Madhav Institute of Technology & Science, Gwalior (M.P.), INDIA

Deemed University

(Declared under Distinct Category by Ministry of Education, Government of India)

NAAC ACCREDITED WITH A++ GRADE

Skill Based Mini Project on

Brain Tumor Detection using CNN

SUBMITTED BY

Anushika Gupta
0901AD221011

Divya Pandey

0901AD221031

6th Semester

Artificial Intelligence and Data Science

SUBMITTED TO

Dr. Tej Singh Assistant Professor	**Dr. Hardev Singh Pal** Assistant Professor

CENTRE FOR ARTIFICIAL INTELLIGENCE

Madhav Institute of Technology and Science

Gwalior – 474005 (MP) est. 1957

Session 2024-2025

DECLARATION

I hereby declare that the Lab Manual for the course Image Processing (2270622) is being submitted in the partial fulfillment of the requirement for the award of Bachelor of Technology in Artificial Intelligence and Machine Learning . All the information in this document has been obtained and presented in accordance with academic rule and ethical conduct.

Date:
Place: Gwalior

Anushika Gupta
0901AD221011

Divya Pandey
0901AD221031

CERTIFICATE

This is to certify that the work contained in this project has been carried out by students mentioned below from the Centre for Artificial Intelligence. This Project was done on partial fulfillment of B.Tech laboratory "Image Processing (2270622)". It has been found to be satisfactory and hereby approved for submission.

Name of Students	Enrollment Number
Anushika Gupta	0901AD221011
Divya Pandey	0901AD221031

Dr. Tej Singh
Assistant Professor

Dr. Hardev Singh Pal
Assistant Professor

ACKNOWLEDGEMENT

I would like to express my greatest appreciation to all the individuals who have helped and supported me throughout this Skill Based Mini Project Report. I am thankful to the whole Centre for Artificial Intelligence department for their ongoing support during the experiments, from initial advice and provision of contact in the first stages through ongoing advice and encouragement, which led to the final report of this Skill Based Mini Project Report.

A special acknowledgement goes to my colleagues who helped me in completing the file and by exchanging interesting ideas to deal with problems and sharing the experience.

I wish to thank our professor Dr. Tej Singh as well for his undivided support and interests which inspired me and encouraged me to go my own way without whom I would be unable to complete my Skill Based Mini Project Report.

At the end, I want to thank my friends who displayed appreciation to my work and motivated me to continue my work.

Anushika gupta
Divya pandey

INDEX

MICRO PROJECT

S.NO	EXPERIMENT	PAGE NO.	SIGNATURE
01.	Image Preprocessing, Train-Test Splitting & Exploratory Data Analysis for Brain Tumor Detection		

MINI PROJECT

S.NO	EXPERIMENT	PAGE NO.	SIGNATURE
01.	Data Augmentation, CNN Model Development & Training for Brain Tumor Classification		

MACRO PROJECT

S.NO	EXPERIMENT	PAGE NO.	SIGNATURE
01.	Model Evaluation, Deployment & Prediction for Brain Tumor Detection		

MICRO PROJECT

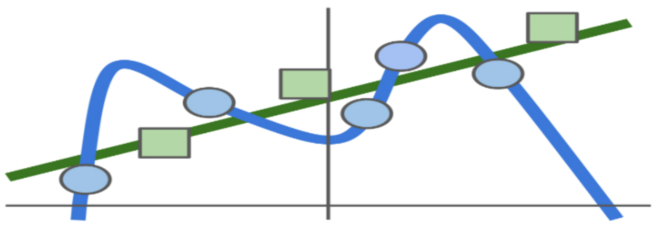
AIM : Image Preprocessing, Train-Test Splitting & Exploratory Data Analysis for Brain Tumor Detection

DATASET : Brain Tumor MRI Dataset from kaggle

THEORY : The **Brain Tumor MRI Dataset** consists of grayscale MRI images categorized into four classes: **glioma, meningioma, pituitary tumor, and no tumor**. These images vary in size and intensity, making preprocessing a crucial step to prepare the data for machine learning models. Preprocessing involves **resizing, normalization, and reshaping** to ensure consistency and improve model performance. **Resizing** standardizes all images to a fixed dimension (e.g., **128x128 pixels**) to maintain uniform input. **Normalization** scales pixel values to the **[0,1] range**, which accelerates model convergence and enhances numerical stability. Finally, **reshaping** adjusts the image dimensions to match the input format required by deep learning architectures, such as CNNs, allowing for efficient training and inference.

METHODOLOGY :

$$L(W) = \underbrace{\frac{1}{N}\sum_{i=1}^{N} L_i(f(x_i, W), y_i)}_{\substack{\textbf{Data loss}\text{: Model predictions} \\ \text{should match training data}}}$$



$$\begin{gathered}
m_t = \beta_1 * m_{t-1} + (1 - \beta_1) * g_t \\
v_t = \beta_2 * v_{t-1} + (1 - \beta_2) * g_t^2 \\
\hat{m}_t = m_t / (1 - \beta_1^t) \\
\hat{v}_t = v_t / (1 - \beta_2^t) \\
\theta = \theta - (\alpha * \hat{m}_t / \sqrt{(\hat{v}_t + \varepsilon)})
\end{gathered}$$

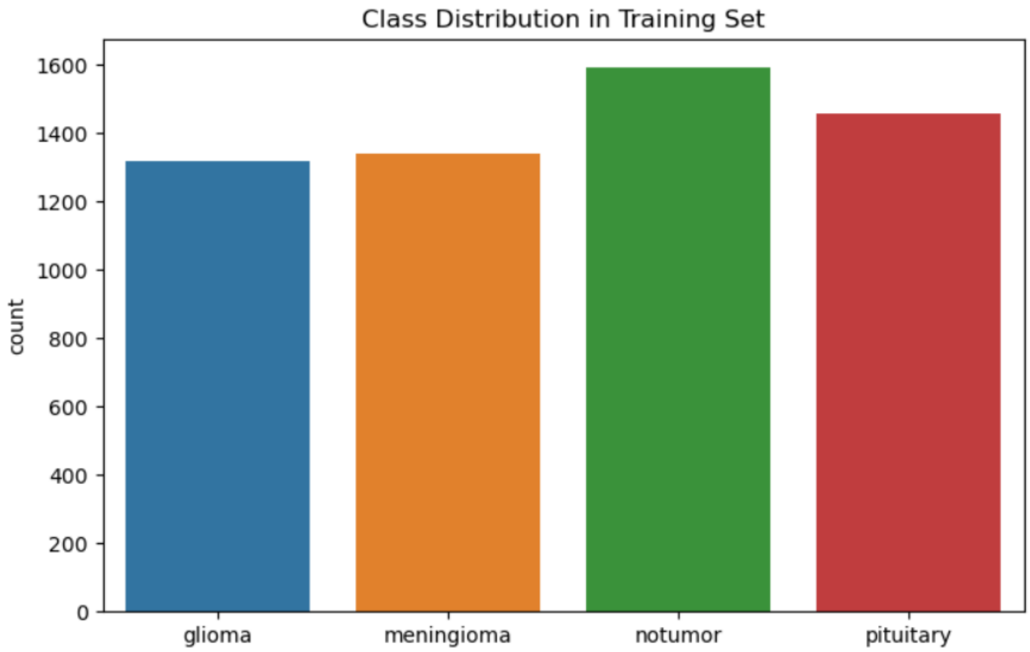
CODE **:**

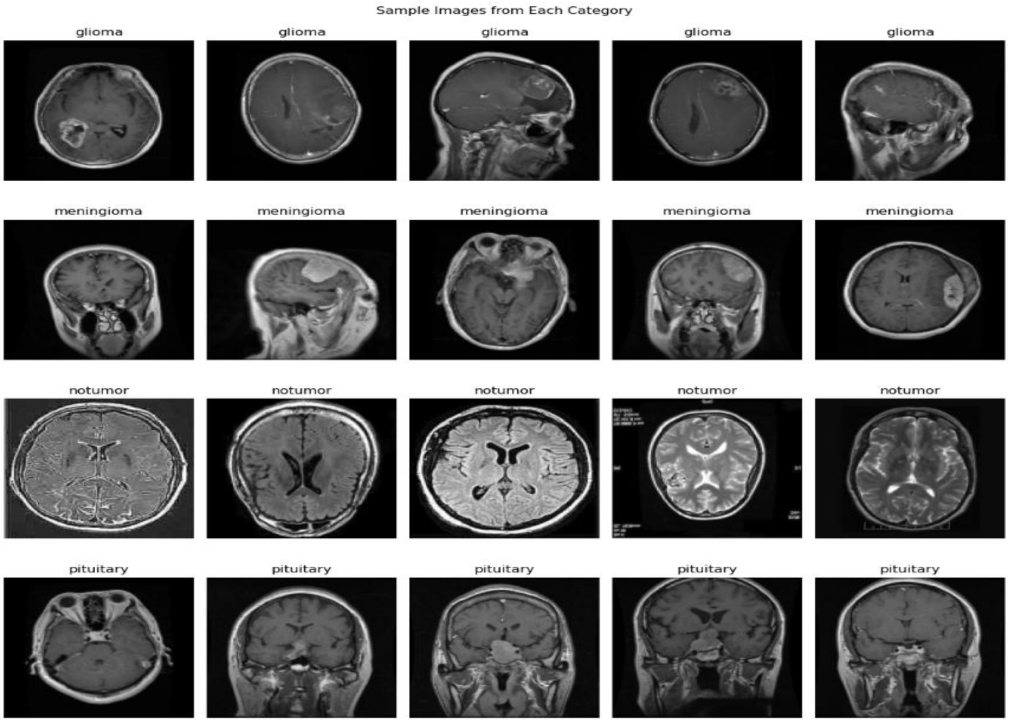
```
import numpy as np
import os
import cv2
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.utils import to_categorical
def load_images(data_dir):
    images = []
    labels = []
    for category in ['glioma', 'meningioma', 'notumor', 'pituitary']:
        folder_path = os.path.join(data_dir, category)
        for img_name in os.listdir(folder_path):
            img_path = os.path.join(folder_path, img_name)
            try:
                img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
                if img is not None:
                    img_resized = cv2.resize(img, (128, 128), interpolation=cv2.INTER_AREA)
                    images.append(img_resized)
                    labels.append(category)
            except Exception as e:
                print(f'Error loading image {img_name}: {e}')
    return np.array(images), np.array(labels)
import os
train_dir = '/Users/divyapandey/Desktop/college/Image Processing /archive (6) copy/training'
test_dir = '/Users/divyapandey/Desktop/college/Image Processing /archive (6) copy/testing'
x_train, y_train = load_images(train_dir)
x_test, y_test = load_images(test_dir)
print("Loaded training images:", x_train.shape)
print("Loaded testing images:", x_test.shape)
plt.figure(figsize=(8, 5))
sns.countplot(x=y_train)
plt.title('Class Distribution in Training Set')
plt.show()
fig, axes = plt.subplots(4, 5, figsize=(12, 12))
fig.suptitle('Sample Images from Each Category')
for i, category in enumerate(['glioma', 'meningioma', 'notumor', 'pituitary']):
    category_images = [x_train[j] for j in range(len(y_train)) if y_train[j] == category][:5]
    for j in range(len(category_images)):
        axes[i, j].imshow(category_images[j], cmap='gray')
        axes[i, j].set_title(category)
        axes[i, j].axis('off')
plt.tight_layout()
```

```
plt.show()

image_shapes = [img.shape for img in x_train]
unique_shapes = set(image_shapes)
print(f'Unique image shapes in training data: {unique_shapes}')

plt.figure(figsize=(8, 5))
flattened_images = [img.flatten() for img in x_train]
sns.histplot(np.concatenate(flattened_images), bins=50, kde=True)
plt.title('Image Intensity Distribution')
plt.xlabel('Pixel Intensity')
plt.show()
```

OUTPUT :

```
Loaded training images: (5712, 128, 128)
Loaded testing images: (1311, 128, 128)
```




Sample Images from Each Category
glioma
glioma
glioma
glioma
glioma
meningioma
meningioma
meningioma
meningioma
meningioma
notumor
notumor
notumor
notumor
notumor
pituitary
pituitary
pituitary
pituitary
pituitary

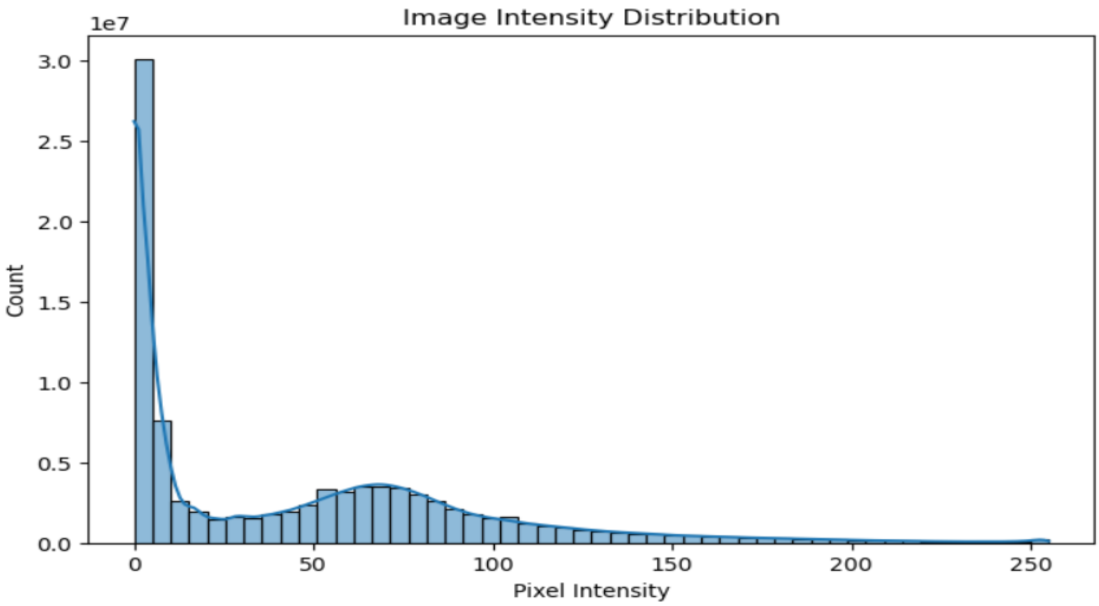


Image Intensity Distribution
1e7
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Count
0
50
100
150
200
250
Pixel Intensity

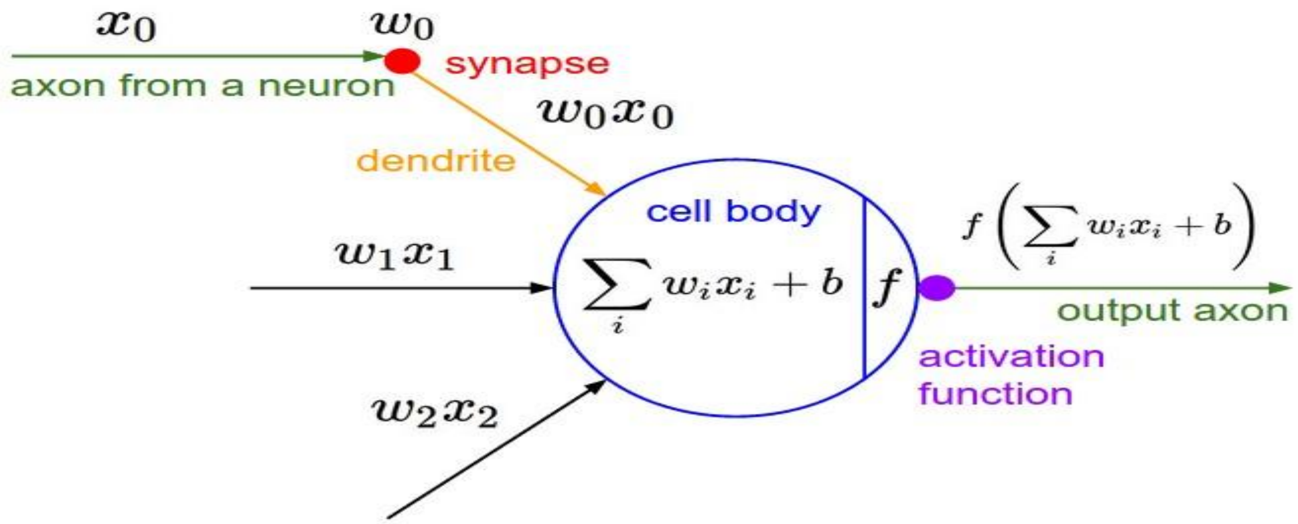
MINI PROJECT

AIM : Data Augmentation, CNN Model Development & Training for Brain Tumor Classification

DATASET : Brain Tumor MRI Dataset from kaggle

THEORY :
This project focuses on brain tumor classification using data augmentation, CNN model development, and optimized training techniques. Since medical datasets are often limited, augmentation techniques such as rotation, zooming, shifting, shearing, and flipping are applied to improve generalization. A Convolutional Neural Network (CNN) is designed with multiple convolutional layers, batch normalization, max pooling, and dropout to extract spatial features and prevent overfitting. The model is compiled using the Adam optimizer with categorical cross-entropy loss, ensuring effective learning for this multi-class classification task. To enhance training, Early Stopping prevents unnecessary iterations, while ReduceLROnPlateau dynamically adjusts the learning rate when progress slows. By combining augmented data with optimized CNN architecture, the model achieves high accuracy, making it robust for real-world medical image classification.

METHODOLOGY :



CODE :

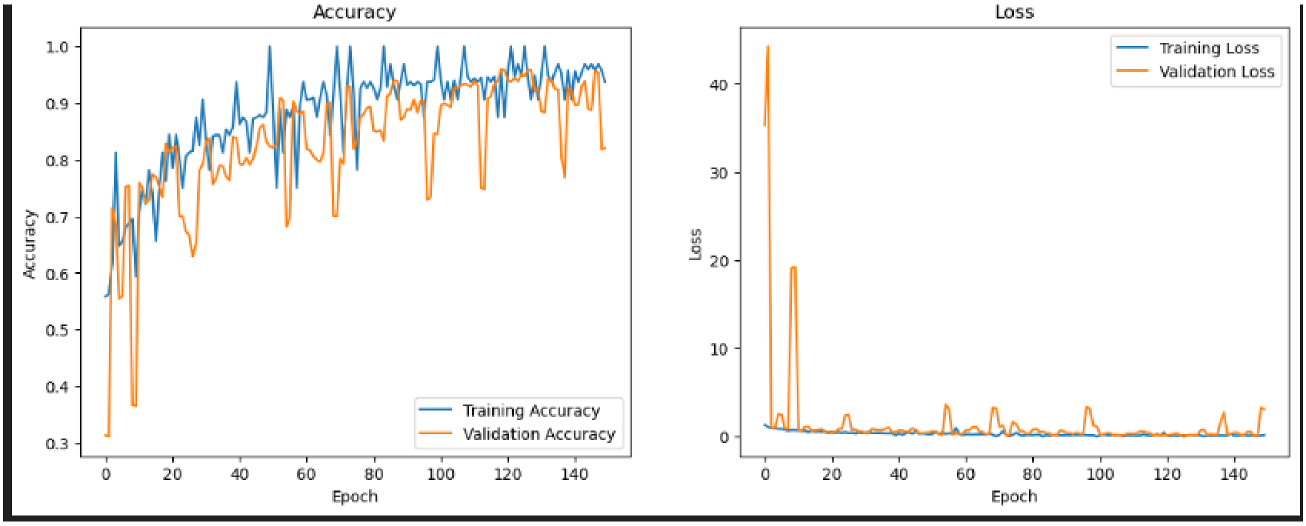
```
label_map = {'glioma': 0, 'meningioma': 1, 'notumor': 2, 'pituitary': 3}
y_train = np.array([label_map[label] for label in y_train])
y_test = np.array([label_map[label] for label in y_test])

y_train = to_categorical(y_train, num_classes=4)
y_test = to_categorical(y_test, num_classes=4)

x_train = x_train.reshape((-1, 128, 128, 1))
x_test = x_test.reshape((-1, 128, 128, 1))
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
```

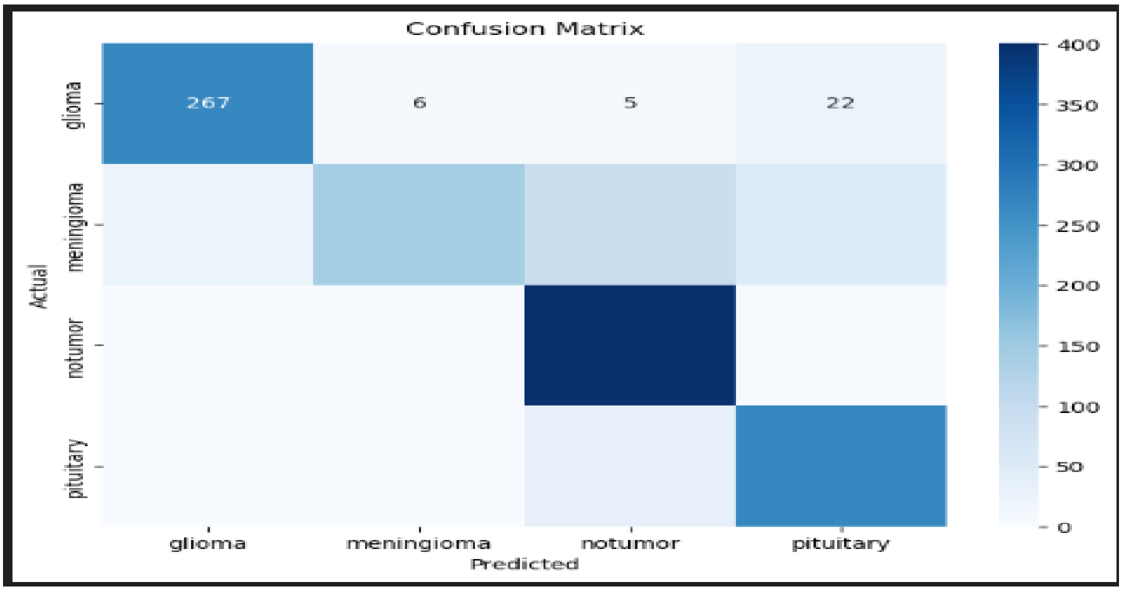
```
    MaxPooling2D((2, 2)),
    BatchNormalization(),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    BatchNormalization(),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    BatchNormalization(),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
model.compile(optimizer=Adam(learning_rate=0.001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
datagen = ImageDataGenerator(
    rotation_range=10,
    width_shift_range=0.1,
    height_shift_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True
)
history = model.fit(
    datagen.flow(x_train, y_train, batch_size=32),
    validation_data=(x_test, y_test),
    epochs=150,
    steps_per_epoch=len(x_train) // 32,
    verbose=1
)
model.save('brain_tumor_detection_model.h5')
loss, accuracy = model.evaluate(x_test, y_test, verbose=1)
print(f"Test Accuracy: {accuracy * 100:.2f}%")
def plot_training_history(history):
    plt.figure(figsize=(14, 5))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()

    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Loss')
```

```
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()
   plot_training_history(history)
   y_pred = np.argmax(model.predict(x_test), axis=1)
   y_true = np.argmax(y_test, axis=1)
 cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=label_map.keys(),
yticklabels=label_map.keys())
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
print("\nClassification Report:\n", classification_report(y_true, y_pred,
target_names=label_map.keys()))
```

OUTPUT :

```
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9062 - loss: 0.1404 - val_accuracy: 0.7689 - val_loss: 2.7461
Epoch 139/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 127ms/step - accuracy: 0.9583 - loss: 0.1269 - val_accuracy: 0.9268 - val_loss: 0.2815
Epoch 140/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 21s 121ms/step - accuracy: 0.9062 - loss: 0.3454
2025-04-03 20:13:51.877750: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9062 - loss: 0.3454 - val_accuracy: 0.9237 - val_loss: 0.3066
Epoch 141/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 128ms/step - accuracy: 0.9556 - loss: 0.1283 - val_accuracy: 0.8963 - val_loss: 0.4604
Epoch 142/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 21s 122ms/step - accuracy: 0.9375 - loss: 0.1377
2025-04-03 20:14:16.016702: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9375 - loss: 0.1377 - val_accuracy: 0.8970 - val_loss: 0.4482
Epoch 143/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 128ms/step - accuracy: 0.9547 - loss: 0.1421 - val_accuracy: 0.9291 - val_loss: 0.2395
Epoch 144/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 21s 121ms/step - accuracy: 0.9688 - loss: 0.2120
2025-04-03 20:14:40.235063: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9688 - loss: 0.2120 - val_accuracy: 0.9390 - val_loss: 0.2194
Epoch 145/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 128ms/step - accuracy: 0.9553 - loss: 0.1331 - val_accuracy: 0.8894 - val_loss: 0.5330
Epoch 146/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 23s 134ms/step - accuracy: 0.9688 - loss: 0.1478
2025-04-03 20:15:04.504049: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9688 - loss: 0.1478 - val_accuracy: 0.8871 - val_loss: 0.5351
Epoch 147/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 129ms/step - accuracy: 0.9576 - loss: 0.1337 - val_accuracy: 0.9565 - val_loss: 0.1185
Epoch 148/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 21s 122ms/step - accuracy: 0.9688 - loss: 0.1079
2025-04-03 20:15:28.908576: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9688 - loss: 0.1079 - val_accuracy: 0.9535 - val_loss: 0.1305
Epoch 149/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 23s 128ms/step - accuracy: 0.9568 - loss: 0.1345 - val_accuracy: 0.8177 - val_loss: 3.2579
Epoch 150/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 21s 123ms/step - accuracy: 0.9375 - loss: 0.2020
2025-04-03 20:15:53.153494: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 1s 7ms/step - accuracy: 0.9375 - loss: 0.2020 - val_accuracy: 0.8200 - val_loss: 3.0881
```






```
Classification Report:
              precision    recall  f1-score   support

      glioma       0.92      0.89      0.91       300
  meningioma       0.95      0.46      0.62       306
     notumor       0.76      0.99      0.86       405
   pituitary       0.78      0.89      0.83       300

    accuracy                           0.82      1311
   macro avg       0.85      0.81      0.80      1311
weighted avg       0.84      0.82      0.81      1311
```

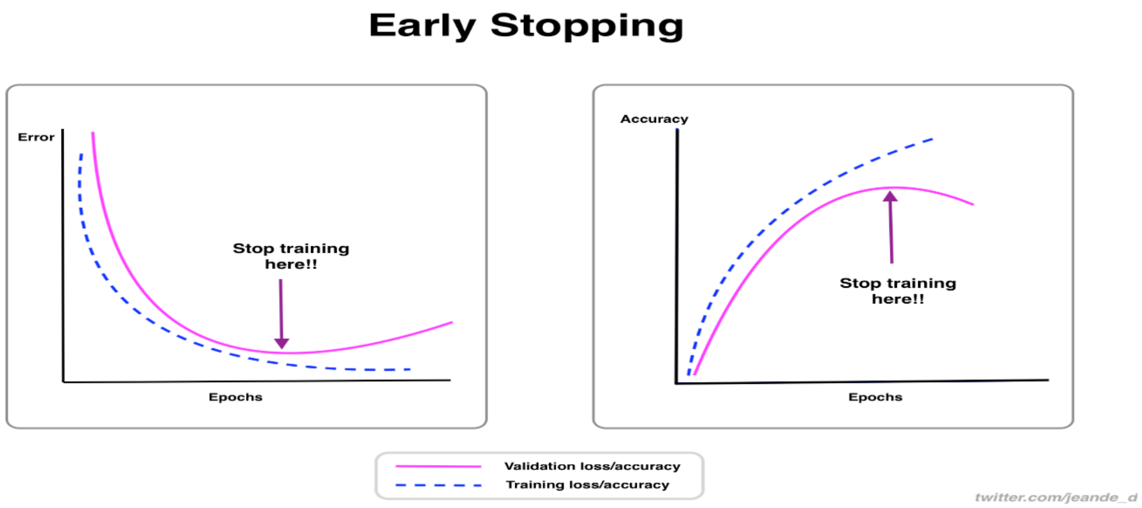
MACRO PROJECT

AIM : Model Evaluation, Deployment & Prediction for Brain Tumor Detection

DATASET : Brain Tumor MRI Dataset from kaggle

THEORY :
In the model evaluation, deployment, and prediction phase for brain tumor detection, EarlyStopping and ReduceLROnPlateau play vital roles in optimizing performance. EarlyStopping prevents overfitting by monitoring validation loss and halting training if no improvement is observed for a set number of epochs, restoring the best model weights for deployment. Meanwhile, ReduceLROnPlateau dynamically adjusts the learning rate when validation loss stagnates, lowering it to help the model escape local minima and improve convergence. These techniques ensure that the trained model generalizes well to unseen MRI scans, making it robust for real-world medical applications. By incorporating these optimization methods, the final deployed model achieves better accuracy, stability, and reliability in predicting brain tumor types.

METHODOLOGY :

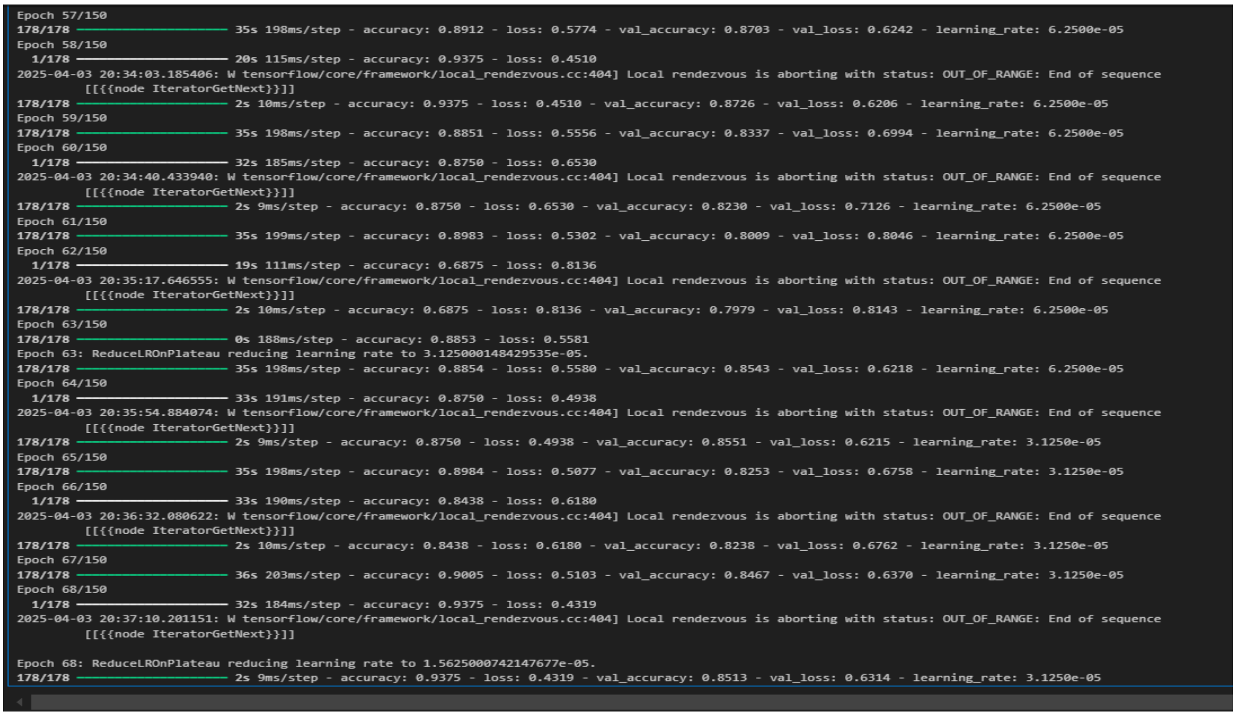


CODE :
```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1), kernel_regularizer='l2'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.2),

    Conv2D(64, (3, 3), activation='relu', kernel_regularizer='l2'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.3),

    Conv2D(128, (3, 3), activation='relu', kernel_regularizer='l2'),
```

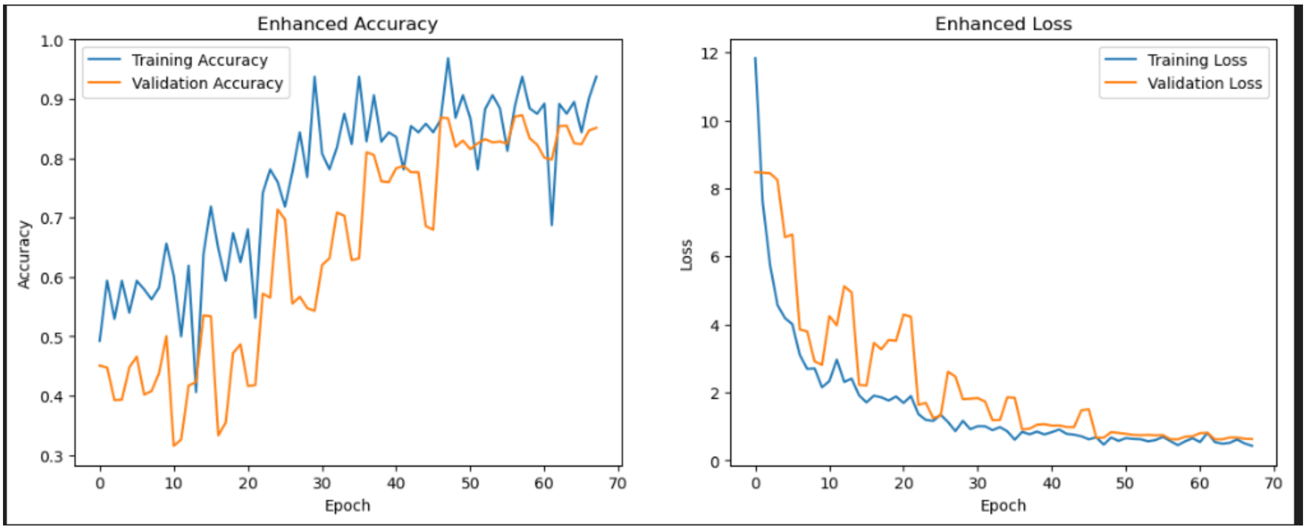
```
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.4),

    Conv2D(256, (3, 3), activation='relu', kernel_regularizer='l2'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Dropout(0.5),

    Flatten(),
    Dense(512, activation='relu', kernel_regularizer='l2'),
    Dropout(0.5),
    Dense(4, activation='softmax')
])
model.compile(optimizer=Adam(learning_rate=0.001),                    loss='categorical_crossentropy',
metrics=['accuracy'])

callbacks = [
    EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True),
    ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=5, verbose=1)
]

datagen = ImageDataGenerator(
    rotation_range=40,
    zoom_range=0.3,
    width_shift_range=0.3,
    height_shift_range=0.3,
    shear_range=0.3,
    horizontal_flip=True,
    vertical_flip=True,
    fill_mode='nearest'
)
datagen.fit(x_train)
history = model.fit(
    datagen.flow(x_train, y_train, batch_size=32),
    validation_data=(x_test, y_test),
    epochs=150,
    steps_per_epoch=len(x_train) // 32,
    callbacks=callbacks,
    verbose=1
)
def plot_training_history(history):
    plt.figure(figsize=(14, 5))
    plt.subplot(1, 2, 1)
    plt.plot(history.history['accuracy'], label='Training Accuracy')
    plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
    plt.title('Enhanced Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend()
```

```
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Enhanced Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

plot_training_history(history)
y_pred = np.argmax(model.predict(x_test), axis=1)
y_true = np.argmax(y_test, axis=1)
print("Classification Report:")
print(classification_report(y_true, y_pred, target_names=['glioma', 'meningioma', 'notumor',
'pituitary']))
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['glioma', 'meningioma', 'notumor',
'pituitary'], yticklabels=['glioma', 'meningioma', 'notumor', 'pituitary'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

OUTPUT :

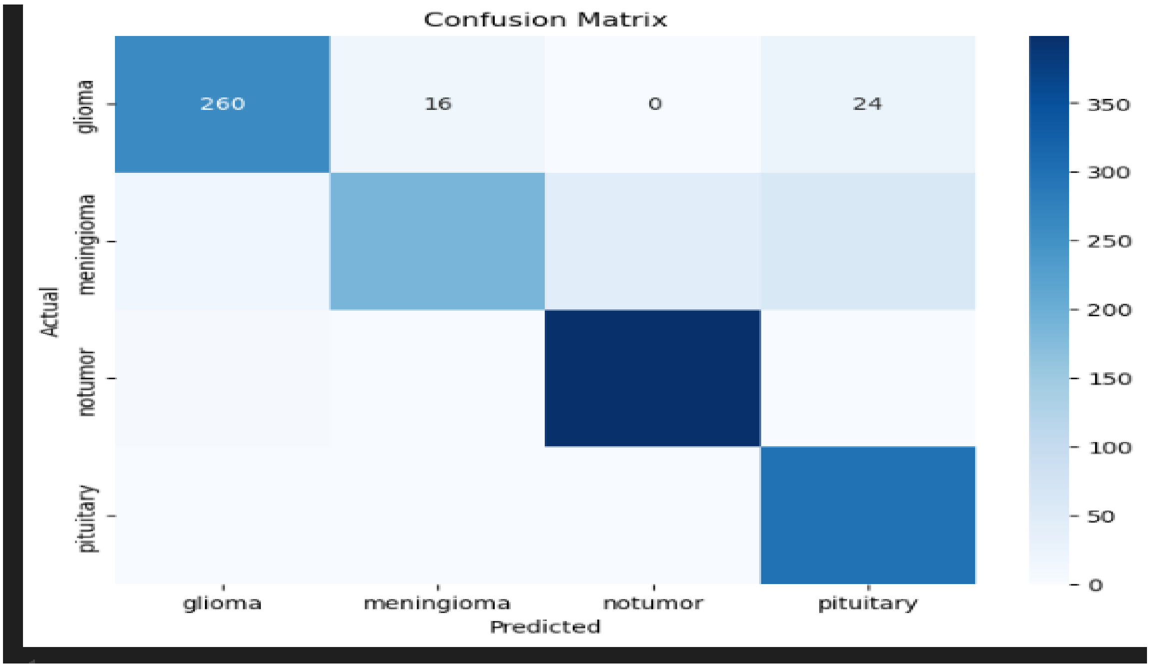


```
Epoch 57/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 35s 198ms/step - accuracy: 0.8912 - loss: 0.5774 - val_accuracy: 0.8703 - val_loss: 0.6242 - learning_rate: 6.2500e-05
Epoch 58/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 20s 115ms/step - accuracy: 0.9375 - loss: 0.4510
2025-04-03 20:34:03.185406: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.9375 - loss: 0.4510 - val_accuracy: 0.8726 - val_loss: 0.6206 - learning_rate: 6.2500e-05
Epoch 59/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 35s 198ms/step - accuracy: 0.8851 - loss: 0.5556 - val_accuracy: 0.8337 - val_loss: 0.6994 - learning_rate: 6.2500e-05
Epoch 60/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 32s 185ms/step - accuracy: 0.8750 - loss: 0.6530
2025-04-03 20:34:40.433940: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.8750 - loss: 0.6530 - val_accuracy: 0.8230 - val_loss: 0.7126 - learning_rate: 6.2500e-05
Epoch 61/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 35s 199ms/step - accuracy: 0.8983 - loss: 0.5302 - val_accuracy: 0.8009 - val_loss: 0.8046 - learning_rate: 6.2500e-05
Epoch 62/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 19s 111ms/step - accuracy: 0.6875 - loss: 0.8136
2025-04-03 20:35:17.646555: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.6875 - loss: 0.8136 - val_accuracy: 0.7979 - val_loss: 0.8143 - learning_rate: 6.2500e-05
Epoch 63/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 0s 188ms/step - accuracy: 0.8853 - loss: 0.5581
Epoch 63: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
178/178 ━━━━━━━━━━━━━━━━━━━━ 35s 198ms/step - accuracy: 0.8854 - loss: 0.5580 - val_accuracy: 0.8543 - val_loss: 0.6218 - learning_rate: 6.2500e-05
Epoch 64/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 33s 191ms/step - accuracy: 0.8750 - loss: 0.4938
2025-04-03 20:35:54.884074: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.8750 - loss: 0.4938 - val_accuracy: 0.8551 - val_loss: 0.6215 - learning_rate: 3.1250e-05
Epoch 65/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 35s 198ms/step - accuracy: 0.8984 - loss: 0.5077 - val_accuracy: 0.8253 - val_loss: 0.6758 - learning_rate: 3.1250e-05
Epoch 66/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 33s 190ms/step - accuracy: 0.8438 - loss: 0.6180
2025-04-03 20:36:32.080622: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 10ms/step - accuracy: 0.8438 - loss: 0.6180 - val_accuracy: 0.8238 - val_loss: 0.6762 - learning_rate: 3.1250e-05
Epoch 67/150
178/178 ━━━━━━━━━━━━━━━━━━━━ 36s 203ms/step - accuracy: 0.9005 - loss: 0.5103 - val_accuracy: 0.8467 - val_loss: 0.6370 - learning_rate: 3.1250e-05
Epoch 68/150
  1/178 ━━━━━━━━━━━━━━━━━━━━ 32s 184ms/step - accuracy: 0.9375 - loss: 0.4319
2025-04-03 20:37:10.201151: W tensorflow/core/framework/local_rendezvous.cc:404] Local rendezvous is aborting with status: OUT_OF_RANGE: End of sequence
	 [[{{node IteratorGetNext}}]]

Epoch 68: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
178/178 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.9375 - loss: 0.4319 - val_accuracy: 0.8513 - val_loss: 0.6314 - learning_rate: 3.1250e-05
```



Classification Report:

	precision	recall	f1-score	support
glioma	0.93	0.87	0.90	300
meningioma	0.92	0.61	0.73	306
notumor	0.90	0.99	0.94	405
pituitary	0.78	1.00	0.87	300
accuracy			0.87	1311
macro avg	0.88	0.86	0.86	1311
weighted avg	0.88	0.87	0.87	1311



REFERENCES :

[1]
A. Upreti, “Convolutional Neural Network (CNN): A comprehensive overview,” *International Journal of Multidisciplinary Research and Growth Evaluation*, pp. 488–493, Aug. 2022, doi: https://doi.org/10.54660/anfo.2022.3.4.18.
[2]
RaskuttiGarvesh, J WainwrightMartin, and YuBin, “Early stopping and non-parametric regression,” Jan. 2014, doi: https://doi.org/10.5555/2627435.2627446.
[3]
Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua, “Scheduling Techniques for Liver Segmentation: ReduceLROnPlateau vs OneCycleLR,” pp. 204–212, Jan. 2022, doi: https://doi.org/10.1007/978-3-031-08277-1_17.
[4]
M. Bayer, M.-A. Kaufhold, and C. Reuter, “A Survey on Data Augmentation for Text Classification,” *ACM Computing Surveys*, p. 3544558, Jun. 2022, doi: https://doi.org/10.1145/3544558.
[5]
Ayman Al-Kababji, Faycal Bensaali, and Sarada Prasad Dakua, “Scheduling Techniques for Liver Segmentation: ReduceLROnPlateau vs OneCycleLR,” pp. 204–212, Jan. 2022, doi: https://doi.org/10.1007/978-3-031-08277-1_17.