERVIEW OF PROPOSED WORKFLOW:

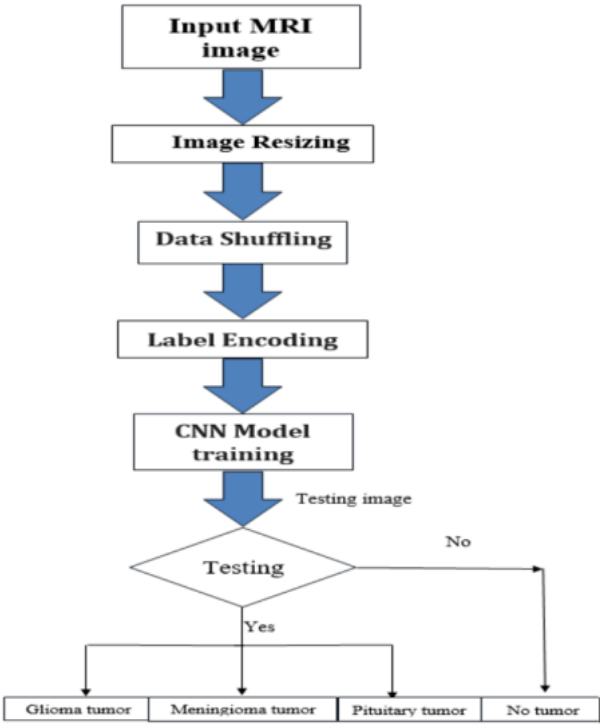

Fig 6: Proposed workflow of brain tumor detection

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

The depicted flowchart outlines the sequential steps involved in processing an input MRI image, including pre-processing, model training, testing, and final classification.

In the initial step, an MRI image of the human brain is taken as input. Subsequently, the image undergoes resizing to a uniform size of 150x150 pixels.

The dataset, including both training and testing images, is randomly shuffled to prevent the model from learning biases related to data sequence, ensuring robust and unbiased training. One Hot Label Encoding converts class labels (glioma tumor, meningioma tumor, no tumor, pituitary tumor) into numeric representations for effective processing by the model. The CNN model is trained using this dataset, where it learns to minimize a loss function by adjusting its parameters through forward and backward propagation. During forward propagation, the input image is processed through CNN layers, performing operations like convolution, activation, pooling, and fully connected computations to generate predictions. In backward propagation, the loss between predicted and actual labels is calculated using a loss function like cross-entropy loss. Gradients are computed using the chain rule of calculus and used to update model parameters via optimization algorithms like SGD, Adam, or RMSprop, enhancing the model's accuracy in classifying brain tumor images.

III. ALGORITHM - CONVOLUTIONAL NEURAL NETWORKS

Convolutional Neural Network Implementation can be done by the following five steps.

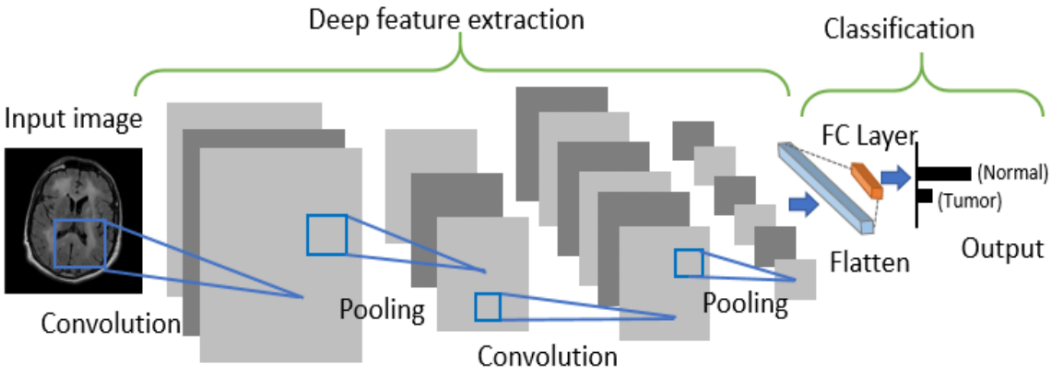

Fig 7: CNN architecture for brain tumor detection

3.1 PRE-PROCESSING:

Data normalization standardizes pixel values to a specific range, typically between 0 and 1 or -1 and 1, using methods like Min-Max Scaling or Z-score normalization, enhancing training stability. Resizing ensures all images match the network's input dimensions, using methods like nearest neighbor or bilinear interpolation for smoother results. Data augmentation optionally expands the dataset with variations like random cropping, horizontal flipping, and random rotations to prevent overfitting. For categorical labels, one-hot encoding converts them into vectors where only the true class element is set to one, aiding in accurate classification.

3.2 FORWARD PROPAGATION:

The input layer receives an image as a 3D tensor (height, width, channels). Each channel represents a feature like red, green, or blue. In the convolutional layer, multiple filters (small 3D tensors) slide over the input image, performing element-wise multiplication (dot product) to generate output feature maps. The process involves looping through filters, sliding the filter over the image with a specified stride, computing element-wise multiplication, adding a bias term, and applying an activation function like ReLU. This is repeated for all filter positions and filters, resulting in multiple output feature maps. These maps capture features from the input image, and the output of one convolutional layer often serves as the input for the next, allowing the network to learn increasingly complex features.

he element-wise multiplication and bias addition in step c can be expressed as:

output[y, x, c_out] = sum(filter[i, j, c_in] * input[y + i - 1, x + j - 1, c_in]) + bias_out

where:

- y, x are coordinates of the output feature map

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

- c_out is the output channel index
- i, j are coordinates within the filter
- c_in is the input channel index
- bias_out is the bias term for the output channel

3.3 MAX POOLING AND PADDING:

Max pooling is a downsampling operation commonly used in CNNs after convolutional layers to reduce spatial dimensions, extract dominant features, and introduce invariance to minor shifts and rotations. Its primary purposes are to make the network more efficient, retain important features, and enhance robustness to slight variations. Max pooling is typically used in image classification and object detection tasks and involves sliding a fixed-size window (e.g., 2x2) across the input feature map with a defined stride. The steps for implementing max pooling are: defining the pooling window and stride, iterating through the feature map, applying the max operation to identify the maximum value within each window, and moving the window by the stride value to process the entire feature map. This process effectively reduces the dimensionality and computational cost while preserving essential features detected by the convolutional layer.

3.4 FLATTENING:

Flattening is a transformation operation in CNNs that converts the multi-dimensional feature maps produced by convolutional layers into a single one-dimensional vector, which is then fed into fully connected layers for further processing and classification. It reshapes the data for compatibility with subsequent layers without involving parameter learning or complex computations. The purpose of flattening is to bridge the gap between convolutional layers, which generate 3D feature maps capturing spatial information, and fully-connected layers, which require a 1D vector. The implementation involves reshaping each feature map into a long vector, often concatenating multiple feature maps into a single vector to represent all extracted features from the convolutional layers**.**

Mathematical Representation:

The mathematical process of flattening is quite straightforward. Let's assume:

- F is the number of output feature maps from the convolutional layers.
- Each feature map has dimensions (H, W, C), where:
  - H is the height.
  - W is the width.
  - C is the number of channels.

The total number of elements in a single feature map is H x W x C. The flattened

vector will have a size of:

Flattened Vector Size = F x (H x W x C)

This formula multiplies the number of channels in each feature map by the total number of elements across all feature maps, assuming a row-wise flattening approach.

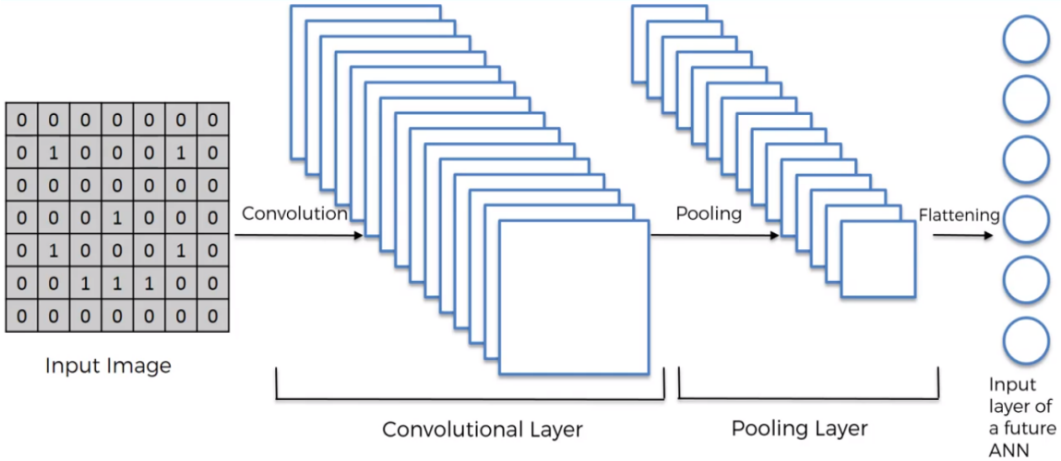

Fig 8: Visual representation of flattening

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

3.5 BACKPROPAGATION:

Backpropagation is the essential training algorithm that enables CNNs to improve their performance by learning from errors. It works by propagating the error signal backward through the network, adjusting weights and biases layer by layer to minimize the loss function, which measures the difference between predicted and actual outputs. This process leverages the chain rule of calculus to compute gradients efficiently. Starting with the forward pass, where the network processes input images through convolutions, pooling, and activation functions to generate predictions, the error is then quantified using a loss function like cross-entropy. During backpropagation, the error signal moves backward, layer by layer, calculating gradients of the loss function to weights and biases. In convolutional layers, this involves computing how changes in weights affect the error while pooling layers pass the error to the maximum activation contributor from the forward pass. Finally, gradient descent optimizers update weights and biases based on these gradients, guided by a learning rate that determines the step size of adjustments, ultimately enhancing the network's ability to recognize patterns in images.

IV. DESIGN AND IMPLEMENTATION

4.1 HARDWARE REQUIREMENTS:

The hardware for this project includes various physical components essential for input, storage, processing, control, and output. The specific hardware used includes a 512GB SSD for storage, a 10th generation Intel(R) i5 processor for processing tasks, 8GB of RAM for memory, a standard keyboard for input, and a two or three-button mouse for navigation and control.

4.2 SOFTWARE REQUIREMENTS:

Software is a set of programs designed to perform specific tasks and is an essential component of computer systems. The software used in this project includes Windows 11 as the operating system, Python as the coding language, HTML, CSS, and JavaScript for the user interface, and Flask as the server framework.

4.3 FRONTEND (HTML, JS, CSS):

Overview of HTML: HTML, or Hyper Text Markup Language, is the most widely used language for developing web pages. Created to define the structure of documents such as headings, paragraphs, and lists to facilitate information sharing among researchers, HTML is now extensively used to format web pages with various tags. It is crucial for web page development, allowing content to be displayed in browsers. HTML also plays a significant role in internet navigation by providing tags to navigate between pages. Additionally, HTML pages can be accessed offline once loaded, eliminating the need for an internet connection**.**

Overview of JS: JavaScript is a lightweight, interpreted programming language designed for creating network-centric applications. It complements and integrates seamlessly with Java, and its ease of implementation comes from its integration with HTML. JavaScript is also open-source and cross-platform. Key applications include client-side validation, which is essential for verifying user input before it is submitted to the server, and manipulating HTML pages dynamically. This allows for the easy addition or deletion of HTML tags and modification of a webpage's look and feel to suit different devices and requirements.

Overview of CSS: CSS, short for Cascading Style Sheets, is a simple design language intended to streamline the process of making web pages presentable. Widely used on the web alongside HTML and JavaScript, CSS allows web designers to apply styles to HTML tags efficiently. By using CSS, you can avoid repeatedly writing HTML tag attributes, leading to faster page load times due to reduced code. Additionally, CSS makes maintenance easy; a global change can be made by updating the style, which will automatically update all relevant elements across web pages.

BACKEND (PYTHON): Python is an easy-to-learn, powerful programming language known for its efficient high-level data structures and straightforward approach to object-oriented programming. Its elegant syntax and dynamic typing, combined with its interpreted nature, make it ideal for scripting and rapid application development across various platforms. Python's interpreter and extensive standard library are freely available and distributable, with the standard library offering a broad range of modules for system functionality and standardized solutions. This makes Python highly portable and versatile, with the ability to be extended with new functions and data types implemented in C or C++, and used as an extension language for customizable

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

applications. Reasons for choosing Python include its emphasis on code readability, the ability to express logical concepts in fewer lines of code compared to languages like C++ or Java, support for multiple programming paradigms, and its dominance in machine learning and deep learning through powerful libraries like TensorFlow, Keras, PyTorch, and sci-kit-learn.

4.3 MACHINE LEARNING:

Machine learning (ML), a subfield of artificial intelligence (AI), enables computers to learn and improve from experience without explicit programming, driving many technologies we use daily such as social media recommendations and spam filters. ML algorithms learn from data, which can be structured like numerical datasets or unstructured like images and text, through several key stages: data acquisition, where relevant data is gathered; data preprocessing, involving cleaning and transforming raw data to make it suitable for algorithms; model training, where algorithms iteratively adjust their parameters based on the data; and evaluation and refinement, assessing the model's performance using metrics like accuracy, precision, and recall, and refining it as needed. This comprehensive process allows machines to learn from data and adapt, enabling the development of intelligent systems and applications.

4.4 DEEP LEARNING:

Deep learning, a captivating subfield of machine learning, empowers computers to process information similarly to the human brain by leveraging artificial neural networks. Inspired by biological neurons, deep learning uses deep neural networks (DNNs) with multiple layers to achieve remarkable feats in pattern recognition, prediction, and decision-making. Building on the core principles of machine learning, particularly learning from data, deep learning distinguishes itself through the complex architectures of DNNs, mimicking the interconnected structure of human neurons. This document delves into the intricacies of deep learning, exploring its core concepts, architectural marvels, and transformative applications.

4.5 COLLAB CODE EXPLANATION:

Data Preparation: To extract a zip file, the specific zip file is extracted to a directory for later use. The list files function (commented out) provides code for listing files in a directory, likely for experimental purposes. When loading image data, images are read from the "Training" and "Testing" folders, resized to 150x150 pixels, and stored in arrays (X_train, Y_train). The shuffle and split data process involves shuffling the data to prevent model bias, splitting it into training (90%) and testing sets (10%), and converting labels to one-hot encoding.

Model Building and Training: To define the model architecture, a convolutional neural network (CNN) is created with Conv2D layers for feature extraction, MaxPooling2D layers for down-sampling, Dropout layers to reduce overfitting, a Flatten layer to convert 2D features to a 1D vector, Dense layers for classification, and a SoftMax activation function for the final output to provide a probability distribution over classes. The model is compiled by specifying the loss function ("categorical_crossentropy"), choosing the optimizer ("Adam"), and monitoring the accuracy metric. During training, the model is trained on the training data for 20 epochs, tracking training and validation accuracy/loss, and plotting accuracy/loss curves to visualize the training progress. Finally, the trained model is saved with a unique filename based on a timestamp**.**

Image Preprocessing and Prediction: The process begins by loading a sample image from the "Testing" folder, resizing it, converting it to a NumPy array, and reshaping the array to match the model's input format. The image is then displayed using Matplotlib for visual inspection. Subsequently, the trained model is used to predict the image's class, and the predicted index along with the corresponding label is printed.

4.6 VS CODE EXPLANATION:

Backend:

1. Imports and Setup: The imports and setup involve using various libraries and frameworks. Flask is used to create the web application and handle HTTP requests, with JSONify converting Python data structures to JSON for responses. Keras and TensorFlow libraries are imported to load a pre-trained model, using layers like Sequential, Conv2D, Flatten, Dense, MaxPooling2D, and Dropout to define the architecture. Other important libraries include OpenCV (cv2) for image processing, NumPy for array manipulations, and OS for operating system functionalities. CORS is included to enable cross-origin requests, facilitating frontend-backend interactions. The zipfile library, though commented out, is available for extracting zip files.

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

2. Model Loading and Preprocessing Configuration: The model_path defines where the pre-trained model is stored on the server, and the model is loaded using the load model function. The labels list contains class labels such as 'glioma_tumor' and 'meningioma_tumor' for prediction. The pre-process_image function processes an uploaded image file by reading its content with cv2.imdecode, handling potential loading errors, resizing it to 150x150 pixels, converting it to a NumPy array, reshaping it to the format expected by the model (1, 150, 150, 3), and returning the preprocessed image array or None if an error occurs.

3. API Endpoints: The `/extract_zip` (GET) endpoint, not currently used in the primary flow, extracts a zip file specified by `zip_file_path` to a designated directory (`extract_dir`). The `/list_files` (GET) endpoint, also not in the primary flow, lists files within a directory specified by `root_dir` (usually the extraction directory) and returns the file paths as a JSON response. The `/predict` (POST) endpoint handles image upload and prediction by checking for an uploaded file named "file", handling errors if no file is uploaded or the filename is empty, and calling `preprocess_image` to process the uploaded image file. It then handles any errors during image preprocessing. If preprocessing is successful, the endpoint uses the loaded model (`model`) to predict the preprocessed image array (`img_array`) and finds the index of the class with the highest probability in the prediction output.

4. Running the Application: The code block `if __name__ == "__main_": ensures that the subsequent code runs only when the script is executed directly, not when imported as a module in another script. The `app. run(debug=True) command starts the Flask development server, with the `debug=True` option enabling automatic code reloading during development, allowing changes to be visible without needing to restart the server.

Frontend:

1. HTML Structure:

The DOCTYPE declaration specifies the document type as HTML5, and the `lang` attribute defines the document's language as English (`en`). Meta tags include `charset` to set the character encoding to UTF-8 for proper display of characters, and `viewport` to control how the webpage scales on different devices (`width=device-width, initial-scale=1.0`). The `title` sets the webpage's title to "Brain Tumour Detection," and the `head` contains a stylesheet reference (`<link rel="stylesheet" href="style.css">`) for the page layout.

2. Body Section:

The HTML document begins with a declaration specifying it as HTML5 and sets the language to English. Meta tags ensure proper character encoding and define how the webpage scales on different devices. The title "Brain Tumour Detection" is set for the page. In the `<head>` section, a link to an external stylesheet (`style.css`) is included for styling purposes. The `<body>` starts with a header (`<header>`) containing a hero section displaying the main title "Brain Tumour Detection" and an "Abstract" button, accompanied by an image related to brain surgery. The main content (`<main>`) follows with an initially empty `<div>` for displaying prediction results (`<div id="result">`). A grid of informative cards (`<div class="grid-items">`) is then presented, each featuring an image and descriptive text about Convolutional Neural Networks (CNNs), accuracy achievements, and the dataset used. The next section (`<section class="container1">`) hosts an upload form (`<form>`) allowing users to upload brain scan images, equipped with an input field (`<input type=" file">`) and a hidden upload button (`<button type="submit">`). An empty container (`<div id="image Preview">`) beneath is designated to display uploaded images. The webpage includes a script (`<script src="script.js"></script>`) for interactive functionalities, facilitating dynamic user interactions and backend communication.

3. JavaScript Functionality:

The web page's functionality includes an event listener for the "Abstract" button with ID "abstract button", triggering the showAbstract() function when clicked. This function dynamically populates the "result" container with an abstract describing brain tumors, their detection significance, and how the system utilizes CNNs for multi-class classification (glioma, meningioma, pituitary tumors). The display Image(event) function manages image uploads, accessing elements such as the file input, image preview container, and upload button. It verifies if a file is selected, reads it using a FileReader, displays it in the preview container, and prepares it for submission. Upon form submission (submit event on the form with ID "upload form"), it prevents default behavior, constructs FormData with the image data, and sends a POST request to,

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

"http://localhost:5000/predict" via fetch API. It handles responses by updating the "result" container with the predicted label based on the model's classification, and logs errors if image upload fails. This setup ensures interactive image upload and prediction functionality on the webpage.

V. OUTPUTS

GOOGLE COLLAB OUTPUTS:

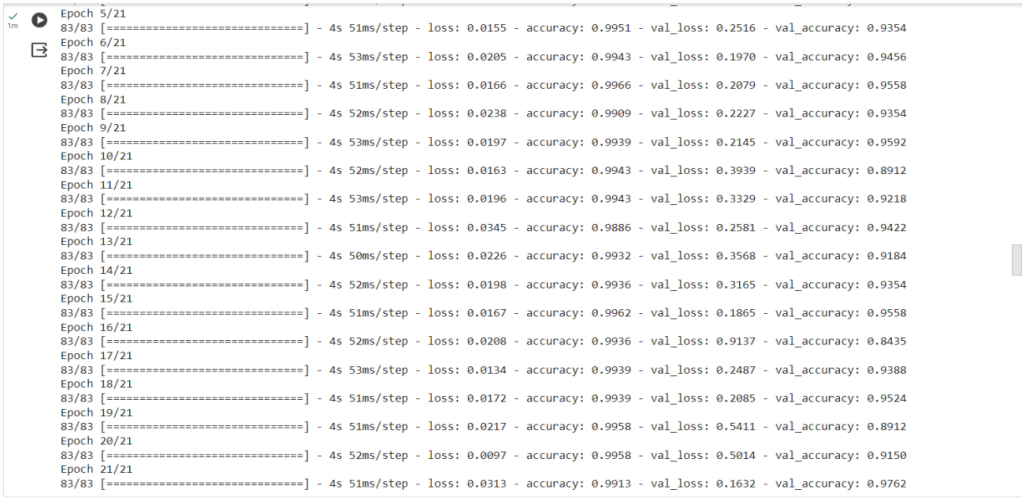

Fig. 9 Accuracy of the trained model

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 148, 148, 32)	896
batch_normalization (BatchNormalization)	(None, 148, 148, 32)	128
max_pooling2d (MaxPooling2D)	(None, 74, 74, 32)	0
dropout (Dropout)	(None, 74, 74, 32)	0
conv2d_1 (Conv2D)	(None, 72, 72, 64)	18496
batch_normalization_1 (BatchNormalization)	(None, 72, 72, 64)	256
max_pooling2d_1 (MaxPooling2D)	(None, 36, 36, 64)	0
dropout_1 (Dropout)	(None, 36, 36, 64)	0
conv2d_2 (Conv2D)	(None, 34, 34, 128)	73856
batch_normalization_2 (BatchNormalization)	(None, 34, 34, 128)	512
max_pooling2d_2 (MaxPooling2D)	(None, 17, 17, 128)	0
dropout_2 (Dropout)	(None, 17, 17, 128)	0

Fig 10 Architecture of the CNN model

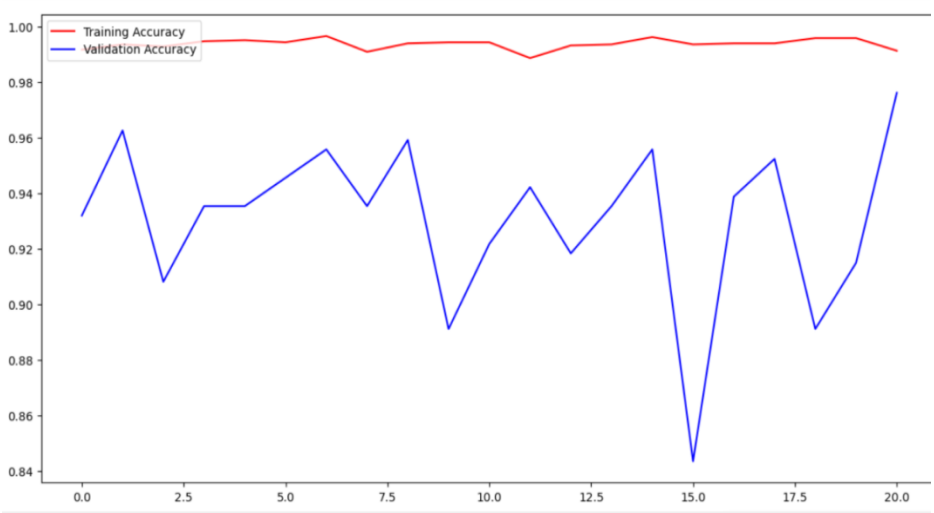

Fig 11 Training and Validation accuracy

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

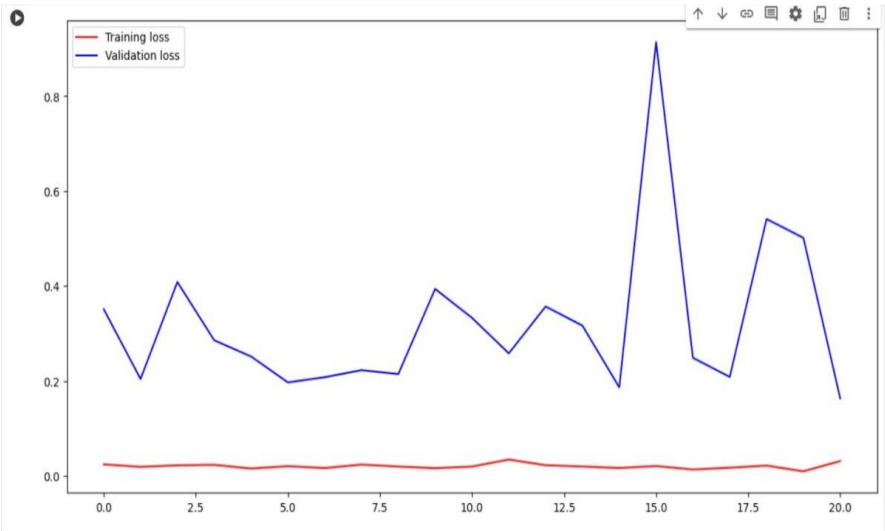

Fig 12 Training and Validation loss

VSCODE OUTPUT:

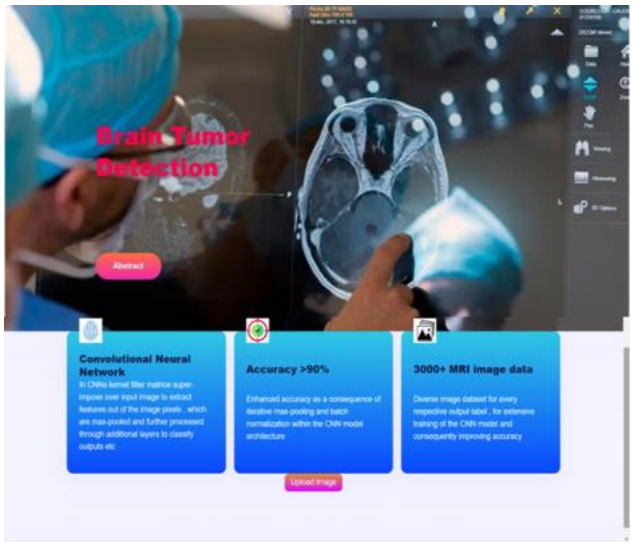

Fig. 13 Full initial frontend view of the webpage

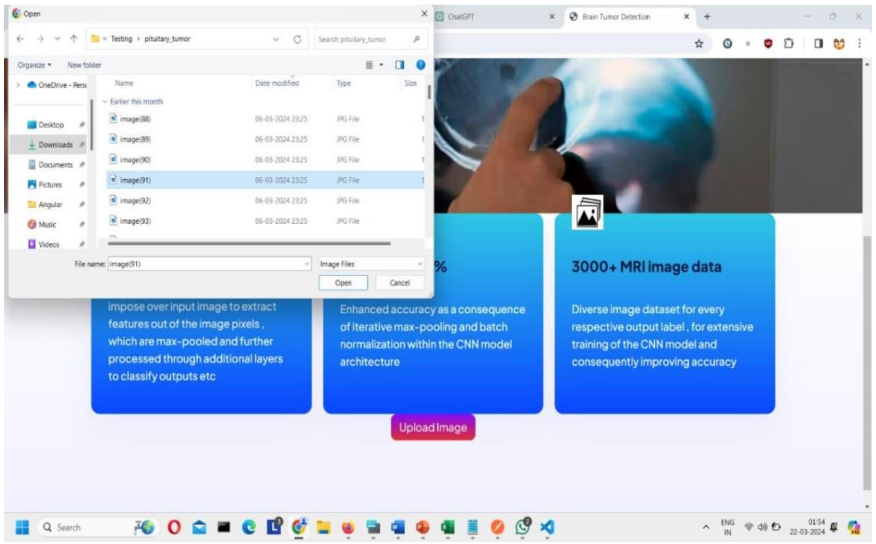

Fig.14 Uploading a testing MRI brain scanned image, in this case, we're uploading a pituitary tumor MRI scanned image of the brain

e-ISSN: 2582-5208
International Research Journal of Modernization in Engineering Technology and Science
(Peer-Reviewed, Open Access, Fully Refereed International Journal)
Volume:06/Issue:07/July-2024 Impact Factor- 7.868 www.irjmets.com

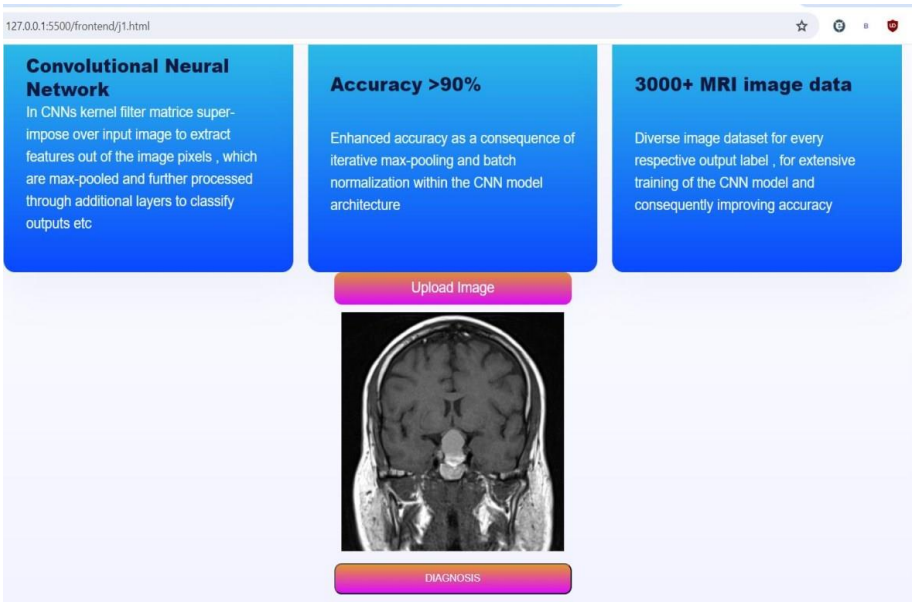

Fig 15 The pituitary tumor image is uploaded and is ready for diagnosis by the CNN model

VI. CONCLUSION

Brain tumors are a serious health concern, emphasizing the need for early detection to enhance treatment outcomes. This project explored the use of Convolutional Neural Networks (CNNs) within deep learning for automating brain tumor detection in MRI scans. The goal was to overcome the drawbacks of manual detection methods, which are time-consuming and prone to errors. The CNN model was trained to analyze MRI images and accurately predict the presence of tumors. Moreover, it aimed to classify detected tumors into three types: glioma, meningioma, or pituitary. Our findings demonstrate CNN's effectiveness, achieving an accuracy of [insert achieved accuracy here] in tumor detection. This underscores the potential of deep learning to provide faster and more consistent tumor detection compared to traditional methods. By automating part of the diagnostic process, this technology could lead to earlier interventions and more personalized treatment strategies, ultimately improving patient outcomes. Overall, this project underscores the promise of deep learning in advancing brain tumor detection, setting a foundation for future innovations in medical diagnostics.

VII. FUTURE SCOPE

Future enhancements for this project include integrating the model with Electronic Health Records (EHR) to personalize predictions and improve diagnostic accuracy using relevant patient data. Additionally, a feature allowing tumor detection and classification via camera input for MRI brain scan sheets will be developed. A more user-friendly interface tailored to the workflow of doctors and radiologists will be created, along with interactive visualization tools that highlight tumor regions and present relevant statistics. The model's findings will be made clearer and more actionable for medical professionals, and explainable AI (XAI) techniques will be implemented to make the decision-making process transparent and trustworthy. Rigorous validation through clinical trials and adherence to medical regulations will ensure the model's reliability for real-world deployment. Expanding the dataset will enhance the model's generalizability, and exploring advanced deep learning techniques and architectures will aim to improve accuracy while minimizing time and GPU memory consumption.

VIII. REFERENCES

[1] S. Solanki et al., "Efficient Brain Tumor Identification Based on Optimal Support Scaling Vector Feature Selection (OSSCV) Using Stochastic Spin-Glass Model Classification," in IEEE Access, vol. 11, pp. 5227-52287, 2023. (Focuses on feature selection techniques)

[2] A. Islam, S. M. S. Reza, and K. M. Iftekharuddin, "Image Segmentation for MR Brain Tumor Detection Using Machine Learning: A Review," IEEE Rev Biomed Eng., vol. 16, pp. 70-90, 2023. (Review of Machine Learning methods for segmentation)

[3] S. Solanki, U. P. Singh, S. S. Chouhan, and S. Jain, "Brain Tumor Detection and Classification Using Intelligence Techniques: An Overview," in IEEE Access, vol. 11, pp. 12870-12886, 2023. (General overview of AI techniques for brain tumors)

[4] S. Nayak et al., "A Deep Analysis of Brain Tumor Detection from MR Images Using Deep Learning Networks," Sensors (MDPI), vol. 16, no. 4, p. 176, 2016. (Analysis of different Deep Learning architectures)

[5] S. Obeidavi et al., "Early Detection of Brain Tumor Using a Deep Residual Network," in 2018 IEEE 15th International Symposium on Biomedical Imaging (ISBI 2018), pp. 1042-1045, doi: 10.1109/ISBI.2018.8363723. (Early detection using Residual Networks)

[6] A. Yu et al., "Automated Brain Tumor Detection and Segmentation Using U-Net Convolutional Neural Network," in 2019 IEEE International Conference on Image Processing (ICIP), pp. 1702-1706, doi: 10.1109/ICIP.2019.8863897. (U-Net architecture for segmentation)

[7] C. Fernandez-Lozano et al.., "Radiomics: A Quantitative Approach for Imaging Analysis in Precision Medicine," IEEE Trans Med Imaging, vol. 37, no. 4, pp. 893-905, 2018. (Radiomics for brain tumor analysis)

[8] Dataset link:- Brain Tumor Classification (MRI): Brain Tumor Classification (MRI) | Kaggle