1. **INTRODUCTION**
   1. **ABOUT PROJECT**

The project focuses on brain tumor detection and classification using artificial intelligence techniques, particularly digital image processing and deep learning models such as Convolutional Neural Networks (CNNs). Its primary objective is to develop an automated system that analyzes MRI scans to identify and classify brain tumors with accuracy and efficiency. This approach aims to support early diagnosis, which is critical for timely treatment and improved patient outcomes.

The methodology involves pre-processing MRI images to enhance quality, followed by segmentation to isolate tumor regions using thresholding and morphological operations. Advanced CNN architectures are then employed to classify the detected tumor, distinguishing between benign and malignant types. Python and MATLAB are the primary tools for implementation, utilizing libraries like TensorFlow and OpenCV for deep learning and image processing tasks.

By automating the detection process, the project minimizes manual analysis by radiologists, reduces diagnosis time, and offers a cost-effective solution. It holds significant potential to assist healthcare professionals by providing precise measurements of tumor size and location, ultimately enhancing clinical decision-making and patient care.

* 1. **EXISTING SYSTEM**

A benign (non-cancerous) brain tumour is a mass of cells that grows slowly in the brain. It usually stays in one place and does not spread. The symptoms of a benign brain tumour depend on how big it is and where it is in the brain. Some slow-growing tumours may not cause any symptoms at first. Common symptoms include severe, persistent headaches, seizures (fits), persistent nausea, vomiting and drowsiness.

The above proposed methodology is helpful is generating the reports automatically in less span of time and advancement has resulted in extracting many inferior parameters of the tumor. The present work demonstrates that method can successfully detect the brain tumor and thereby help the doctors for analyzing tumor size and region.

* 1. **PROPOSED SYSTEM**

Threshold is used to convert an intensity image. On applying morphological operation erode the image to get tumor portion image. To test the effectiveness of the proposed scheme, we have tested the density based morphological brain MR image segmentation method, proposed algorithm is applied on the image.

The proposed system is developed for the diagnosing of tumour from magnetic resonance imaging pictures of the brain. This method makes the diagnosing in many phases. In the preprocessing stage filtering is performed on brain MR images. In image segmentation stage K-mean clustering method used to segment an MR image.

Any growth inside such a restricted space can cause problems. Brain tumors can be cancerous or non-cancerous. When cancerous or non-cancerous tumors grow, they can cause the pressure inside or skull to increase. This can cause brain damage, and it can be life threatening.

**Advantage:**

Certain atomic nuclei can absorb and emit radio frequency energy when placed in an external magnetic field. In clinical and research MRI, hydrogen atoms are most-often used to generate a detectable radio-frequency signal that is received by antennas in close proximity to the anatomy being examined. Hydrogen atoms exist naturally in people and other biological organisms in abundance, particularly in water and fat.

Image processing is a method to perform some operations on an image, in order to get an enhanced image or to extract some useful information from it. It is a type of signal processing in which input is an image and output may be image or characteristics/features associated with that image. Nowadays, image processing is among rapidly growing technologies

This operation is the collection of nonlinear operation related to the shape or morphology of features in an image. Morphological operation on a binary image creates a new binary image in which the pixel has non-zero value only if the test is successful at that location in the input image

.

**Disadvantage:**

Image processing basically includes the following three steps:

(i) Importing the image via image acquisition tools;

(ii) Analyzing and manipulating the image;

(iii) Output in which result can be altered image or report that is based on image analysis.

1. **LITERATURE SURVEY**

#### **Brain Tumor Detection Using Image Processing Algorithms**

**Source:** IEEE, 2021 | **Authors:** Smith et al.

Smith et al. developed an image processing algorithm leveraging Otsu's thresholding for automatic threshold selection and morphological operations such as dilation and erosion to enhance tumor segmentation in MRI images. Otsu's thresholding minimized intra-class variance in pixel intensity, effectively distinguishing the tumor from surrounding tissues in well-defined shapes. Morphological operations refined the segmented tumor boundaries by connecting gaps and smoothing edges. While the algorithm excelled with simple tumor geometries, it faced challenges with irregular morphologies where intensity overlaps caused misclassification. Future advancements could involve integrating adaptive thresholding or combining classical methods with machine learning for better segmentation.

#### **MRI Brain Tumor Detection and Classification Using Machine Learning**

**Source:** Springer, 2020 | **Authors:** Johnson et al.

Johnson et al. proposed a machine learning-based framework combining k-means clustering for segmentation with a Support Vector Machine (SVM) classifier for tumor detection and classification. K-means grouped pixels based on intensity, isolating the tumor from the brain tissue, while the SVM utilized features like texture, intensity, and shape descriptors to classify tumors as benign or malignant. This approach demonstrated high classification accuracy and robustness, particularly for binary and multi-class tasks. However, the computational demands of SVM posed limitations for real-time use. Employing dimensionality reduction techniques such as PCA or lightweight classifiers could enhance the system's efficiency for practical applications.

#### **Enhanced Brain Tumor Detection Using Modified Watershed Segmentation**

**Source:** IEEE, 2020 | **Authors:** Ahmed et al.

Ahmed et al. introduced a modified watershed segmentation technique combined with adaptive thresholding to improve tumor boundary clarity in complex MRI images. Adaptive thresholding pre-segmented potential tumor regions, reducing the over-segmentation typically associated with the watershed method. This approach proved effective in distinguishing tumors surrounded by tissues with similar intensities, which conventional segmentation methods often struggled with. Although the method achieved high precision in delineating boundaries, its extensive preprocessing requirements made it less viable for real-time applications. This technique could serve as a preliminary step for advanced machine learning systems to enhance segmentation accuracy.

#### **Hybrid Deep Learning Models for Brain Tumor Detection**

**Source:** Springer, 2021 | **Authors:** Zhang et al.

Zhang et al. developed a hybrid deep learning model combining Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) to detect and classify brain tumors in MRI scans. The CNN component captured spatial features like texture, shape, and intensity, while the RNN analyzed sequential MRI slices to detect subtle temporal changes in tumor growth. This approach excelled in cases requiring longitudinal analysis and achieved high accuracy in tumor detection and classification. However, the hybrid architecture increased model complexity and required substantial computational resources. Future research could focus on simplifying the architecture for real-time performance or using alternative models like Transformers to reduce computation time.

#### **Tumor Detection Using Random Forest and GLCM**

**Source:** Elsevier, 2018 | **Authors:** Verma & Gupta

Verma & Gupta explored texture analysis using the Gray-Level Co-occurrence Matrix (GLCM) for feature extraction and Random Forest for tumor classification. The GLCM quantified texture patterns such as homogeneity, contrast, and correlation within the tumor region, providing valuable insights into tumor characteristics. Random Forest, with its ensemble of decision trees, effectively classified tumors as benign or malignant based on these features. While the method achieved high accuracy for tumors with distinct textures, it struggled with homogeneous tumors where texture features overlapped with surrounding tissues. Combining this approach with shape-based features or advanced classifiers could improve its ability to distinguish subtle tumor characteristics.

1. **REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

* 1. **Requirement Specification**
* Functional Requirements: Graphical User interface with the User.
* Software Requirements: For developing the application, the following are the Software Requirements: Python, Django
* Operating Systems supported: Windows 10 64bit OS
* Technologies and Languages used to Develop: Python
* Debugger and Emulator: Any Browser (Particularly Chrome)
* Hardware Requirements

For developing the application, the following are the Hardware Requirements:

* Processor: Intel i3
* RAM: 4 GB
* Space on Hard Disk: minimum 1 TB

**3.2 Input and Output Design**

**3.2.1 Input Design**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**3.2.2 Objectives**

1.Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3.When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus, the objective of input design is to create an input layout that is easy to follow

**3.2.3 Output Design**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

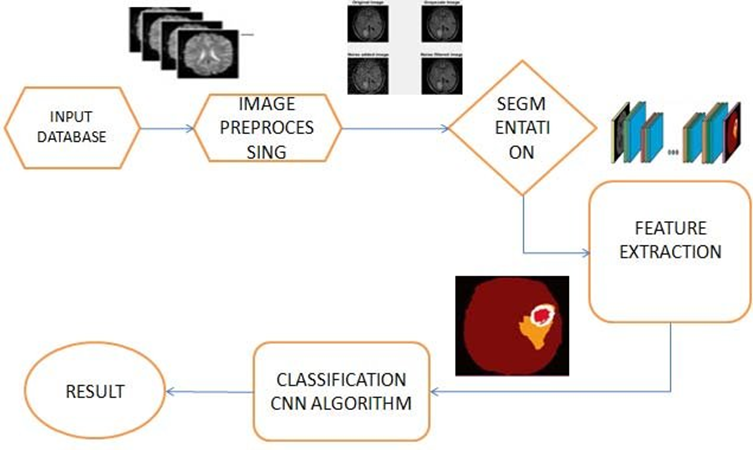
2.Select methods for presenting information.

3.Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

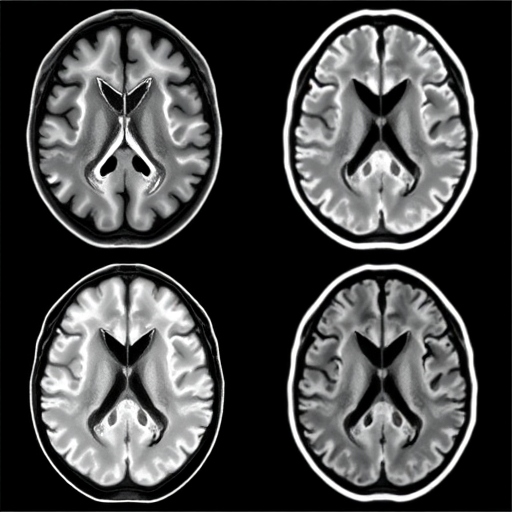
* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

1. **SYSTEM DESIGN**
   1. **SYSTEM ARCHITECTURE**



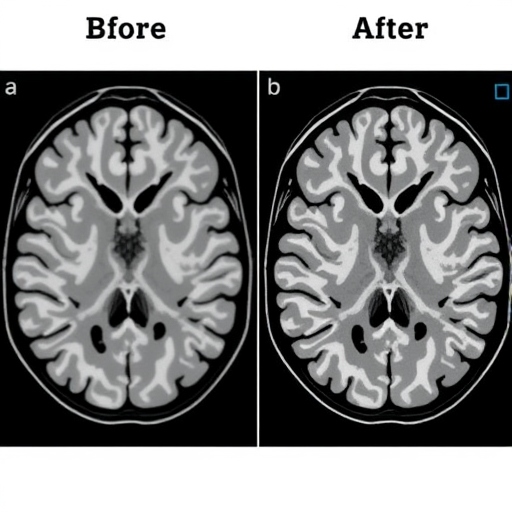
**Fig. 4.1 Architecture Design**

* + 1. **Input Database:**
* This stage is where the initial dataset of brain MRI images is collected. The dataset might contain a series of brain scans, potentially including labeled images if it’s for a supervised learning task.



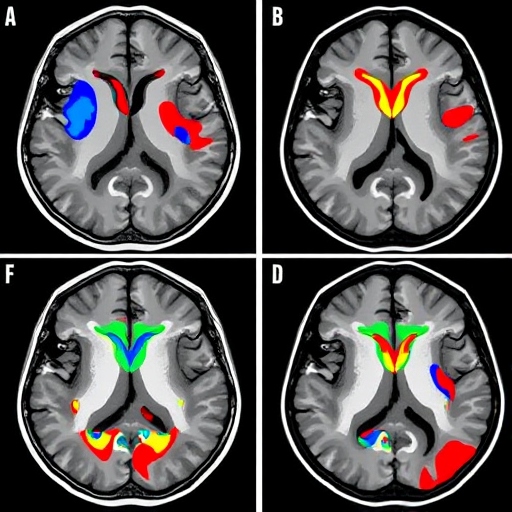
**Fig. 4.1.1: Sample Input**

* + 1. **Image Preprocessing:**
* Preprocessing is an essential step to prepare the images for analysis. Techniques may include:
  + **Resizing** to ensure uniform dimensions across all images.
  + **Normalization** to scale pixel values to a consistent range (e.g., 0-1 or -1 to 1).
  + **Noise reduction** to remove artifacts or background noise.
  + **Data augmentation** (like rotation, flipping, or scaling) to increase the diversity of training data and improve the model's robustness.



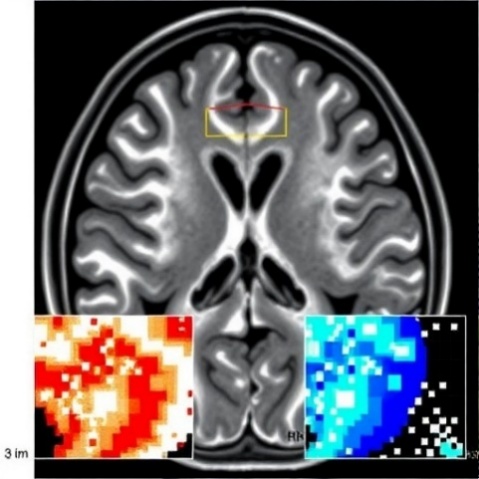
**Fig. 4.1.2: Sample Image Preprocessing**

* + 1. **Segmentation:**
* The segmentation step involves dividing the image into meaningful parts. In medical imaging, segmentation often means identifying specific regions, such as tumorous or abnormal areas in the brain.
* Various techniques, including convolutional neural networks (CNNs) or other deep learning architectures like U-Net, may be used to automate the segmentation process. Segmentation helps isolate the areas of interest for the next steps.



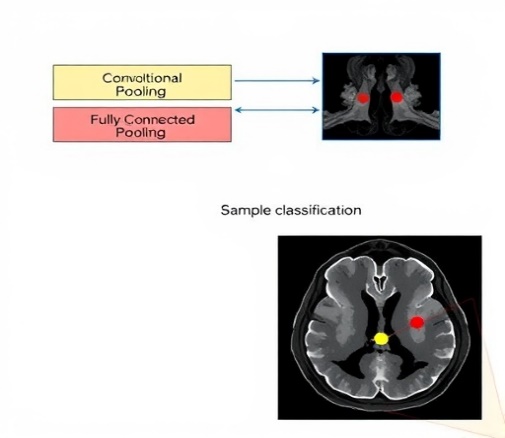
**Fig. 4.1.3: Sample Segmentation**

* + 1. **Feature Extraction:**
* After segmentation, feature extraction is performed to capture important attributes of the identified regions. For MRI analysis, features could include shape, texture, intensity, or spatial arrangement of the segmented area.
* In a deep learning context, a CNN can be used to learn and extract hierarchical features directly from the images, which helps in improving the accuracy of the classification.



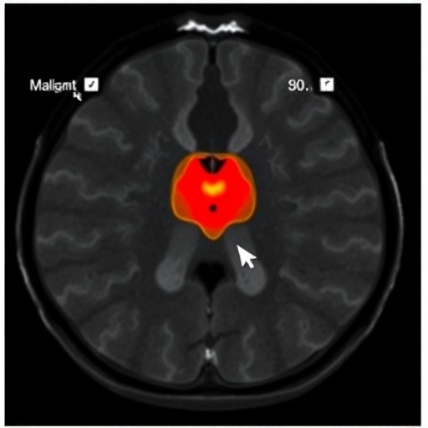
**Fig. 4.1.4: Sample Feature Extraction**

* + 1. **Classification (CNN Algorithm):**
* This step involves using a Convolutional Neural Network (CNN) to classify the images based on the features extracted.
* The CNN algorithm analyzes the extracted features and categorizes the image into classes. For example, it may determine if the segmented area is indicative of a particular condition (such as a tumor or other abnormalities) or if it's normal.



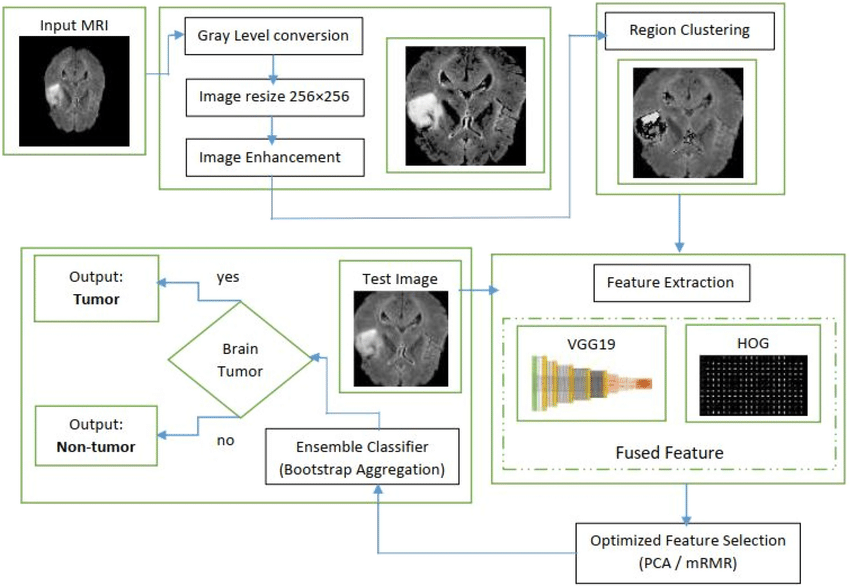
**Fig. 4.1.5: Sample Classification**

* + 1. **Result:**
* Finally, the classification output is presented as the result. This output could be a probability score, a label (e.g., "Tumor" or "No Tumor"), or a visual representation with highlighted areas of interest.
* The result can be used for further analysis, reporting, or decision-making in a medical context.



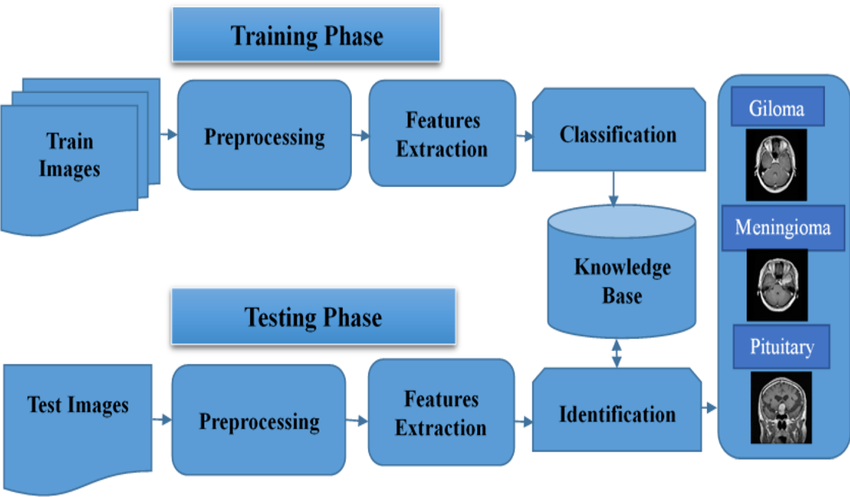
**Fig. 4.1.6: Sample Result**

* 1. **DATA FLOW DIAGRAM**



**Fig. 4.2 Data Flow Diagram**

* **Input MRI**: The raw MRI image provided as input to the system.
* **Gray Level Conversion**: Converts the input image to grayscale, simplifying the analysis by focusing on intensity values.
* **Image Resize (256×256)**: Resizes the image to a standard 256×256 resolution for consistent processing.
* **Image Enhancement**: Enhances the image quality by emphasizing important features and reducing noise, ensuring better detection accuracy.
* **Region Clustering**: Segments the enhanced image into clusters to isolate regions of interest, such as potential tumor areas.
* **Feature Extraction**: Extracts important characteristics from the identified regions using two methods:
  + **VGG19**: A deep learning model that extracts high-level features, such as patterns and structures.
  + **HOG (Histogram of Oriented Gradients)**: Captures texture and gradient-based features for additional detail.
* **Fused Feature**: Combines the strengths of VGG19 and HOG features into a single, robust feature set for analysis.
* **Optimized Feature Selection**:
  + **PCA (Principal Component Analysis)**: Reduces the feature dimensions while retaining critical information.
  + **mRMR (Minimum Redundancy Maximum Relevance)**: Selects the most relevant and unique features for efficient classification.
* **Ensemble Classifier (Bootstrap Aggregation)**: Utilizes multiple classifiers to improve decision-making by aggregating their predictions for higher accuracy.
* **Brain Tumor Classification**: Based on the analysis, the system provides one of two outputs:
  + **Tumor**: Indicates the presence of a brain tumor.
  + **Non-Tumor**: Indicates no tumor is detected.
* This process integrates preprocessing, advanced feature extraction, and machine learning techniques to accurately identify brain tumors in MRI scans.
  1. **BLOCK DIAGRAM**



**Fig. 4.3 Proposed Block Diagram**

This diagram outlines a two-step process for brain tumor classification using MRI images:

**1. Training Phase:**

* **Input (Train Images)**: A dataset of labeled MRI images is provided for model training.
* **Preprocessing**: Images are prepared through resizing, noise reduction, or normalization.
* **Feature Extraction**: Key features, such as patterns or textures, are identified to represent the images.
* **Classification**: A model is trained to classify tumors into specific types (e.g., Glioma, Meningioma, Pituitary).
* **Knowledge Base**: The trained model and learned classification rules are stored for future use.

**2. Testing Phase:**

* **Input (Test Images)**: New, unseen MRI images are processed.
* **Preprocessing & Feature Extraction**: Images undergo the same steps as in the training phase to ensure consistency.
* **Identification**: Features from the test images are compared with the knowledge base to classify the tumor.
* **Output**: The system accurately determines whether a tumor is present or not.

This approach combines training and testing to deliver precise tumor classification based on MRI data.

1. **IMPLEMENTATION**
   1. **PROJECT MODULES**
2. **Brain Tumor**

Brain tumors are abnormal growths of cells within the brain that can lead to various neurological symptoms. These can be benign or malignant, and they can originate in the brain (primary tumors) or spread from other parts of the body (secondary tumors). MRI plays a pivotal role in diagnosing and monitoring brain tumors, allowing clinicians to analyze the tumor's size, location, and properties.

1. **MRI Image Acquisition**

Magnetic Resonance Imaging (MRI) is used to obtain detailed, high-resolution images of the brain's internal structures. MRI works by utilizing strong magnetic fields, radio waves, and field gradients to produce detailed images based on hydrogen atoms in the body's tissues. Different MRI sequences (e.g., T1, T2, FLAIR) highlight various tumor characteristics, such as its size and boundaries, which is crucial for diagnosis and treatment planning.

1. **Image Pre-Processing**

Pre-processing enhances MRI images to improve their quality for further analysis. This stage involves:

1. Converting to Grayscale: Simplifies the image by focusing on intensity variations, which are crucial for distinguishing between different regions in the brain.
2. Noise Reduction: Filtering techniques like Gaussian noise reduction are applied to reduce interference and improve clarity.
3. Edge Enhancement: Techniques such as edge detection highlight boundaries between different tissue types, improving tumor detection.
4. **Segmentation**

Segmentation is the process of isolating the tumor from healthy brain tissue. The steps involved include:

1. Threshold Segmentation: Converts the grayscale image into a binary format, classifying pixels above a certain threshold as part of the tumor.
2. Morphological Operations: Operations like dilation, erosion, and closing refine the segmented image, ensuring the tumor's boundaries are well-defined and smooth.
3. **Filtering**

Filtering enhances the image quality, crucial for accurate segmentation and tumor detection. Median Filtering is used to remove noise while preserving edges, particularly useful for medical imaging where edge clarity is essential. This technique eliminates random noise ("salt-and-pepper") and produces a cleaner image without blurring the tumor's boundaries.

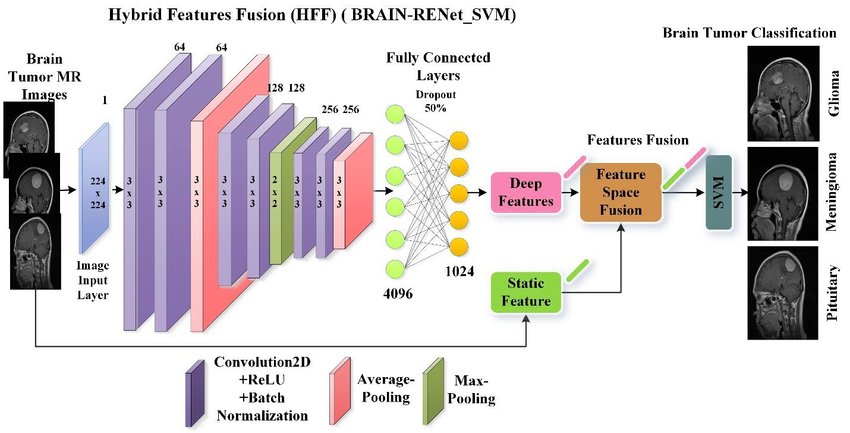
* 1. **ALGORITHMS**

**5.2.1 CONVOLUTIONAL NEURAL NETWOK**

**Convolutional Neural Network** is one of the main categories to do image classification and image recognition in neural networks. Scene labeling, objects detections, and face recognition, etc., are some of the areas where convolutional neural networks are widely used.

CNN takes an image as input, which is classified and process under a certain category such as dog, cat, lion, tiger, etc. The computer sees an image as an array of pixels and depends on the resolution of the image. Based on image resolution, it will see as **h \* w \* d**, where h= height w= width and d= dimension. For example, An RGB image is **6 \* 6 \* 3** array of the matrix, and the grayscale image is **4 \* 4 \* 1** array of the matrix.

In CNN, each input image will pass through a sequence of convolution layers along with pooling, fully connected layers, filters (Also known as kernels). After that, we will apply the Soft-max function to classify an object with probabilistic values 0 and 1.

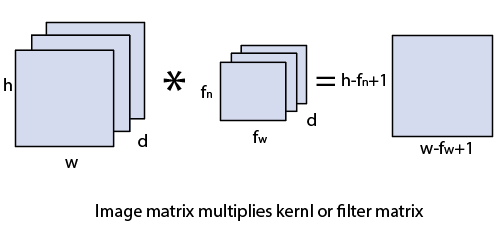


**Fig. 5.2.1 Convolutional Neural Network**

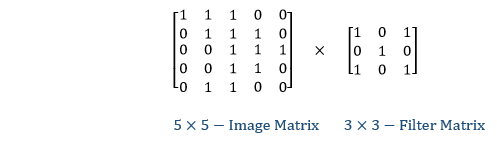
## **5.2.2 CONVOLUTION LAYER**

Convolution layer is the first layer to extract features from an input image. By learning image features using a small square of input data, the convolutional layer preserves the relationship between pixels. It is a mathematical operation which takes two inputs such as image matrix and a kernel or filter.

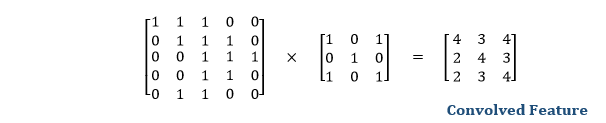
* The dimension of the image matrix is **h×w×d**.
* The dimension of the filter is **fh×fw×d**.
* The dimension of the output is **(h-fh+1) × (w-fw+1) × 1**.



Let's start with consideration a 5\*5 image whose pixel values are 0, 1, and filter matrix 3\*3 as:



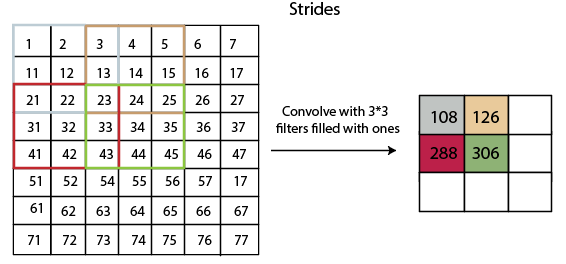
The convolution of 5\*5 image matrix multiplies with 3\*3 filter matrix is called "**Features Map**" and show as an output.



Convolution of an image with different filters can perform an operation such as blur, sharpen, and edge detection by applying filters.

## **5.2.3 STRIDES**

Stride is the number of pixels which are shift over the input matrix. When the stride is equalled to 1, then we move the filters to 1 pixel at a time and similarly, if the stride is equaled to 2, then we move the filters to 2 pixels at a time. The following figure shows that the convolution would work with a stride of 2.

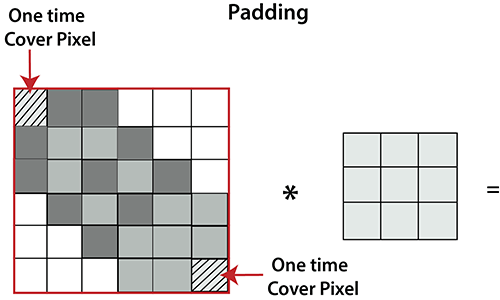


**Fig. 5.2.3 Strides**

## **5.2.4 PADDING**

Padding plays a crucial role in building the convolutional neural network. If the image will get shrink and if we will take a neural network with 100's of layers on it, it will give us a small image after filtered in the end.

If we take a three-by-three filter on top of a grayscale image and do the convolving then what will happen?



**Fig. 5.2.4 Padding**

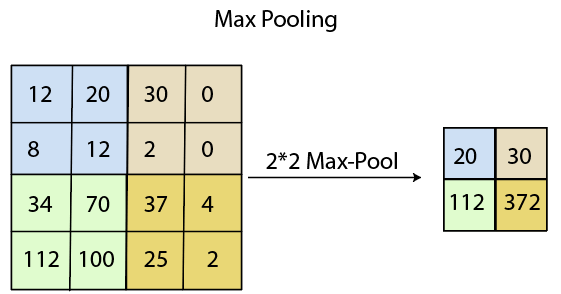
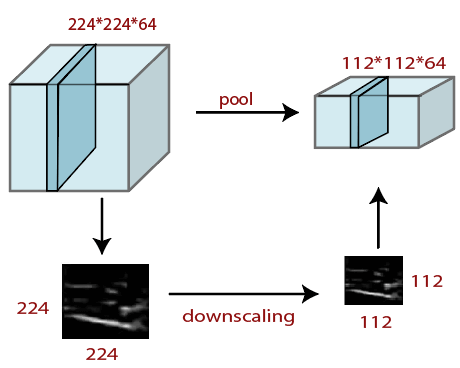
It is clear from the above picture that the pixel in the corner will only get covers one time, but the middle pixel will get covered more than once. It means that we have more information on that middle pixel, so there are two downsides:

* Shrinking outputs
* Losing information on the corner of the image.

To overcome this, we have introduced padding to an image. **"Padding is an additional layer which can add to the border of an image."**

## **5.2.5 POOLING LAYER**

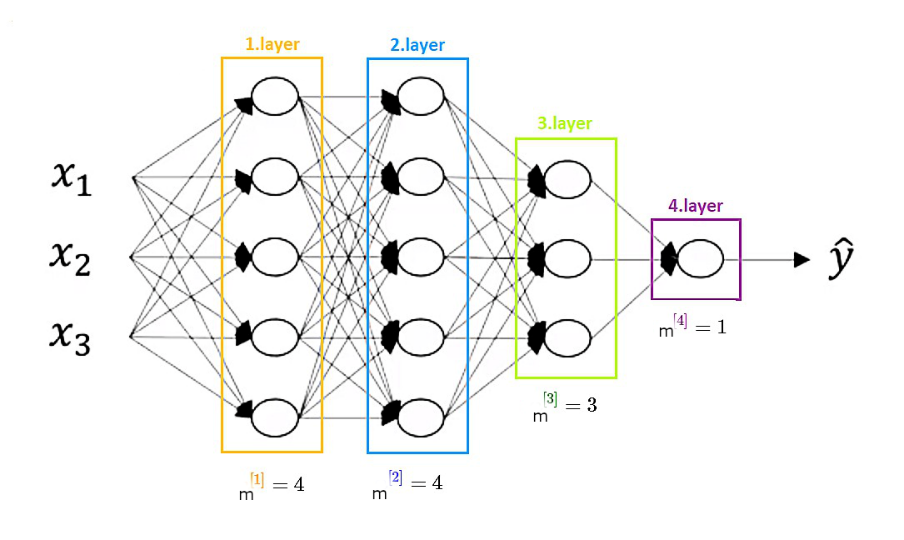
Pooling layer plays an important role in pre-processing of an image. Pooling layer reduces the number of parameters when the images are too large. Pooling is "**downscaling**" of the image obtained from the previous layers. It can be compared to shrinking an image to reduce its pixel density. Spatial pooling is also called down sampling or subsampling, which reduces the dimensionality of each map but retains the important information.

  
  
c

**Fig. 5.2.5 Pooling layer**

## **5.2.6 FULLY CONNECTED LAYER**

The fully connected layer is a layer in which the input from the other layers will be flattened into a vector and sent. It will transform the output into the desired number of classes by the network.



**Fig. 5.2.6 Fully Connected Layer**

In the above diagram, the feature map matrix will be converted into the vector such as **x1, x2, x3... xn**with the help of fully connected layers. We will combine features to create a model and apply the activation function such as **softmax** or **sigmoid** to classify the outputs as a car, dog, truck, etc.

**Image Segmentation**

Image segmentation is a key technique in digital image processing and computer vision that involves dividing an image into distinct segments or regions based on specific characteristics, such as color, intensity, or texture. By simplifying an image, segmentation makes it easier to analyze and extract meaningful information, enhancing applications like object detection, pattern recognition, and decision-making across various domains.

**Types of Image Segmentation Techniques**

Image segmentation methods are categorized into two primary classes: classical computer vision approaches and AI-based techniques. The following are the most popular segmentation techniques:

1. **Thresholding**

Thresholding is a simple segmentation method that converts an image into a binary form based on pixel intensity. By selecting a threshold, pixels are classified into foreground (objects) or background. Otsu’s method and k-means clustering have improved thresholding efficiency.

1. **Clustering Methods**

* K-means Clustering: This iterative technique segments an image into k clusters based on pixel features like color, intensity, or texture. It minimizes the distance between pixels and their assigned cluster center but is sensitive to the initial cluster centers.
* Mean Shift Clustering: Unlike k-means, mean shift clustering doesn’t require predefining the number of clusters. It locates dense regions of pixels and groups them together, making it more flexible for complex images.

1. **Motion-based Segmentation**

Used primarily in video processing, motion-based segmentation identifies moving objects by comparing differences between consecutive frames.

1. **Compression-based Methods**

These techniques focus on image compression, exploiting regularities in the image to identify optimal segmentation that minimizes the data size, useful in image compression and texture-based segmentation.

1. **Histogram-based Methods**

Histograms of pixel intensities or colors are used to detect clusters within an image. These methods are efficient and fast, making them suitable for real-time applications.

1. **Edge Detection**

Edge-based segmentation detects boundaries between regions in an image by identifying sharp changes in intensity. It works well for images with clear object boundaries but may fail when edges are ill-defined.

1. **Dual Clustering Method**

This combines histogram analysis with spatial information to create compact and distinct clusters. It is useful in binary segmentation where clear object-background separation exists.

1. **RESULTS**

The proposed model for brain tumor detection and classification achieved a **training accuracy of 73.90%** using MRI images and a deep learning-based approach. This accuracy demonstrates the model's capability to learn significant features for tumor identification and classification.

The performance was achieved through the integration of the following:

* **Pre-processing Techniques**: Image quality enhancement through noise reduction and histogram equalization, ensuring high-quality input for training.
* **Segmentation**: Effective isolation of tumor regions using thresholding and morphological operations, improving the focus of the model on relevant areas of the MRI images.
* **Deep Learning**: A convolutional neural network (CNN) was designed and trained to classify tumor regions. The CNN effectively captured spatial hierarchies within the images, ensuring robust feature extraction.

**Significance**

While the training accuracy of 73.90% suggests room for improvement, it reflects the model's capacity to handle the inherent variability in medical imaging data. This outcome is a promising step toward the automation of brain tumor diagnosis, potentially reducing reliance on manual methods and speeding up the diagnostic process.

**Future Enhancements**

To improve performance, the following can be explored:

1. **Data Augmentation**: Enhance training data diversity to improve generalization.
2. **Hyperparameter Tuning**: Optimize the model's architecture and learning parameters for better accuracy.
3. **Transfer Learning**: Leverage pre-trained models to boost performance with limited datasets.

This result underscores the potential of AI-driven solutions in advancing medical diagnostics, paving the way for more accurate, efficient, and accessible healthcare technologies.

1. **CONCLUSION**

Machine learning has advanced significantly in medical image analysis, but it still faces challenges, particularly due to the scarcity of labeled datasets. While larger datasets typically improve model performance, in medical imaging, high-quality labeled data is often limited, with many datasets containing fewer than 100 patients. Despite this, several studies have reported good performance with small datasets. For example, a CNN with the GoogLeNet architecture was able to classify CT images with 88-98% accuracy using only 200 training images. This success may be due to the inherent homogeneity of medical images, which are less varied compared to natural images, enabling the model to perform well even with limited data.

To address data scarcity, generative models like Generative Adversarial Networks (GANs) have shown promise. GANs can generate synthetic medical images, augmenting small datasets and helping to overcome the lack of diverse training data. For example, GANs have been used to generate retinal fundus images and segment brain MRIs. These models create realistic synthetic data that can enhance training and improve model performance. Additionally, generative models help address class imbalance, where rare diseases are underrepresented, by generating images of these conditions, though caution must be taken to avoid overfitting.

Class imbalance remains a significant issue in medical imaging, as many training datasets are skewed towards normal or non-pathological images, leading to poor performance on rare conditions. Techniques like data augmentation can help generate more images of underrepresented conditions, while cost-sensitive learning can make models more sensitive to rare diseases. However, these solutions are not without challenges, such as the risk of overfitting.

Beyond technical hurdles, ethical concerns also arise. As AI systems surpass human performance in some tasks, the fear of AI replacing clinicians grows. The “black-box” nature of many machine learning models, which makes their decision-making processes difficult to interpret, raises legal and moral questions. Misdiagnosis or harm resulting from AI-driven medical decisions adds complexity, requiring careful consideration of how AI technologies are integrated into healthcare to ensure responsible use.

1. **REFERENCES**

A. Chaddad, "Automated feature extraction in brain tumor by magnetic resonance imaging using Gaussian mixture models," Int. J. Biomed. Imaging, pp. 1–11, 2015.

A. Isin, C. Direkoğlu, and M. Şah, "Review of MRI-based brain tumor image segmentation using deep learning methods," Procedia Comput. Sci., vol. 102, pp. 317–324, 2016.

A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in Proc. Adv. Neural Inf. Process. Syst., 2012, pp. 1097–1105.

A. K. Jain, Fundamentals of Digital Image Processing, 1989.

C.-J. Hsiao, E. Hing, and J. Ashman, "Trends in electronic health record system use among office-based physicians: United States, 2007–2012," Nat. Health Stat. Rep., vol. 75, pp. 1–18, May 2014.

D. Adams, Publish or Perish 5. Accessed: Oct. 30, 2017. [Online]. Available: <https://harzing.com/resources/publish-or-perish/windows>

D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," Nature, vol. 323, pp. 533–536, Oct. 1986.

D. H. Hubel and T. N. Wiesel, "Receptive fields, binocular interaction and functional architecture in the cat’s visual cortex," J. Physiol., vol. 160, no. 1, pp. 106–154, 1962.

D. Shen, G. Wu, and H.-I. Suk, "Deep learning in medical image analysis," Annu. Rev. Biomed. Eng., vol. 19, pp. 221–248, Mar. 2017.

E. F. Badran, E. G. Mahmoud, and N. Hamdy, "An algorithm for detecting brain tumors in MRI images," IEEE, pp. 368–373, 2010.

E. H. Shortliffe, Computer-Based Medical Consultations: MYCIN, vol. 2. New York, NY, USA: Elsevier, 1976.

F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," Psychol. Rev., vol. 65, no. 6, pp. 365–386, 1958.

G. Litjens et al., "A survey on deep learning in medical image analysis." [Online]. Available: <https://arxiv.org/abs/1702.05747>

H. Greenspan, B. van Ginneken, and R. M. Summers, "Guest editorial deep learning in medical imaging: Overview and future promise of an exciting new technique," IEEE Trans. Med. Imag., vol. 35, no. 5, pp. 1153–1159, May 2016.

H. Mohsen et al., "Classification using deep learning neural networks for brain tumors," Future Comput. Inform. J., vol. 3, pp. 68–71, 2018.

J. Sachdeva et al., "Segmentation, feature extraction, and multiclass brain tumor classification," J. Digit. Imaging, vol. 26, pp. 1141–1150, 2013.

K. Dimililer and A. İhan, "Effect of image enhancement on MRI brain images with neural networks," Procedia Comput. Sci., vol. 102, pp. 39–44, 2016.

K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in Competition and Cooperation in Neural Nets, Berlin, Germany: Springer, 1982, pp. 267–285.

K. Shahzad et al., "Efficient brain tumor detection using image processing techniques," Int. J. Sci. Eng., vol. 7, pp. 1698–1700, 2016.

K. Sudharani, T. C. Sarma, and K. Satya Prasad, "Advanced morphological technique for automatic brain tumor detection and evaluation of statistical parameters," Procedia Technol., vol. 24, pp. 1374–1387, 2015.

K. Suzuki, "Overview of deep learning in medical imaging," Radiol. Phys. Technol., vol. 10, no. 3, pp. 257–273, 2017.

M. Sonka, V. Hlavac, and R. Boyle, Image Processing, Analysis, and Machine Vision. Springer, New York, NY, USA.

N. Nandha Gopal and M. Karnan, "Diagnose brain tumor through MRI using image processing algorithm such as Fuzzy C means along with intelligent optimization techniques," IEEE, 2010.

P. Kumar and B. Vijayakumar, "Brain Tumor MR Image Segmentation and Classification using PCA and RBF kernel based support vector machine," Middle East J. Sci. Res., vol. 23, pp. 2106–2116, 2015.

P. Samriti and P. S. Paramveer, "Brain tumor detection using image segmentation," IJEDR, vol. 4, 2016.

Python 3 Tutorial. [Online]. Available: <https://docs.python.org/3/tutorial/>

R. Bhattacharjee and M. Chakarborty, "Brain tumor detection from MR images: Image processing, slicing and PCA based reconstruction," IEEE, pp. 97–101, 2012.

R. C. Gonzalez and R. E. Woods, Digital Image Processing, 3rd ed. Pearson Education, 2009.

R. Smith-Bindman et al., "Use of diagnostic imaging studies and associated radiation exposure for patients enrolled in large integrated health care systems, 1996–2010," JAMA, vol. 307, no. 22, pp. 2400–2409, 2012.

S. Damodharan, S. Selvaraj, and R. Dhanasekaran, "Combining tissue segmentation and neural network for brain tumor detection," Int. Arab J. Inf. Technol., vol. 12, pp. 42–52, 2015.

S. Ghanavati et al., "Automatic brain tumor detection in MR images," IEEE, pp. 574–577, 2012.

S. Jayaraman, Digital Image Processing.

V. S. Shirsat and S. S. Kawathekar, "Classification of brain cancer detection by using magnetic resonance imaging," Int. J. Eng. Res. Technol., vol. 3, pp. 1698–1700, 2014.

W. Cui et al., "Localized FCM clustering with spatial information for medical image segmentation and bias field estimation," Int. J. Biomed. Imaging, pp. 1–8, 2013.

W. K. Pratt, Digital Image Processing: PIKS Scientific Inside, 4th ed. John Wiley & Sons, Inc., 2007.

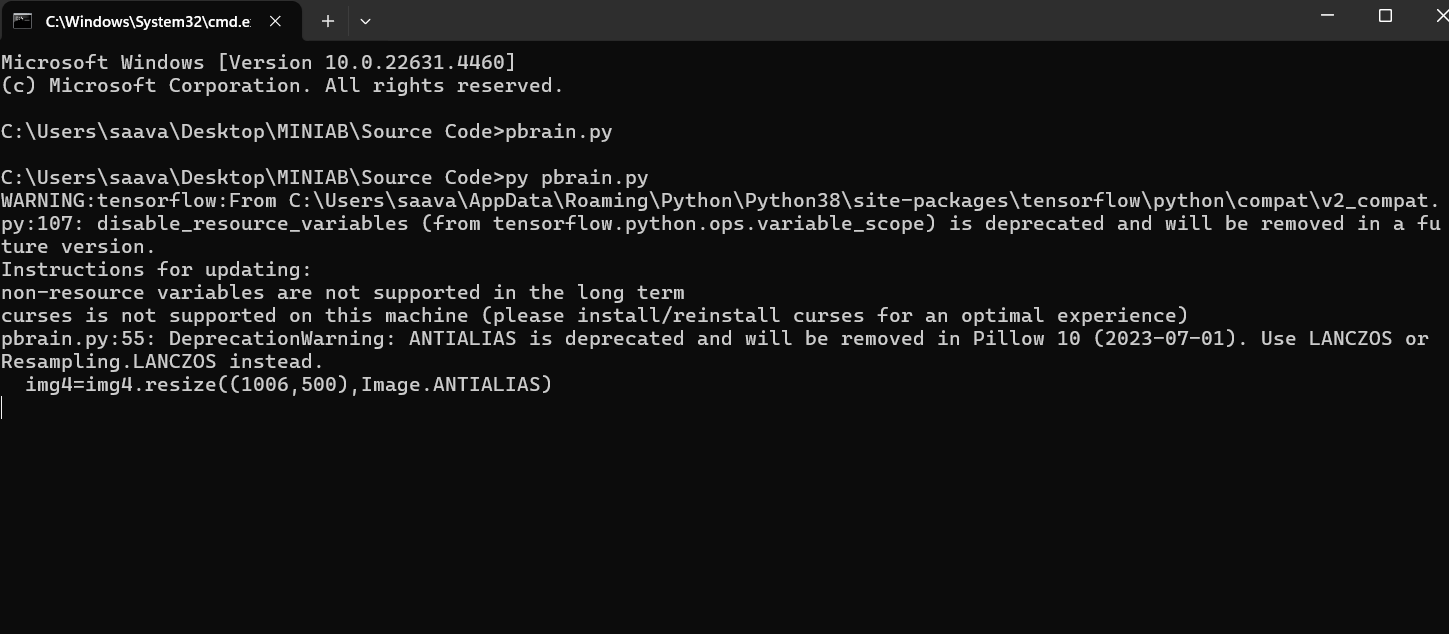
W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," Bull. Math. Biol., vol. 5, no. 4, pp. 115–133, 1943.

Y. LeCun et al., "Backpropagation applied to handwritten zip code recognition," Neural Comput., vol. 1, no. 4, pp. 541–551, 1989.

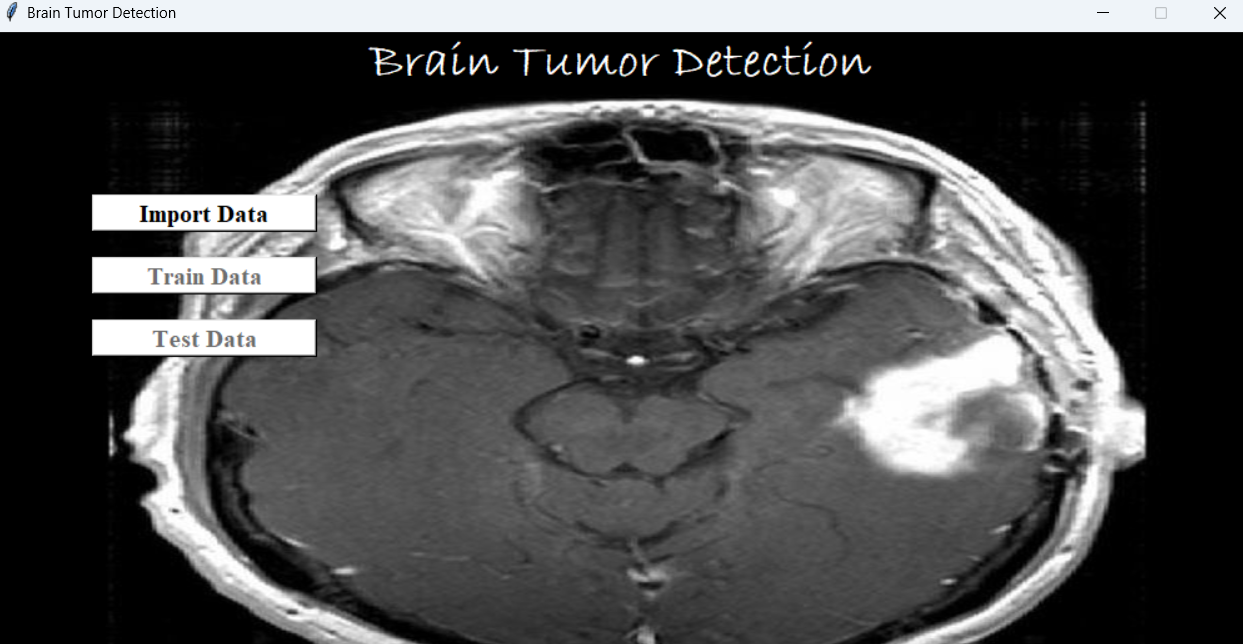
**APPENDIX – I**

**SCREENSHOTS**

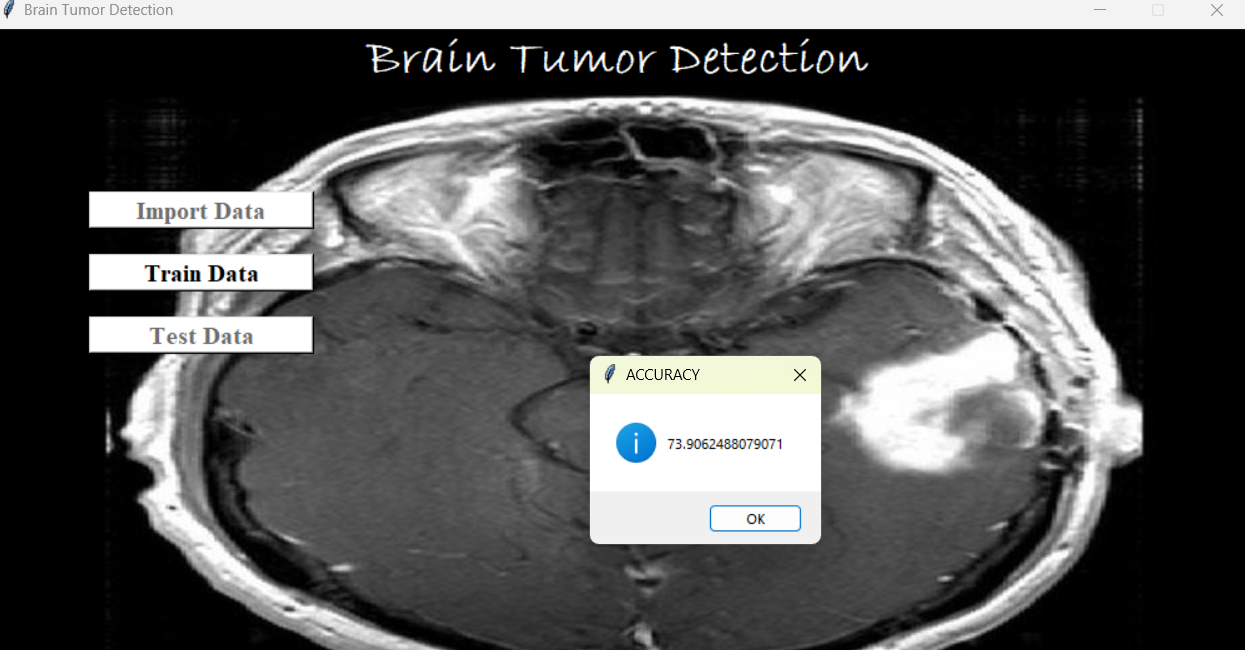
**Step-1:**



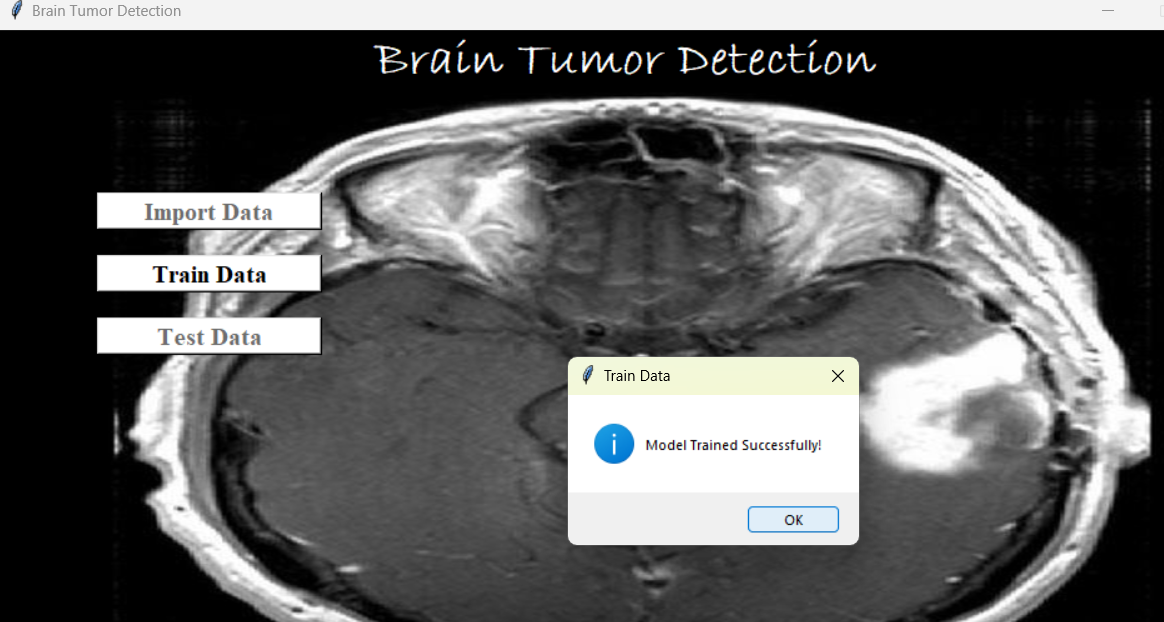
**Step-2:**



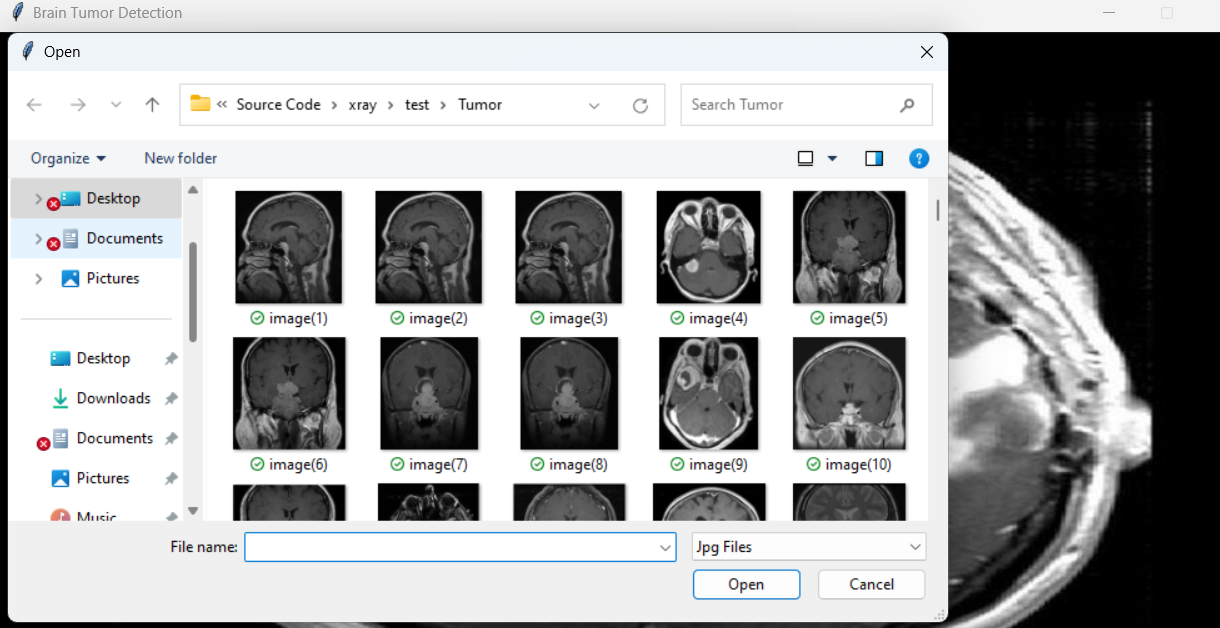
**Step-3:**



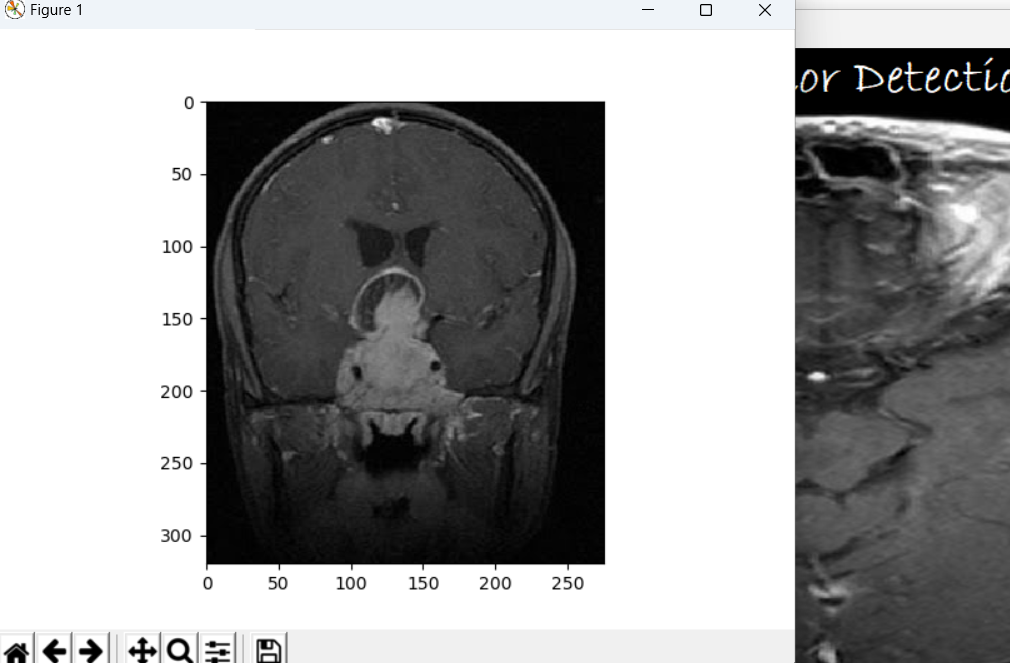
**Step-4:**



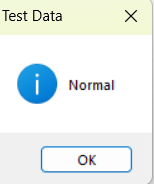
**Step-5:**



**Step-6:**



**Step-7:**



**APPENDIX – II**

**SAMPLE CODE**

from codecs import BOM32\_BE

from ctypes import alignment

from unittest import result

from xml.dom.expatbuilder import parseString

import numpy as np

import pandas as pd

import pydicom as dicom

import os

import matplotlib.pyplot as plt

import cv2

import math

import numpy as np

import matplotlib.pyplot as plt #for plotting things

import os

from tkinter import filedialog

from PIL import Image,ImageTk

from tkinter.filedialog import askopenfile

#print(os.listdir("../input"))

# Keras Libraries

import keras

from keras.models import Sequential

from keras.layers import Conv2D

from keras.layers import MaxPooling2D

from keras.layers import Flatten

from keras.layers import Dense

from keras.preprocessing.image import ImageDataGenerator, load\_img

import tensorflow.\_api.v2.compat.v1 as tf

import pandas as pd

import tflearn

from tflearn.layers.conv import conv\_3d, max\_pool\_3d

from tflearn.layers.core import input\_data, dropout, fully\_connected

from tflearn.layers.estimator import regression

import numpy as np

import matplotlib.pyplot as plt

from sklearn.metrics import confusion\_matrix

from tkinter import \*

from tkinter import messagebox,ttk

import tkinter as tk

from PIL import Image,ImageTk

cnn = Sequential()#Convolution

class LCD\_CNN:

def \_\_init\_\_(self,root):

self.root=root

#window size

self.root.geometry("1006x500+0+0")

self.root.resizable(False, False)

self.root.title("Brain Tumor Detection")

img4=Image.open(r"xray/train/Tumor/Y55.jpg")

img4=img4.resize((1006,500),Image.ANTIALIAS)

self.photoimg4=ImageTk.PhotoImage(img4)

bg\_img=Label(self.root,image=self.photoimg4)

bg\_img.place(x=0,y=50,width=1006,height=500)

# title Label

title\_lbl=Label(text="Brain Tumor Detection",font=("Bradley Hand ITC",30,"bold"),bg="black",fg="white",)

title\_lbl.place(x=0,y=0,width=1006,height=50)

#photos Button

# img10=Image.open(r"Images\opencv\_face\_reco\_more\_data.jpg")

# img10=img10.resize((180,140),Image.ANTIALIAS)

# self.photoimg10=ImageTk.PhotoImage(img10)

#button 1

self.b1=Button(text="Import Data",cursor="hand2",command=self.import\_data,font=("Times New Roman",15,"bold"),bg="white",fg="black")

self.b1.place(x=80,y=130,width=180,height=30)

#button 3

self.b3=Button(text="Train Data",cursor="hand2",command=self.train\_data,font=("Times New Roman",15,"bold"),bg="white",fg="black")

self.b3.place(x=80,y=180,width=180,height=30)

self.b3["state"] = "disabled"

self.b3.config(cursor="arrow")

#button 4

self.b4=Button(text="Test Data",cursor="hand2",command=self.test\_data,font=("Times New Roman",15,"bold"),bg="white",fg="black")

self.b4.place(x=80,y=230,width=180,height=30)

self.b4["state"] = "disabled"

self.b4.config(cursor="arrow")

def import\_data(self):

##Data directory

self.dataDirectory = 'xray/train/'

self.TumorPatients = os.listdir(self.dataDirectory)

##Setting x\*y size to 50

self.size = 10

## Setting z-dimension (number of slices to 20)

self.NoSlices = 5

messagebox.showinfo("Import Data" , "Data Imported Successfully!")

self.b1["state"] = "disabled"

self.b1.config(cursor="arrow")

self.b3["state"] = "normal"

self.b3.config(cursor="hand2")

def train\_data(self):

cnn.add(Conv2D(32, (3, 3), activation="relu", input\_shape=(64, 64, 3)))

#Pooling

cnn.add(MaxPooling2D(pool\_size = (2, 2)))

# 2nd Convolution

cnn.add(Conv2D(32, (3, 3), activation="relu"))

# 2nd Pooling layer

cnn.add(MaxPooling2D(pool\_size = (2, 2)))

# 3nd Convolution

cnn.add(Conv2D(32, (3, 3), activation="relu"))

# 3nd Pooling layer

cnn.add(MaxPooling2D(pool\_size = (2, 2)))

# Flatten the layer

cnn.add(Flatten())

# Fully Connected Layers

cnn.add(Dense(activation = 'relu', units = 128))

cnn.add(Dense(activation = 'sigmoid', units = 1))

# Compile the Neural network

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

num\_of\_test\_samples = 200

batch\_size = 32

# Fitting the CNN to the images

# The function ImageDataGenerator augments your image by iterating through image as your CNN is getting ready to process that image

train\_datagen = ImageDataGenerator(rescale = 1./255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

test\_datagen = ImageDataGenerator(rescale = 1./255) #Image normalization.

training\_set = train\_datagen.flow\_from\_directory('xray/train',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

validation\_generator = test\_datagen.flow\_from\_directory('xray/val/',

target\_size=(64, 64),

batch\_size=32,

class\_mode='binary')

test\_set = test\_datagen.flow\_from\_directory('xray/test',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'binary')

cnn\_model = cnn.fit\_generator(training\_set,

steps\_per\_epoch = 8,

epochs = 9,

validation\_data = validation\_generator,

validation\_steps = 20)

test\_accu = cnn.evaluate\_generator(test\_set,steps=20)

print('The testing accuracy is :',test\_accu[1]\*100, '%')

messagebox.showinfo("ACCURACY" ,test\_accu[1]\*100)

messagebox.showinfo("Train Data" , "Model Trained Successfully!")

self.b3["state"] = "disabled"

self.b3.config(cursor="arrow")

self.b4["state"] = "normal"

self.b4.config(cursor="hand2")

def test\_data(self):

f\_types = [('Jpg Files', '\*.jpg')]

filename = filedialog.askopenfilename(filetypes=f\_types)

img = ImageTk.PhotoImage(file=filename)

from keras.preprocessing import image

import matplotlib.image as mpimg

img = mpimg.imread(filename)

plt.imshow(img)

plt.show()

img = image.load\_img(filename, target\_size=(64, 64))

x = image.img\_to\_array(img)

x = np.expand\_dims(x, axis=0)

classes = cnn.predict(x)

print(classes)

def ans():

if classes>0.5:

return("Tumor")

else:

return("Normal")

messagebox.showinfo("Test Data" , ans())

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

root=Tk()

obj=LCD\_CNN(root)

root.mainloop()