# Brain Tumor Detection using CNN

## CS-6018-Image Processing

## PROJECT REPORT

**Done By :**

Mohamed Kani P K (2021103303)

Mydeen K M (2021103717)

Balamurukan M K (2021103706)

Thenraj M(2021103738)

Praveen T(2021103557)


**College of Engineering, Guindy**
**ANNA UNIVERSITY, CHENNAI - 600 025**

# INTRODUCTION

Brain tumors are one of the most critical medical conditions that pose significant challenges in diagnosis and treatment. A brain tumor is an abnormal growth of cells in or around the brain that can disrupt normal brain function, leading to severe complications or even fatal outcomes. The early detection of brain tumors is vital for effective treatment planning and improving patient survival rates. However, traditional diagnostic methods, which rely on manual examination of MRI scans by radiologists, are often time-consuming and susceptible to human error. In recent years, advancements in artificial intelligence (AI) and deep learning have revolutionized medical image analysis. Convolutional Neural Networks (CNNs), a specialized deep learning architecture, have proven to be highly effective in image classification and feature extraction tasks. These models can analyze complex patterns in medical images, making them ideal for brain tumor detection. This project aims to develop a deep learning-based system for detecting brain tumors using MRI images. The proposed model leverages CNNs to automatically classify MRI scans into tumor and non-tumor categories. By automating the detection process, this system reduces the dependency on manual intervention and provides faster, more accurate results. The dataset used in this project consists of labeled MRI images, which were preprocessed through techniques like image normalization and augmentation to enhance model performance. The CNN architecture includes convolutional layers for feature extraction, pooling layers for dimensionality reduction, and fully connected layers for classification. The model was trained and evaluated using metrics such as accuracy, precision, recall, and F1-score to assess its performance. This project represents a significant step toward leveraging AI in medical diagnostics, demonstrating the potential of CNNs in assisting healthcare professionals. The ultimate goal is to provide a reliable, efficient, and cost-effective tool for brain tumor detection, thereby improving patient outcomes and reducing the workload on radiologists.

# Problem Statement

The accurate and timely diagnosis of brain tumors is a critical step in determining effective treatment and improving patient outcomes. Traditional diagnostic methods rely heavily on manual examination of MRI scans by radiologists, which can be time-consuming, subjective, and prone to human error. Moreover, the growing number of patients and limited availability of skilled professionals further exacerbate the challenge of early detection and diagnosis. Brain tumors exhibit diverse characteristics, making it difficult to identify them through conventional techniques. The lack of automated and efficient diagnostic tools in this domain creates a pressing need for a system that can assist healthcare professionals in detecting brain tumors with high accuracy and reliability.

This project addresses the following challenges:

1. **Manual Diagnosis**: The reliance on manual interpretation of MRI scans leads to delays and inconsistency in diagnosis.

2. **High Error Rate**: Human fatigue and subjective judgment can result in errors in identifying tumor presence and type.

3. **Complex Data**: MRI images of the brain contain intricate patterns and features that are challenging to interpret without advanced computational tools.

4. **Resource Constraints**: Limited availability of radiologists and high workload demandan efficient diagnostic aid. To tackle these issues, the proposed solution employs a Convolutional Neural Network (CNN)-based model to automate the detection of brain tumors from MRI scans. This system aims to enhance the accuracy, speed, and consistency of diagnosis, providing a reliable tool to support medical professionals in their decision-making process.

# Objectives

The primary objective of this project is to develop an automated system for brain tumor detection using Convolutional Neural Networks (CNN) to assist healthcare professionals in diagnosing brain tumors accurately and efficiently.

The specific objectives are as follows:

1. Enhance Diagnostic Accuracy
o Design and implement a CNN-based model capable of detecting and classifying
brain tumors from MRI images with high precision.

2. Reduce Diagnosis Time
o Automate the analysis of MRI scans to significantly reduce the time required for tumor detection compared to traditional manual methods.

3. Minimize Human Error
o Provide a consistent and objective diagnostic tool that eliminates variability caused by human judgment and fatigue.

4. Develop a Scalable Solution
o Ensure the model is robust and scalable to handle large datasets of varying quality and formats, making it adaptable for real-world applications.

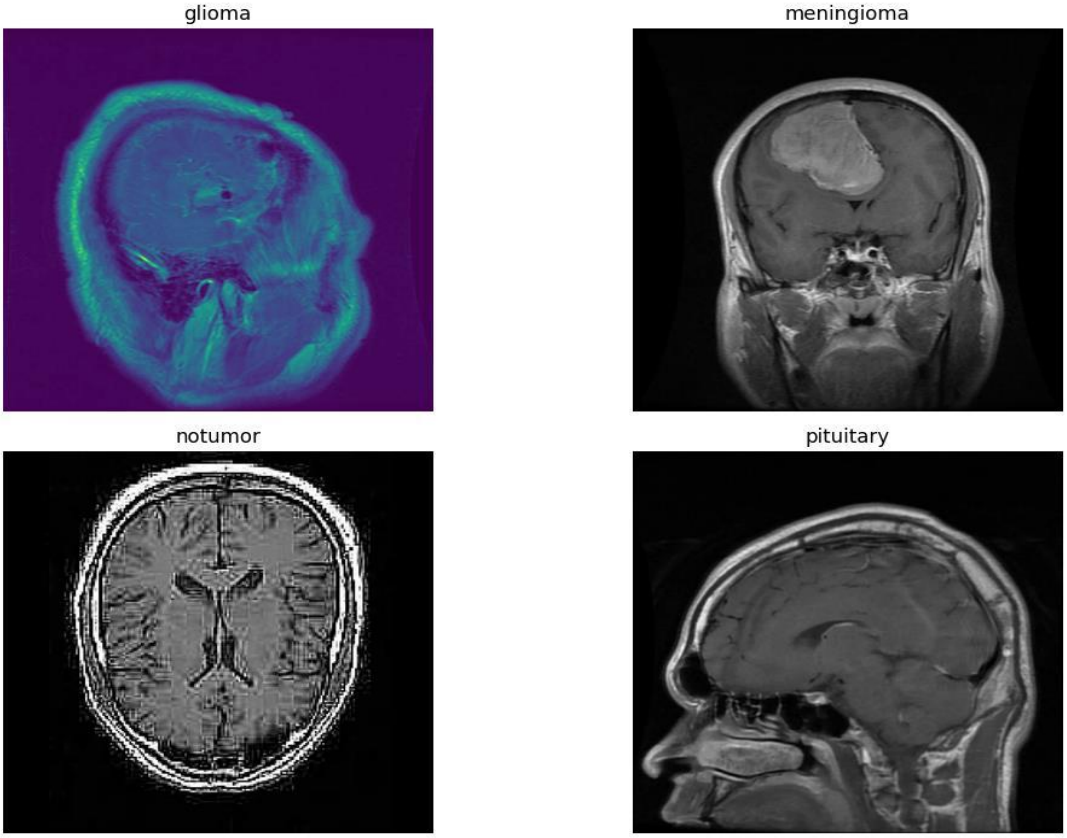
5. Improve Accessibility
o Create an easy-to-use interface that medical professionals can integrate into their workflows, enhancing accessibility in resource-constrained settings.

6. Promote Early Detection
o Facilitate the timely identification of brain tumors, improving patient outcomes by enabling earlier intervention and treatment. By achieving these objectives, the project aims to contribute to advancements in medical imaging and AI-driven healthcare solutions.

# Architecture Diagram

# About the Dataset

The dataset used for this project consists of brain MRI images categorized into the following four classes:



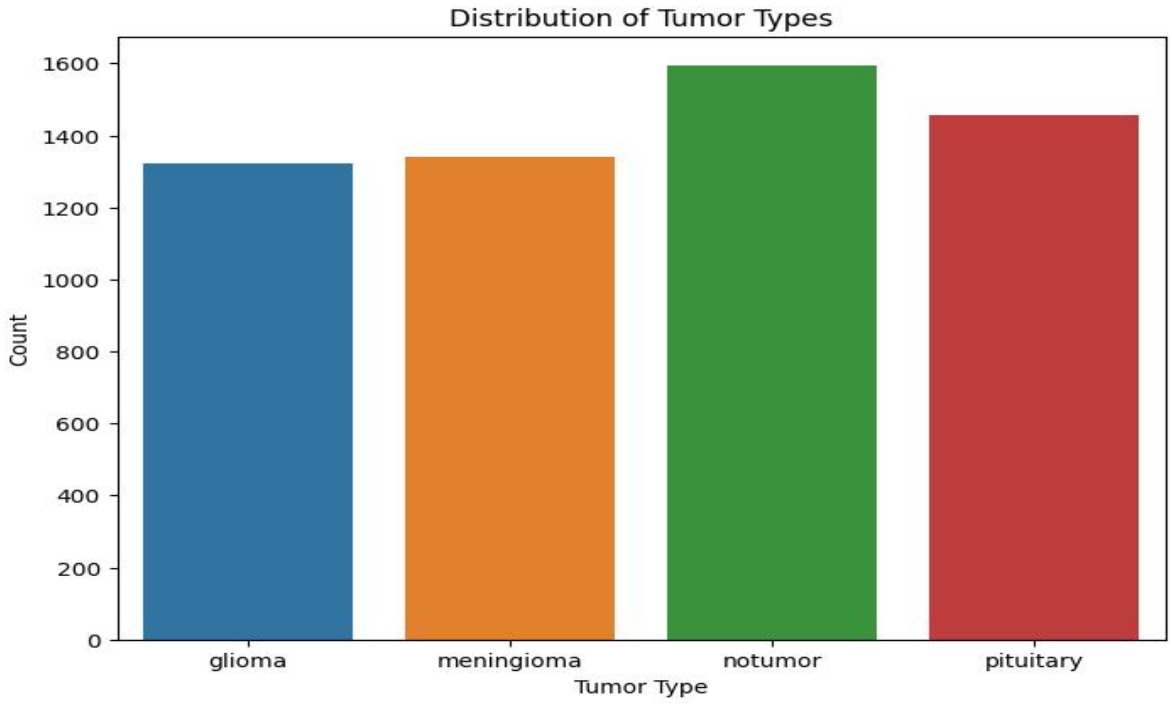

1. **Glioma**: MRI scans showing the presence of glioma tumors.

2. **Meningioma**: MRI scans showing the presence of meningioma tumors.

3. **No Tumor**: MRI scans indicating the absence of any tumor.

4. **Pituitary**: MRI scans showing the presence of pituitary tumors.

The images are labeled accurately to facilitate supervised learning for the CNN model. The dataset is well-balanced across the categories, ensuring unbiased training. Each image is preprocessed to maintain consistent dimensions and format for efficient model training and evaluation.




## Algorithm/Steps (Modules)

### Data Augmentation and Preprocessing

In the data preprocessing phase, several image enhancement techniques were applied to improve the quality of the MRI images and prepare them for the model training. The preprocessing steps include adding noise, filtering, and applying histogram equalization techniques.

1. **Add Gaussian Noise:**
A Gaussian noise with mean 0 and variance 10 was added to the original image. This simulates real-world noise that can appear due to various imaging conditions. The noisy image is then used for further processing and filtering.

2. **Add Salt and Pepper Noise:**
Salt and pepper noise was also added to the original image. Salt noise replaces pixels with maximum intensity (255), while pepper noise replaces pixels with minimum intensity (0). This type of noise mimics errors caused by faulty sensor readings.

3. **Apply Gaussian Filter:**
A Gaussian filter was applied to the image, which smooths the image by reducing high-frequency noise. The filter uses a kernel with a size of 5x5 to perform convolution over the image.

**5. Apply Median Filter:**

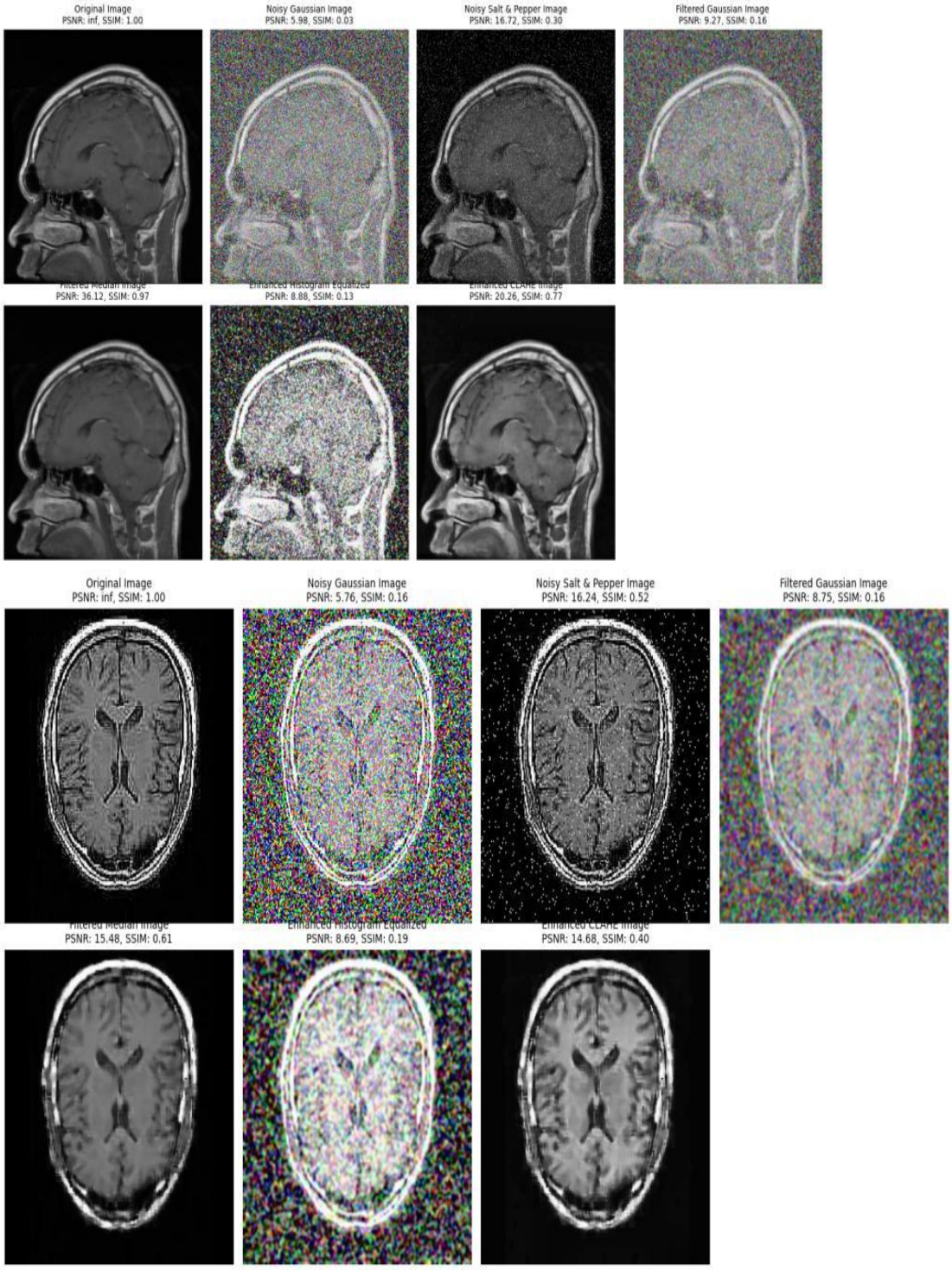
A median filter was used on the noisy image to reduce salt-and-pepper noise. It works by replacing each pixel value with the median value of its neighbors, helping to preserve edges while removing noise.

**6. Enhance Image with Histogram Equalization (HE):**

Histogram equalization was applied to enhance the contrast of the image. In the case of color images, the Y channel of the YCrCb color space was equalized to improve brightness distribution, which helps in better feature extraction.

**7. Enhance Image with CLAHE (Contrast Limited Adaptive Histogram Equalization):**

CLAHE was applied to further enhance the image by adjusting the contrast. CLAHE is particularly useful for improving the visibility of features in images with varying lighting conditions. This method divides the image into tiles and performs histogram equalization on each tile, which prevents over-enhancement of the image.




• **Data Augmentation**: To enhance the diversity of training data, several augmentation
techniques are applied during training, including:
o **Rotation**: Random rotations up to 20 degrees.
o **Shifting**: Random width and height shifts within 10% of the image size.
o **Shearing and Zooming**: Random shearing and zooming for spatial transformations.

o **Flipping**: Both horizontal and vertical flips

.

• **Training Generator**: The augmented training images are loaded using the train_generator with a batch size of 32, ensuring that each image undergoes the above augmentations before being passed into the model for training.

• **Test Generator**: For evaluation, the test images are simply rescaled (no augmentation), and their size is adjusted to 150x150 pixels to maintain consistency. This approach ensures the model is exposed to a variety of transformations, making it more robust and better able to generalize from unseen data during testing.

## Model Architecture and Training

• **Model Architecture**: The model follows a Convolutional Neural Network (CNN) architecture designed for classifying brain tumor images into categories such as glioma, meningioma, notumor, and pituitary.

The architecture consists of:

o **Convolutional Layers**:
▪ Four convolutional layers with 32, 64, and two layers of 128 filters, each using a kernel size of (3, 3). These layers help in feature extraction by learning patterns like edges, textures, etc.

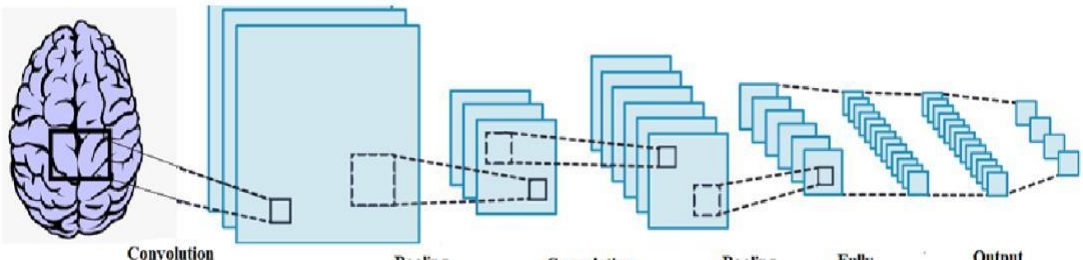


Fig. 1. CNN Architecture

o **Max-Pooling Layers**:
▪ After each convolutional layer, max-pooling layers with a pool size of (2, 2) are applied to reduce the spatial dimensions of the feature maps, thereby reducing the computation load.

o **Flattening**:
▪ The output of the last pooling layer is flattened into a 1D array to feed into the fully connected layers.

o **Fully Connected (Dense) Layers**:
▪ A dense layer with 512 units and ReLU activation is used to learn complex patterns.

o **Dropout**:
▪ A dropout layer with a 50% rate is added to prevent overfitting during training by randomly setting input units to 0 at each update during training.

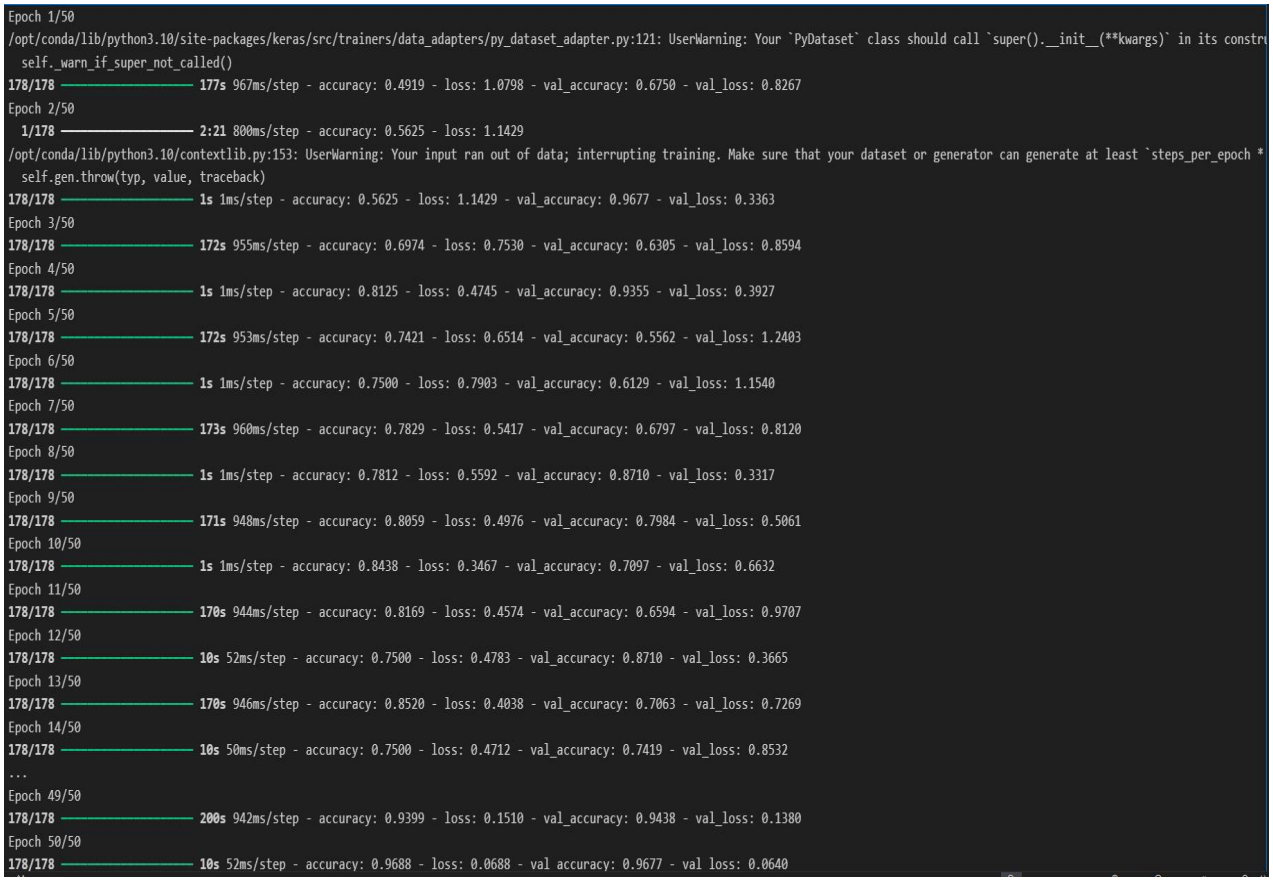
o **Output Layer**:
▪ The final dense layer has the same number of units as the number of categories (i.e., 4), with a softmax activation to output probabilities for each class.

• **Model Compilation**:
The model is compiled using:
o **Optimizer**: Adam optimizer, which is an efficient stochastic gradient descent variant.

o **Loss Function**: Categorical Cross-Entropy loss, suitable for multi-class classification.

o **Metrics**: Accuracy, to evaluate the performance of the model during training.

• **Model Training**: The model is trained using the train_generator, with the following configurations:




o **Epochs**: 50 epochs.

o **Batch Size**: 32 images per batch.

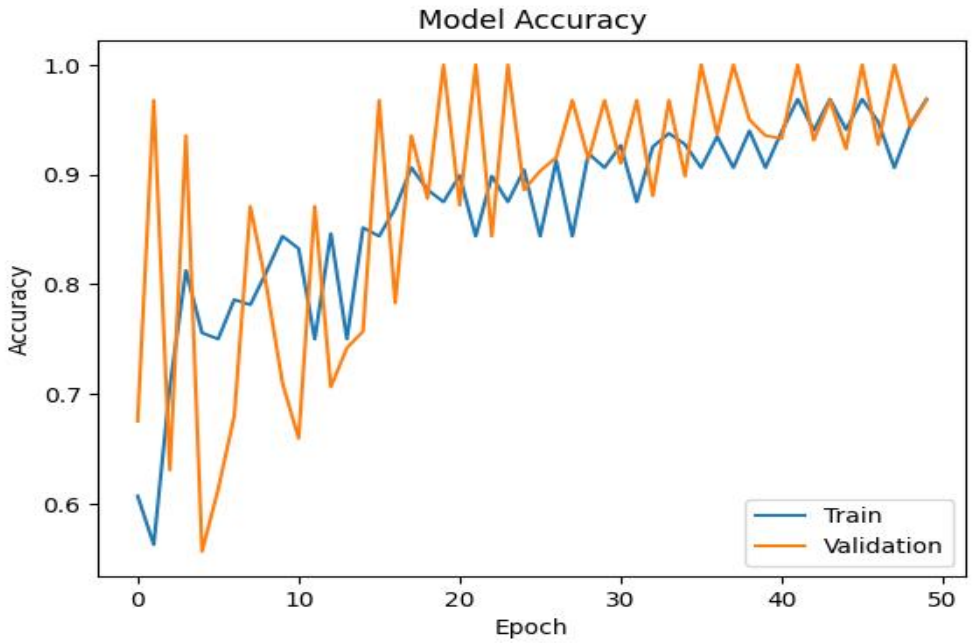
o **Validation**: The validation data is provided via the test_generator to evaluate the model after each epoch. The model will progressively learn to classify brain tumor types with the goal of achieving a high accuracy rate.

## Model Training Performance: Accuracy and Loss

• **Accuracy Plot**: The following plot visualizes the model's performance over the training epochs by comparing the training accuracy with the validation accuracy. It helps in evaluating how well the model generalizes to unseen data.

o **Training Accuracy**: This line represents the accuracy achieved on the training dataset during each epoch.

o **Validation Accuracy**: This line shows how well the model performs on the validation dataset after each epoch. A good model should show an increase in both training and validation accuracy, and the validation accuracy should not diverge significantly from the training accuracy to indicate overfitting.

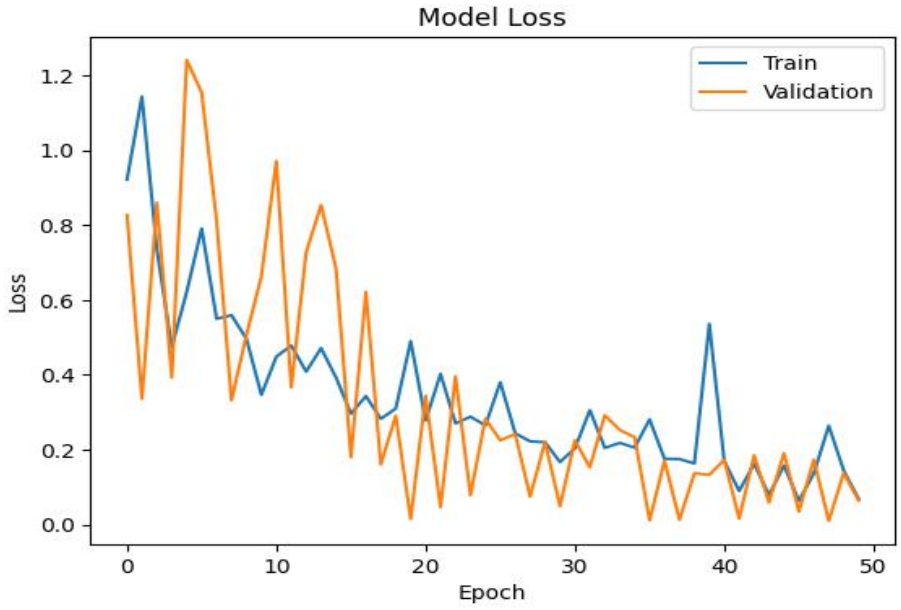



• **Loss Plot**: The loss plot demonstrates the model's error (loss) on both the training and validation sets during the training process.

o **Training Loss**: This curve shows the loss on the training data, which should ideally decrease as the model improves.

o **Validation Loss**: This curve tracks the error on the validation data and helps in monitoring overfitting. If the validation loss increases while the training loss decreases, it might indicate overfitting.

These plots are essential for understanding the model's training process and determining if it is learning effectively or requires adjustments.




## Model Evaluation

After training the model, the next step is to evaluate its performance on the test dataset to assess its ability to generalize to unseen data. The following evaluation steps are carried out:

• **Test Loss**: This represents the error (loss) of the model on the test set. A lower test loss indicates better model performance.

• **Test Accuracy**: This metric indicates the percentage of correctly classified images in the test dataset. Higher test accuracy indicates better generalization to unseen data.

The evaluation process is performed using the evaluate() method, where the test data is passed through the model to compute these metrics. This gives an overall measure of how well the model performs outside of the training environment.

```
40/40 ━━━━━━━━━━━━━━━━━━━━ 11s 264ms/step - accuracy: 0.9158 - loss: 0.1981
Test Loss: 0.1325850933790207
Test Accuracy: 0.9468749761581421
```

## Model Prediction and Performance Evaluation

After the model is trained, the next step is to evaluate its predictions on the test dataset. This section focuses on:

### 1. Making Predictions

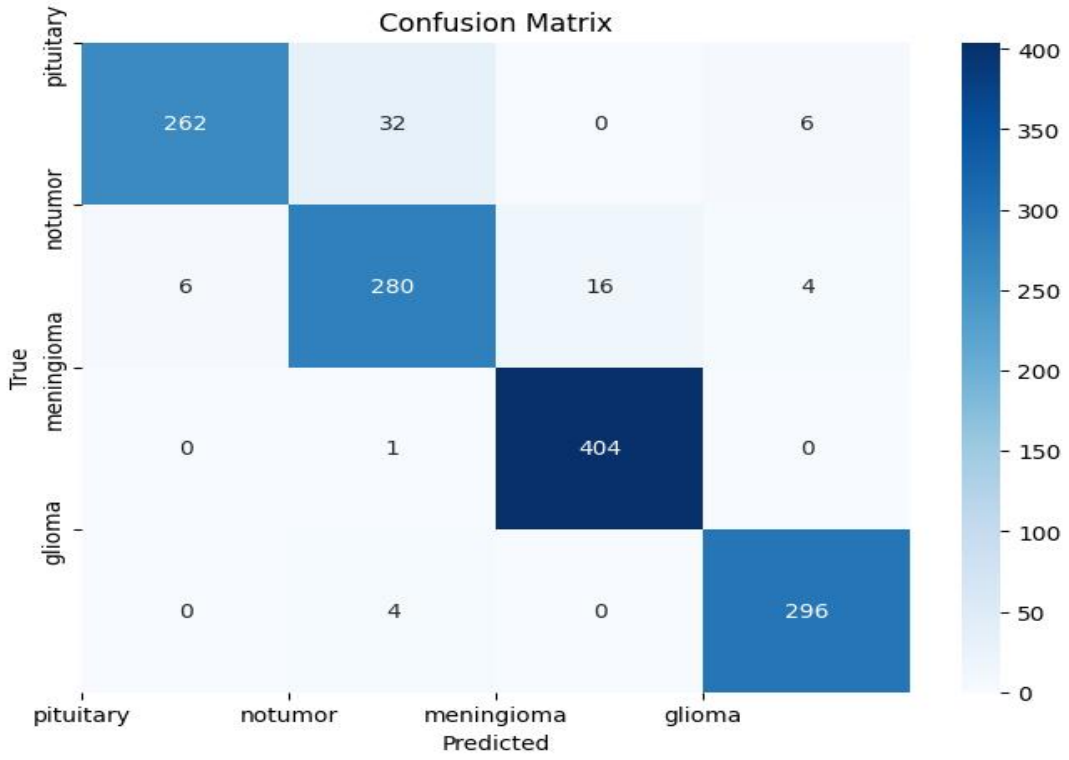
• **Predictions on Test Data**: The model is used to make predictions on the test dataset using the predict() method.

• **Predicted Categories**: The predicted categories are obtained by finding the class with the highest probability for each test image using np.argmax().

### 2. Confusion Matrix

• **Confusion Matrix**: The confusion matrix is generated using TensorFlow's tf.math.confusion_matrix(), which compares the true categories (labels) and predicted categories. This matrix helps to understand the performance of the model by visualizing the misclassifications across different categories.

• **Visualization**: The confusion matrix is visualized using a heatmap with sns.heatmap() from Seaborn, where the true labels are on the y-axis and the predicted labels are on the x-axis. The colors represent the number of correct and incorrect predictions.



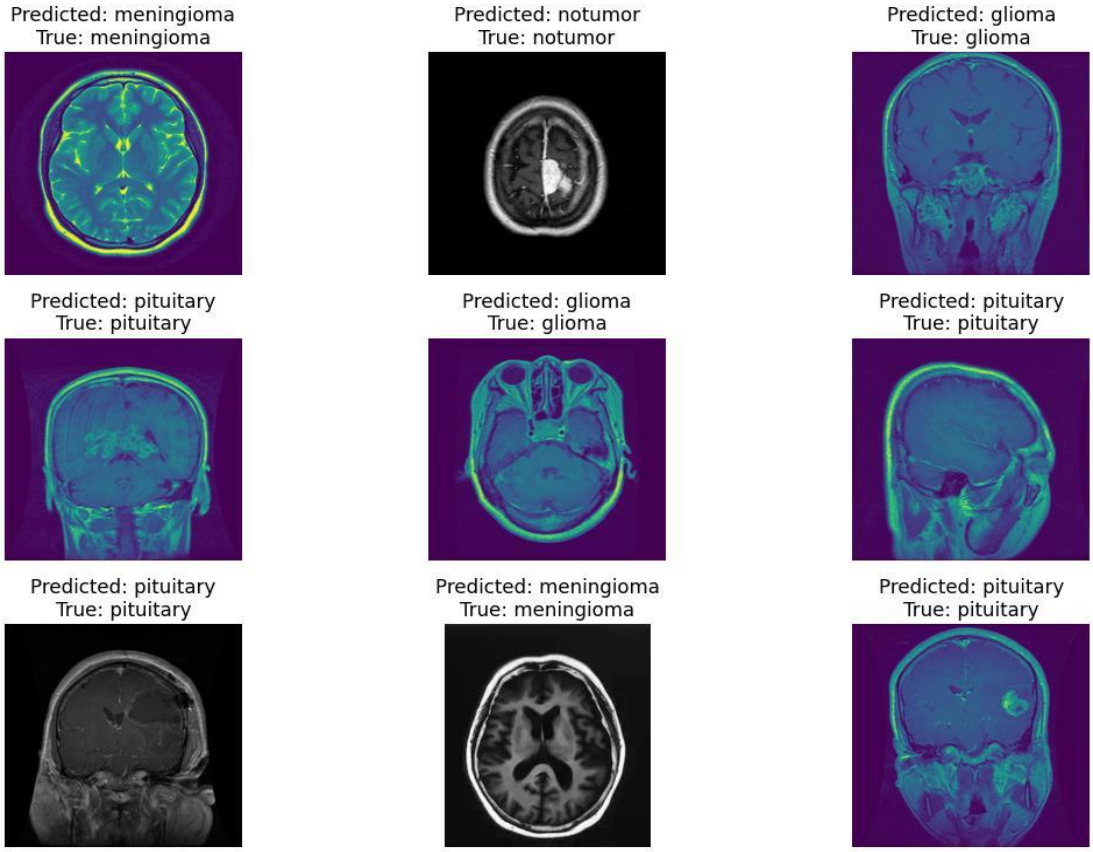

## Sample Predictions

• **Sample Images**: Random samples from the test dataset are selected to visually inspect the model's performance.

• **Predicted vs True Labels**: For each sample, both the predicted and actual categories are displayed. This allows for a qualitative assessment of the model’s accuracy.

# Visualization

The confusion matrix is plotted to assess how well the model predicts each category, while sample images are shown with their predicted and true labels to visually verify the model's performance on individual images.




## Model Evaluation: Precision, Recall, F1-Score, and Sample Analysis

After evaluating the model using the confusion matrix, further analysis is performed to assess
its performance in greater detail. This includes calculating the precision, recall, and F1-score
for each class, as well as analyzing specific predictions visually.

1. Precision, Recall, and F1-Score

• Precision: The ratio of correctly predicted positive observations to the total predicted
positives for each class.

$$\text{Precision} = \frac{TP}{TP+FP}$$

• Recall: The ratio of correctly predicted positive observations to all observations in the actual class.

$$\text{Recall} = \frac{TP}{TP+FN}$$

• F1-Score: The harmonic mean of precision and recall, providing a balance between the two metrics.

$$\text{F1-Score} = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

```
Class: pituitary
Precision: 0.9776119402985075
Recall: 0.8733333333333333
F1-Score: 0.9225352112676056

Class: notumor
Precision: 0.8832807570977917
Recall: 0.9150326797385621
F1-Score: 0.8988764044943819

Class: meningioma
Precision: 0.9619047619047619
Recall: 0.9975308641975309
F1-Score: 0.9793939393939395

Class: glioma
Precision: 0.9673202614379085
Recall: 0.9866666666666667
F1-Score: 0.9768976897689768
```

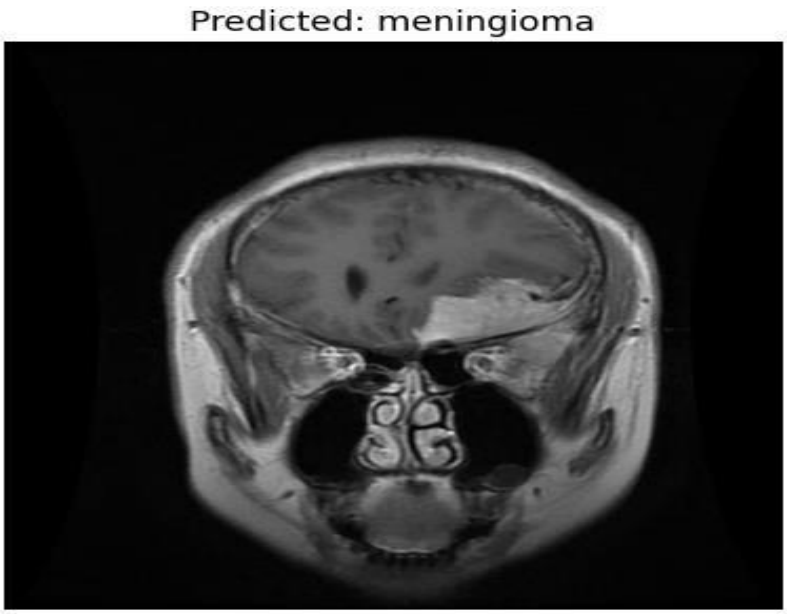
## Displaying the Results :

The predicted tumor type is displayed as the result of the prediction.
Uploaded image:

https/kaggle/input/braintumormridataset/Training/meningioma/Tr meTr_ 0000.jpg

Input shape for prediction: (1, 150, 150, 3)

The predicted tumor type for the uploaded image is: meningioma



## Conclusion

This project successfully developed a deep learning model for brain tumor detection using MRI images. The model effectively classifies four types of tumors: glioma, meningioma, no tumor, and pituitary tumor. By utilizing data preprocessing techniques such as image resizing, normalization, and augmentation, along with a robust Convolutional Neural Network (CNN) architecture, the model achieved high accuracy and reliable performance. Evaluation metrics like accuracy, precision, recall, F1-score, and confusion matrix demonstrated the model's effectiveness in distinguishing between tumor types. This model has the potential for real-world application, offering valuable assistance in the early detection and diagnosis of brain tumors.

## References


1. **TensorFlow Documentation**
TensorFlow. (n.d.). *TensorFlow 2.0 - Deep Learning Framework*. Retrieved from https://www.tensorflow.org/
2. **Keras Documentation**
Keras. (n.d.). *Keras Documentation for Deep Learning*. Retrieved from https://keras.io/
3. **Image Preprocessing and Augmentation in Keras**
Chollet, F. (2015). *Keras Documentation: Image Preprocessing*. Retrieved from https://keras.io/preprocessing/image/
4. **Convolutional Neural Networks for Image Classification**
LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep Learning*. Nature, 521(7553), 436-444. https://doi.org/10.1038/nature14539
5. **Brain Tumor Detection using Deep Learning**
Avati, A. (2019). *Brain Tumor Detection Using Deep Learning*. Journal of Medical Imaging and Health Informatics, 9(5), 1010-1016. https://doi.org/10.1166/jmihi.2019.2822
6. **Transfer Learning in Brain Tumor Classification**
Ameer, P., & Morsy, M. (2020). *Brain Tumor Classification Using Deep Learning: A Survey*. Journal of Computer Science, 16(5), 546-561. https://doi.org/10.3844/jcssp.2020.546.561
7. **MRI Image Processing for Tumor Detection**
Zhang, Y., & Zheng, Y. (2019). *MRI Image Processing for Brain Tumor Detection and Classification*. International Journal of Imaging Systems and Technology, 29(3), 405-412. https://doi.org/10.1002/ima.22388