```
# This Python 3 environment comes with many helpful analytics libraries installed
# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files u

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of t
```

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
import random
```

```
pip install --upgrade protobuf==4.25.*
```

```
Requirement already satisfied: protobuf==4.25.* in c:\users\mahes\appdata\local\programs\p
ython\python313\lib\site-packages (4.25.8)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
```

```
#%pip install keras
import keras
```

```
#%pip install keras
import keras
from keras.models import Sequential
from keras.layers import Conv2D, Flatten, Dense, MaxPooling2D, Dropout
from sklearn.metrics import accuracy_score
```

```
#%pip install ipywidgets
#%pip install tqdm
import ipywidgets as widgets
import io
from PIL import Image
import tqdm
from sklearn.model_selection import train_test_split
import cv2
from sklearn.utils import shuffle
import tensorflow as tf
```

```
import os
import cv2
import numpy as np

X_train, Y_train = [], []
X_test, Y_test = [], []
```

```python
image_size = 150
labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']

# Base paths for training and testing
train_base_path = r"C:\Users\mahes\OneDrive - United Nations\Data Science course Naresh\3 D
test_base_path  = r"C:\Users\mahes\OneDrive - United Nations\Data Science course Naresh\3 D

# Load training data
for label in labels:
    folderPath = os.path.join(train_base_path, label)
    for filename in os.listdir(folderPath):
        img_path = os.path.join(folderPath, filename)
        img = cv2.imread(img_path)
        if img is not None:
            img = cv2.resize(img, (image_size, image_size))
            X_train.append(img)
            Y_train.append(label)

# Load testing data
for label in labels:
    folderPath = os.path.join(test_base_path, label)
    for filename in os.listdir(folderPath):
        img_path = os.path.join(folderPath, filename)
        img = cv2.imread(img_path)
        if img is not None:
            img = cv2.resize(img, (image_size, image_size))
            X_test.append(img)
            Y_test.append(label)

# Convert to numpy arrays
X_train = np.array(X_train)
Y_train = np.array(Y_train)
X_test = np.array(X_test)
Y_test = np.array(Y_test)

print(f"Training data: {X_train.shape}, Testing data: {X_test.shape}")
```

```
Training data: (2870, 150, 150, 3), Testing data: (394, 150, 150, 3)
```

```python
# ✅ Basic dataset info
print("Total images:", len(X_train))
print("Image shape:", X_train[0].shape)
print("Unique labels:", np.unique(Y_train))
```

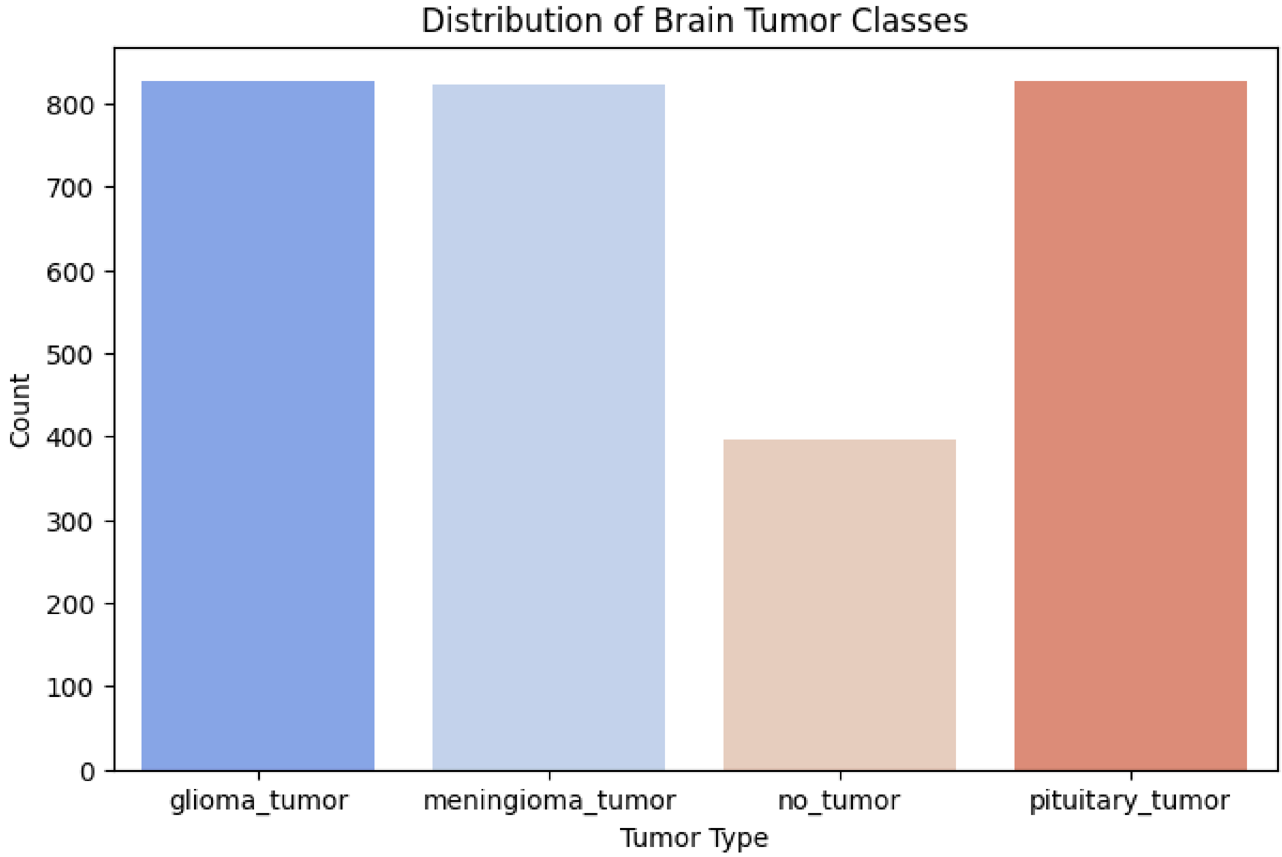
```
Total images: 2870
Image shape: (150, 150, 3)
Unique labels: ['glioma_tumor' 'meningioma_tumor' 'no_tumor' 'pituitary_tumor']
```

```python
# ✅ Convert labels to DataFrame for easy analysis
df = pd.DataFrame(Y_train, columns=['Tumor_Type'])
print("\nClass Distribution:")
print(df['Tumor_Type'].value_counts())
```

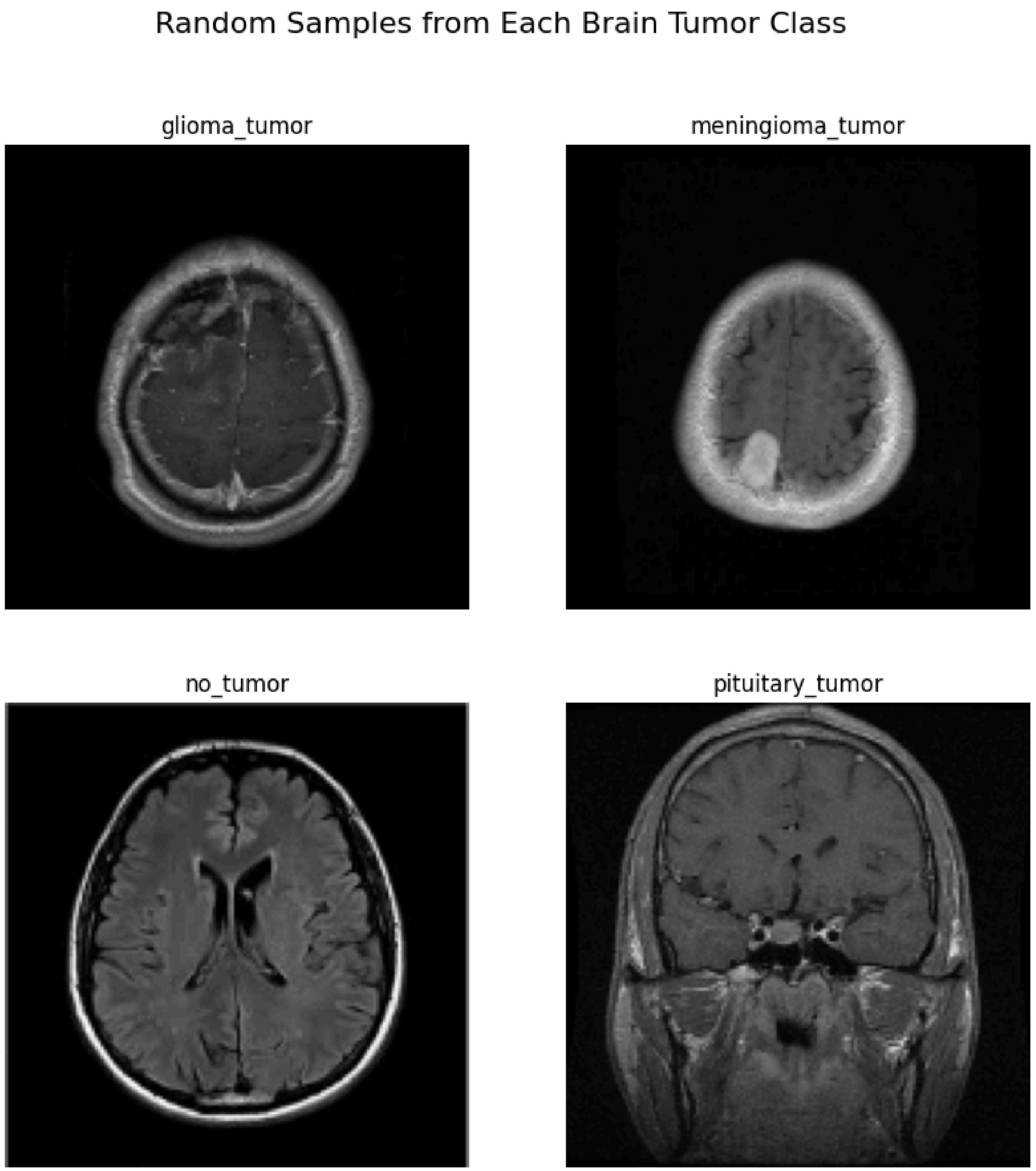
```
Class Distribution:
Tumor_Type
pituitary_tumor     827
glioma_tumor        826
meningioma_tumor    822
no_tumor            395
Name: count, dtype: int64
```

```
# ✅ Plot class distribution
plt.figure(figsize=(8,5))
sns.countplot(x='Tumor_Type', data=df, palette='coolwarm')
plt.title('Distribution of Brain Tumor Classes')
plt.xlabel('Tumor Type')
plt.ylabel('Count')
plt.show()
```

```
C:\Users\mahes\AppData\Local\Temp\ipykernel_9992\1265809927.py:3: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. As
sign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(x='Tumor_Type', data=df, palette='coolwarm')
```



```
plt.figure(figsize=(10,10))
for i, label in enumerate(labels):
    plt.subplot(2,2,i+1)
# Find the indices that match the given label
    indices = np.where(Y_train == label)[0]
    if len(indices) > 0:
        sample = random.choice(indices)
        plt.imshow(cv2.cvtColor(X_train[sample], cv2.COLOR_BGR2RGB))
        plt.title(label)
        plt.axis('off')
    else:
        plt.text(0.3, 0.5, f"No images for {label}", fontsize=12)
        plt.axis('off')
plt.suptitle("Random Samples from Each Brain Tumor Class", fontsize=16)
plt.show()
```



```
X_train,Y_train = shuffle(X_train,Y_train,random_state=0)
X_train.shape
```

```
(2870, 150, 150, 3)
```

```
X_train,X_test,y_train,y_test = train_test_split(X_train,Y_train,test_size=0.1,random_state
```

```
y_train_new = []
for i in y_train:
    y_train_new.append(labels.index(i))
y_train=y_train_new
y_train = tf.keras.utils.to_categorical(y_train)

y_test_new = []
for i in y_test:
    y_test_new.append(labels.index(i))
```

```
y_test=y_test_new
y_test = tf.keras.utils.to_categorical(y_test)
```

```
model = Sequential()
model.add(Conv2D(32,(3,3),activation = 'relu',input_shape=(150,150,3)))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Conv2D(64,(3,3),activation='relu'))
model.add(Dropout(0.3))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Conv2D(128,(3,3),activation='relu'))
model.add(Conv2D(256,(3,3),activation='relu'))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.3))
model.add(Flatten())
model.add(Dense(512,activation = 'relu'))
model.add(Dense(512,activation = 'relu'))
model.add(Dropout(0.3))
model.add(Dense(4,activation='softmax'))
```

```
c:\Users\mahes\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\layers
\convolutional\base_conv.py:113: UserWarning: Do not pass an `input_shape`/`input_dim` arg
ument to a layer. When using Sequential models, prefer using an `Input(shape)` object as t
he first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
```

```
model.summary()
```

Model: "sequential"

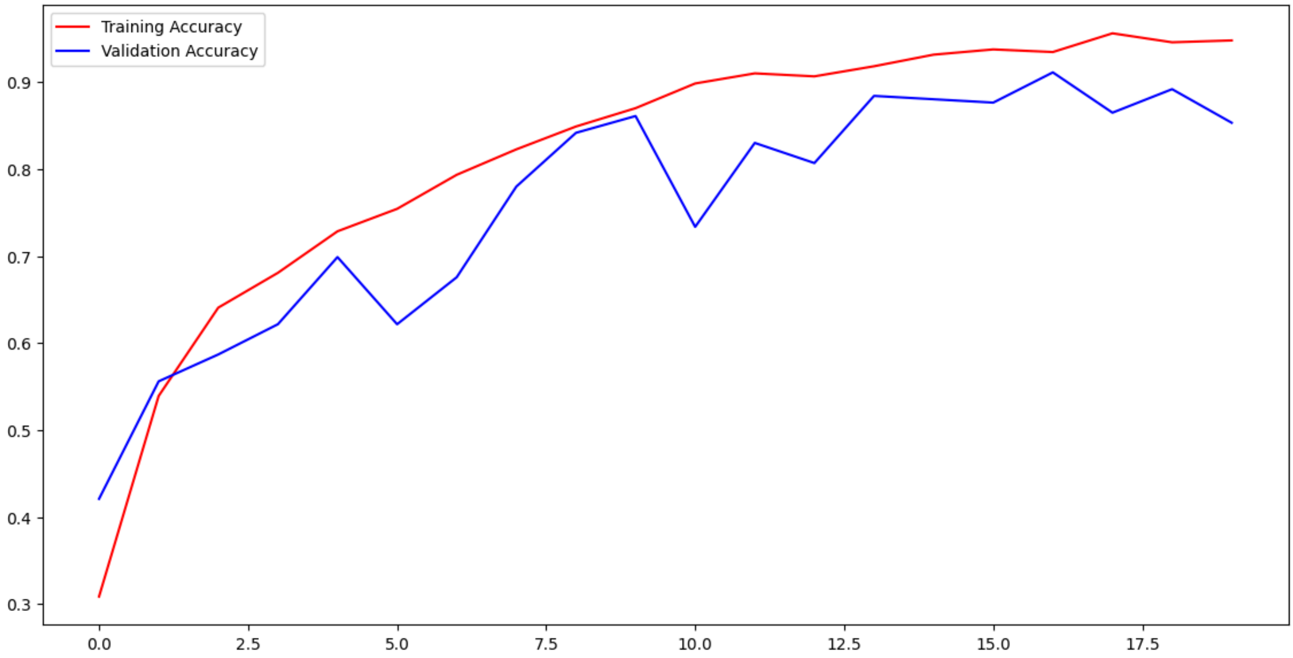
```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ conv2d (Conv2D)                 │ (None, 148, 148, 32)   │           896 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_1 (Conv2D)               │ (None, 146, 146, 64)   │        18,496 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d (MaxPooling2D)    │ (None, 73, 73, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout (Dropout)               │ (None, 73, 73, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_2 (Conv2D)               │ (None, 71, 71, 64)     │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_3 (Conv2D)               │ (None, 69, 69, 64)     │        36,928 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_1 (Dropout)             │ (None, 69, 69, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_1 (MaxPooling2D)  │ (None, 34, 34, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_2 (Dropout)             │ (None, 34, 34, 64)     │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_4 (Conv2D)               │ (None, 32, 32, 128)    │        73,856 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_5 (Conv2D)               │ (None, 30, 30, 128)    │       147,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_6 (Conv2D)               │ (None, 28, 28, 128)    │       147,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_2 (MaxPooling2D)  │ (None, 14, 14, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_3 (Dropout)             │ (None, 14, 14, 128)    │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_7 (Conv2D)               │ (None, 12, 12, 128)    │       147,584 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ conv2d_8 (Conv2D)               │ (None, 10, 10, 256)    │       295,168 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ max_pooling2d_3 (MaxPooling2D)  │ (None, 5, 5, 256)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_4 (Dropout)             │ (None, 5, 5, 256)      │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ flatten (Flatten)               │ (None, 6400)           │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense (Dense)                   │ (None, 512)            │     3,277,312 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 512)            │       262,656 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dropout_5 (Dropout)             │ (None, 512)            │             0 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_2 (Dense)                 │ (None, 4)              │         2,052 │
└─────────────────────────────────┴────────────────────────┴───────────────┘
 Total params: 4,447,044 (16.96 MB)
 Trainable params: 4,447,044 (16.96 MB)
 Non-trainable params: 0 (0.00 B)
```

```
model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
```

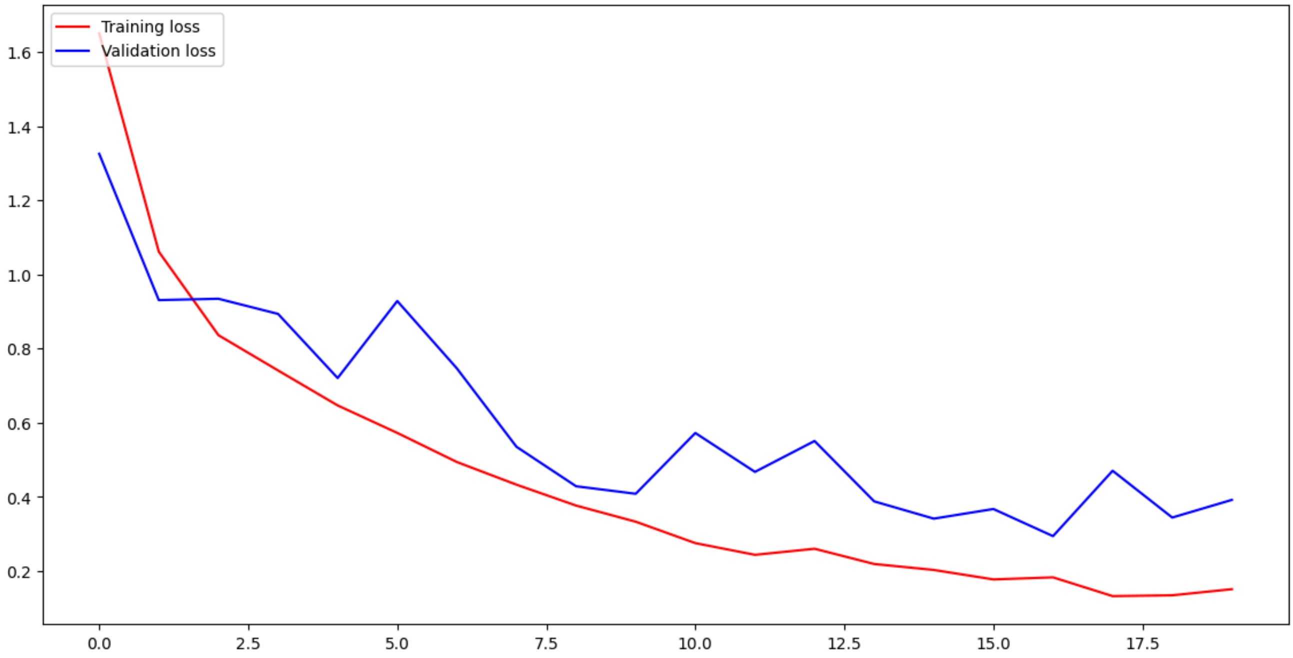
```
history = model.fit(X_train,y_train,epochs=20,validation_split=0.1)
```

```
Epoch 1/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 29s 375ms/step - accuracy: 0.3085 - loss: 1.6501 - val_accurac
y: 0.4208 - val_loss: 1.3252
Epoch 2/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 31s 423ms/step - accuracy: 0.5392 - loss: 1.0609 - val_accurac
y: 0.5560 - val_loss: 0.9304
Epoch 3/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 32s 443ms/step - accuracy: 0.6407 - loss: 0.8359 - val_accurac
y: 0.5869 - val_loss: 0.9341
Epoch 4/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 33s 454ms/step - accuracy: 0.6807 - loss: 0.7407 - val_accurac
y: 0.6216 - val_loss: 0.8933
Epoch 5/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 87s 1s/step - accuracy: 0.7285 - loss: 0.6464 - val_accuracy:
0.6988 - val_loss: 0.7203
Epoch 6/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 42s 580ms/step - accuracy: 0.7543 - loss: 0.5723 - val_accurac
y: 0.6216 - val_loss: 0.9280
Epoch 7/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 43s 592ms/step - accuracy: 0.7935 - loss: 0.4938 - val_accurac
y: 0.6757 - val_loss: 0.7463
Epoch 8/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 48s 660ms/step - accuracy: 0.8227 - loss: 0.4327 - val_accurac
y: 0.7799 - val_loss: 0.5349
Epoch 9/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 44s 601ms/step - accuracy: 0.8490 - loss: 0.3763 - val_accurac
y: 0.8417 - val_loss: 0.4284
Epoch 10/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 43s 590ms/step - accuracy: 0.8701 - loss: 0.3328 - val_accurac
y: 0.8610 - val_loss: 0.4081
Epoch 11/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 42s 570ms/step - accuracy: 0.8985 - loss: 0.2751 - val_accurac
y: 0.7336 - val_loss: 0.5721
Epoch 12/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 553ms/step - accuracy: 0.9101 - loss: 0.2435 - val_accurac
y: 0.8301 - val_loss: 0.4672
Epoch 13/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 546ms/step - accuracy: 0.9066 - loss: 0.2598 - val_accurac
y: 0.8069 - val_loss: 0.5505
Epoch 14/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 545ms/step - accuracy: 0.9182 - loss: 0.2187 - val_accurac
y: 0.8842 - val_loss: 0.3876
Epoch 15/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 547ms/step - accuracy: 0.9316 - loss: 0.2027 - val_accurac
y: 0.8803 - val_loss: 0.3411
Epoch 16/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 554ms/step - accuracy: 0.9376 - loss: 0.1771 - val_accurac
y: 0.8764 - val_loss: 0.3671
Epoch 17/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 543ms/step - accuracy: 0.9346 - loss: 0.1827 - val_accurac
y: 0.9112 - val_loss: 0.2939
Epoch 18/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 41s 563ms/step - accuracy: 0.9561 - loss: 0.1321 - val_accurac
y: 0.8649 - val_loss: 0.4702
Epoch 19/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 548ms/step - accuracy: 0.9458 - loss: 0.1343 - val_accurac
y: 0.8919 - val_loss: 0.3440
Epoch 20/20
73/73 ━━━━━━━━━━━━━━━━━━━━ 39s 539ms/step - accuracy: 0.9479 - loss: 0.1509 - val_accurac
y: 0.8533 - val_loss: 0.3917
```

```python
#model.save('braintumor.h5')

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
epochs = range(len(acc))
fig = plt.figure(figsize=(14,7))
plt.plot(epochs,acc,'r',label="Training Accuracy")
plt.plot(epochs,val_acc,'b',label="Validation Accuracy")
plt.legend(loc='upper left')
plt.show()
```



```python
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(len(loss))
fig = plt.figure(figsize=(14,7))
plt.plot(epochs,loss,'r',label="Training loss")
plt.plot(epochs,val_loss,'b',label="Validation loss")
plt.legend(loc='upper left')
plt.show()
```



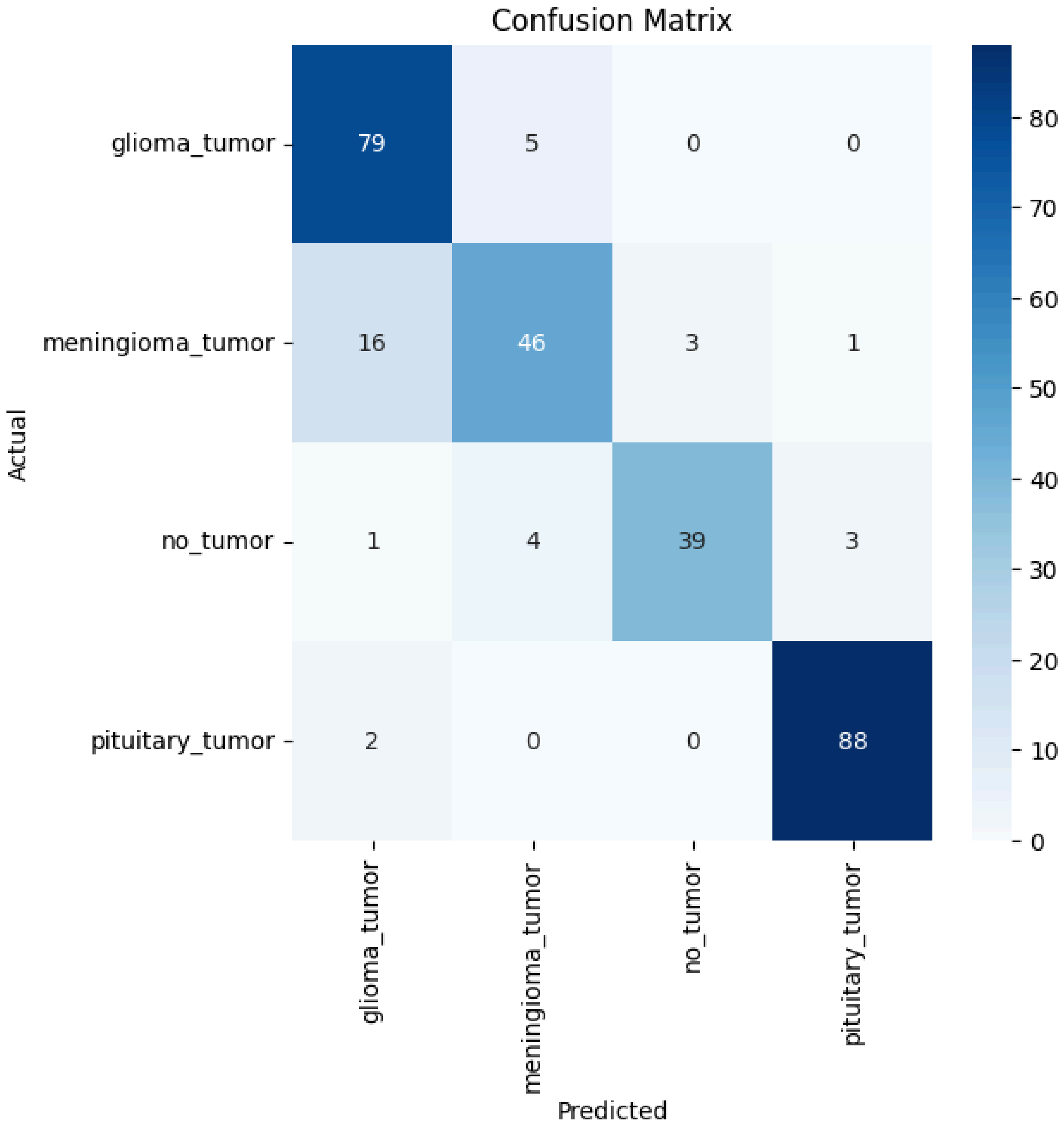
```python
test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc*100:.2f}%")
print(f"Test Loss: {test_loss:.4f}")
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 92ms/step - accuracy: 0.8780 - loss: 0.3121
Test Accuracy: 87.80%
Test Loss: 0.3121
```

```python
from sklearn.metrics import classification_report, confusion_matrix

# Predictions
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 101ms/step
```

```python
# Confusion matrix
cm = confusion_matrix(y_true, y_pred_classes)
plt.figure(figsize=(6,6))
sns.heatmap(cm, annot=True, fmt='d', xticklabels=labels, yticklabels=labels, cmap='Blues')
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()
```



```
# Classification report
print(classification_report(y_true, y_pred_classes, target_names=labels))
```

```
                  precision    recall  f1-score   support

    glioma_tumor       0.81      0.94      0.87        84
meningioma_tumor       0.84      0.70      0.76        66
        no_tumor       0.93      0.83      0.88        47
 pituitary_tumor       0.96      0.98      0.97        90

        accuracy                           0.88       287
       macro avg       0.88      0.86      0.87       287
    weighted avg       0.88      0.88      0.88       287
```

```
import cv2
import numpy as np

img = cv2.imread(r"C:\Users\mahes\OneDrive - United Nations\Data Science course Naresh\3 Da

if img is None:
    print("❌ Image not found. Check your file path.")
else:
    print("✅ Image loaded successfully.")



img = cv2.resize(img, (150, 150))
img_array = np.array(img)
img_array.shape
img_array = img_array.reshape(1, 150, 150, 3)
prediction = model.predict(img_array)
predicted_class = labels[np.argmax(prediction)]
print(f"The predicted class for the input image is: {predicted_class}")
```

```
✅ Image loaded successfully.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
The predicted class for the input image is: pituitary_tumor
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 39ms/step
The predicted class for the input image is: pituitary_tumor
```

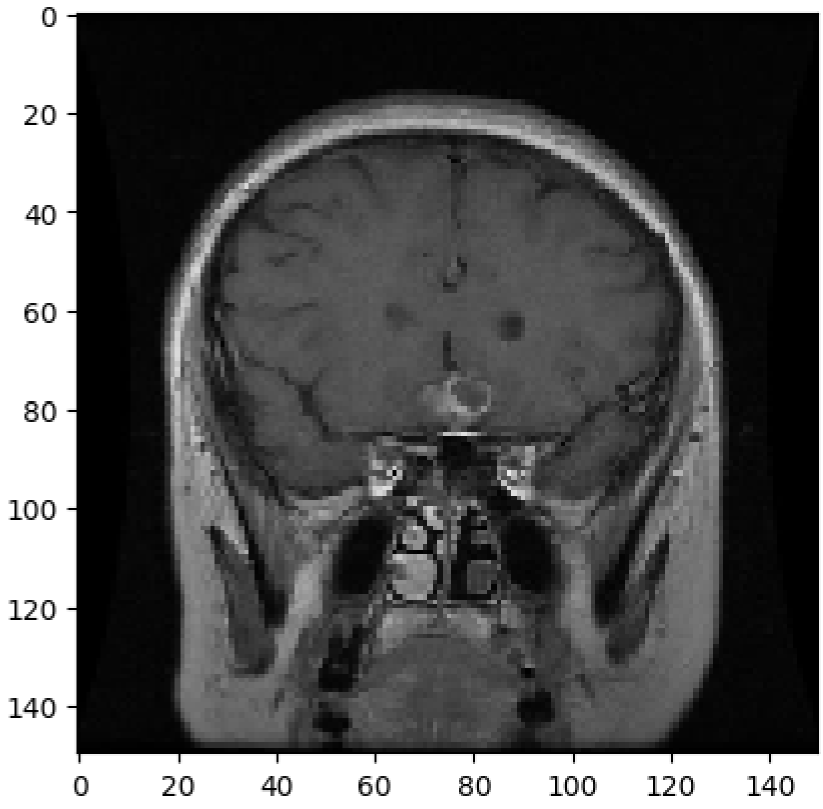
```
img_array = img_array.reshape(1,150,150,3)
img_array.shape
```

```
(1, 150, 150, 3)
```

```
import numpy as np
from tensorflow.keras.preprocessing import image

# Path to your image
img_path = (r"C:\Users\mahes\OneDrive - United Nations\Data Science course Naresh\3 Data Sc


if img is None:
    print("❌ Image not found. Check your file path.")
else:
    print("✅ Image loaded successfully.")



# Load image and resize to 150x150
img = image.load_img(img_path, target_size=(150, 150))

# Convert to numpy array
img_array = image.img_to_array(img)

# Add batch dimension (needed for prediction)
img_array = np.expand_dims(img_array, axis=0)

# Predict using your trained model
prediction = model.predict(img_array)
predicted_class = labels[np.argmax(prediction)]

print(f"The predicted class for the input image is: {predicted_class}")
```

```
✅ Image loaded successfully.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 86ms/step
The predicted class for the input image is: pituitary_tumor
```

```
from tensorflow.keras.preprocessing import image
img_path = (r"C:\Users\mahes\OneDrive - United Nations\Data Science course Naresh\3 Data Sc
plt.imshow(img,interpolation='nearest')
plt.show()
```



```
a=model.predict(img_array)
indices = a.argmax()
indices
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
```

```
np.int64(3)
```

```
# In order to improve the model accuracy further, we can consider techniques such as data a
# architectures or pre-trained models.
```

```
# Hyperparameter Tuning
"""
| Hyperparameter                   | What it does                                         | How
| -------------------------------- | ---------------------------------------------------- | ---
| **Learning rate**                | Step size for gradient descent                       | Try
| **Batch size**                   | Number of images processed before updating weights | Exp
| **Number of epochs**             | How many passes over the dataset                     | Mor
| **Optimizer**                    | How weights are updated                              | Try
| **Dropout rate**                 | Prevents overfitting                                 | Add
| **Number of layers / filters** | Model complexity                                     | Mor
"""
```

```
'\n| Hyperparameter                 | What it does                                         |
How to tune                                               |\n| ----------------------------
--- | --------------------------------------------------- | -------------------------------
-------------------------- |\n| **Learning rate**              | Step size for gradient de
scent                      | Try smaller/larger rates (e.g., 0.001, 0.0005, 0.0001)     |\n|
**Batch size**                 | Number of images processed before updating weights | Expe
riment (16, 32, 64)                                   |\n| **Number of epochs**
| How many passes over the dataset                   | Monitor validation loss to avoid ov
erfitting              |\n| **Optimizer**                  | How weights are updated
| Try `adam`, `sgd`, `rmsprop`                              |\n| **Dropout rate**
| Prevents overfitting                               | Add 0.2 - 0.5 in dense layers
|\n| **Number of layers / filters** | Model complexity                                   |
More layers/filters can improve accuracy, but can overfit |\n'
```

```
#Data Augmentation
"""
With MRI images, you often have limited data. Using augmentation artificially increases dat

Flip images horizontally/vertically

Rotate slightly (10 - 20°)

Zoom, shift, shear

Adjust brightness/contrast

Example in Keras:

        """
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    zoom_range=0.1,
    horizontal_flip=True,
    fill_mode='nearest'
)
# This helps the model generalize better, reducing overfitting and improving validation acc
```

```
# Regularization Techniques

from tensorflow.keras import regularizers
Dense(128, activation='relu', kernel_regularizer=regularizers.l2(0.001))
```

```
<Dense name=dense_3, built=False>
```

```
# Early Stopping and Learning Rate Scheduler
# Stop training when validation loss stops improving to prevent overfitting.


from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

early_stop = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.5, patience=3)
model.fit(X_train, y_train, epochs=50, validation_split=0.1, callbacks=[early_stop, reduce_
Epoch 1/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 28s 384ms/step - accuracy: 0.9574 - loss: 0.1213 - val_accurac
y: 0.8842 - val_loss: 0.4026 - learning_rate: 0.0010
Epoch 2/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 32s 433ms/step - accuracy: 0.9703 - loss: 0.0938 - val_accurac
y: 0.8185 - val_loss: 0.5745 - learning_rate: 0.0010
Epoch 3/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 32s 439ms/step - accuracy: 0.9634 - loss: 0.1029 - val_accurac
y: 0.9035 - val_loss: 0.3826 - learning_rate: 0.0010
Epoch 4/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 32s 443ms/step - accuracy: 0.9763 - loss: 0.0708 - val_accurac
y: 0.9305 - val_loss: 0.4757 - learning_rate: 0.0010
Epoch 5/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 32s 445ms/step - accuracy: 0.9686 - loss: 0.0873 - val_accurac
y: 0.8842 - val_loss: 0.3684 - learning_rate: 0.0010
Epoch 6/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 33s 454ms/step - accuracy: 0.9686 - loss: 0.0932 - val_accurac
y: 0.8533 - val_loss: 0.4579 - learning_rate: 0.0010
Epoch 7/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 33s 457ms/step - accuracy: 0.9660 - loss: 0.1035 - val_accurac
y: 0.9189 - val_loss: 0.4564 - learning_rate: 0.0010
Epoch 8/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 34s 469ms/step - accuracy: 0.9798 - loss: 0.0596 - val_accurac
y: 0.8880 - val_loss: 0.4832 - learning_rate: 0.0010
Epoch 9/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 35s 478ms/step - accuracy: 0.9793 - loss: 0.0511 - val_accurac
y: 0.8996 - val_loss: 0.3493 - learning_rate: 5.0000e-04
Epoch 10/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 36s 490ms/step - accuracy: 0.9910 - loss: 0.0292 - val_accurac
y: 0.9228 - val_loss: 0.3616 - learning_rate: 5.0000e-04
Epoch 11/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 37s 502ms/step - accuracy: 0.9884 - loss: 0.0426 - val_accurac
y: 0.9305 - val_loss: 0.3402 - learning_rate: 5.0000e-04
Epoch 12/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 37s 513ms/step - accuracy: 0.9948 - loss: 0.0155 - val_accurac
y: 0.9073 - val_loss: 0.4336 - learning_rate: 5.0000e-04
Epoch 13/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 39s 532ms/step - accuracy: 0.9927 - loss: 0.0191 - val_accurac
y: 0.9151 - val_loss: 0.3392 - learning_rate: 5.0000e-04
Epoch 14/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 39s 539ms/step - accuracy: 0.9957 - loss: 0.0142 - val_accurac
y: 0.9228 - val_loss: 0.4253 - learning_rate: 5.0000e-04
Epoch 15/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 542ms/step - accuracy: 0.9935 - loss: 0.0229 - val_accurac
y: 0.9151 - val_loss: 0.3976 - learning_rate: 5.0000e-04
Epoch 16/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 40s 546ms/step - accuracy: 0.9957 - loss: 0.0110 - val_accurac
y: 0.9073 - val_loss: 0.3912 - learning_rate: 5.0000e-04
Epoch 17/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 41s 558ms/step - accuracy: 0.9978 - loss: 0.0057 - val_accurac
y: 0.9151 - val_loss: 0.4235 - learning_rate: 2.5000e-04
Epoch 18/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 41s 562ms/step - accuracy: 0.9978 - loss: 0.0075 - val_accurac
y: 0.9189 - val_loss: 0.4572 - learning_rate: 2.5000e-04
<keras.src.callbacks.history.History at 0x1f89ea29810>
```

Understanding each metric:

Epoch 17/50 & 18/50: In the above code, 17th and 18th training cycles out of a total of 50. Each epoch means my model has seen all training images once.

accuracy: 0.9978 (≈ 99.78%) This is the model's accuracy on the training dataset. The model predicts the correct tumor class almost perfectly during training — excellent, but it might also hint at overfitting (too good on training, slightly worse on validation).

loss: 0.0057 → 0.0075 The training loss is extremely low — this means the model's predictions on training data are very close to the actual labels.

val_accuracy: 0.9151 → 0.9189 (≈ 91.9%) The accuracy on validation data, which represents new/unseen images. This is quite solid for medical image classification! A model with >90% validation accuracy is promising — especially if the data is real MRI scans.

val_loss: 0.4235 → 0.4572 The validation loss is slowly increasing — this might be an early sign of overfitting, meaning the model memorizes training data patterns but struggles with unseen data.

learning_rate: 2.5e-04 (0.00025) A relatively low learning rate — ideal for fine-tuning. It helps the model converge smoothly instead of overshooting the optimal weights.

In the next block, I am trying to improve validation accuracy further and avoid overfitting:

```python
## improve validation accuracy further and avoid overfitting:

from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau

# Stop training early if validation loss doesn't improve for 5 epochs
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True
)

# Reduce learning rate by 50% if validation loss doesn't improve for 3 epochs
reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.5,
    patience=3
)

# Train model with both callbacks
model.fit(
    X_train,
    y_train,
    epochs=50,
    validation_split=0.1,
    callbacks=[early_stop, reduce_lr]
)
```

```
Epoch 1/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 29s 397ms/step - accuracy: 0.9983 - loss: 0.0070 - val_accurac
y: 0.9035 - val_loss: 0.3844 - learning_rate: 2.5000e-04
Epoch 2/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 34s 458ms/step - accuracy: 0.9978 - loss: 0.0077 - val_accurac
y: 0.9151 - val_loss: 0.4287 - learning_rate: 2.5000e-04
Epoch 3/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 34s 468ms/step - accuracy: 0.9996 - loss: 0.0048 - val_accurac
y: 0.9228 - val_loss: 0.4069 - learning_rate: 2.5000e-04
Epoch 4/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 33s 457ms/step - accuracy: 0.9983 - loss: 0.0036 - val_accurac
y: 0.9266 - val_loss: 0.4282 - learning_rate: 2.5000e-04
Epoch 5/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 34s 460ms/step - accuracy: 0.9991 - loss: 0.0027 - val_accurac
y: 0.9305 - val_loss: 0.4330 - learning_rate: 1.2500e-04
Epoch 6/50
73/73 ━━━━━━━━━━━━━━━━━━━━ 35s 478ms/step - accuracy: 0.9983 - loss: 0.0056 - val_accurac
y: 0.9305 - val_loss: 0.4442 - learning_rate: 1.2500e-04
```

```
<keras.src.callbacks.history.History at 0x1f89ea29950>
```

What the above means:

Training accuracy: 0.9983 → this model is fitting almost perfectly to training data.

Validation accuracy: 0.9305 → That’s excellent performance on unseen data (93%).

Validation loss: 0.4442 → Acceptable; slightly higher than training loss, meaning there’s minor overfitting, but nothing alarming.

Learning rate: 1.25e-04 → The ReduceLROnPlateau callback has likely reduced the learning rate, helping fine-tune the model gradually.

Epoch: 6/50 → And EarlyStopping may stop earlier than 50 epochs once validation loss stops improving.

```python
# Let it train until early stopping triggers


test_loss, test_acc = model.evaluate(X_test, y_test)
print(f"Test Accuracy: {test_acc:.4f}, Test Loss: {test_loss:.4f}")
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 93ms/step - accuracy: 0.9338 - loss: 0.2995
Test Accuracy: 0.9338, Test Loss: 0.2995
```

Check confusion matrix & classification report to confirm per-class performance:

```python
from sklearn.metrics import classification_report, confusion_matrix
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

print(classification_report(y_true, y_pred_classes))
print(confusion_matrix(y_true, y_pred_classes))
```

```
9/9 ━━━━━━━━━━━━━━━━━━━━ 1s 95ms/step
              precision    recall  f1-score   support

           0       0.88      0.94      0.91        84
           1       0.91      0.88      0.89        66
           2       0.96      0.94      0.95        47
           3       1.00      0.97      0.98        90

    accuracy                           0.93       287
   macro avg       0.94      0.93      0.93       287
weighted avg       0.94      0.93      0.93       287

[[79  5  0  0]
 [ 6 58  2  0]
 [ 2  1 44  0]
 [ 3  0  0 87]]
```

```python
model.save("brain_tumor_classifier_v1.keras")
```

```python
from tensorflow import keras

# Load the trained model
model = keras.models.load_model("brain_tumor_classifier_v1.keras")
print("✅ Model loaded successfully!")
```

```
✅ Model loaded successfully!
```

```
c:\Users\mahes\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\saving
\saving_lib.py:797: UserWarning: Skipping variable loading for optimizer 'rmsprop', becaus
e it has 26 variables whereas the saved optimizer has 50 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
```

```python
# Predict on a new MRI image

labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']
```

```python
import numpy as np
from tensorflow.keras.preprocessing import image

# Path to the new MRI image
img_path = r"C:\path\to\your\test_image.jpg" ## Change this to your image path (replace it

# Load and preprocess the image
img = image.load_img(img_path, target_size=(150, 150))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
img_array = img_array / 255.0  # Normalize if your training data was normalized

# Predict
prediction = model.predict(img_array)
predicted_class = labels[np.argmax(prediction)]

print(f"🧠 The predicted tumor type is: {predicted_class}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 89ms/step
🧠 The predicted tumor type is: glioma_tumor
```

```python
#%pip install streamlit
import streamlit as st
import streamlit as st
import numpy as np
```

```
from tensorflow import keras
from tensorflow.keras.preprocessing import image
from PIL import Image

# Load the trained model
st.title("🧠 Brain Tumor Classification App")
st.write("Upload an MRI brain scan image to predict the tumor type.")

# Load model
@st.cache_resource
def load_model():
    model = keras.models.load_model("brain_tumor_classifier_v1.keras")
    return model

model = load_model()

# Labels
labels = ['glioma_tumor', 'meningioma_tumor', 'no_tumor', 'pituitary_tumor']

# File uploader
uploaded_file = st.file_uploader("Upload an MRI image", type=["jpg", "jpeg", "png"])

if uploaded_file is not None:
    # Display uploaded image
    image_display = Image.open(uploaded_file)
    st.image(image_display, caption="Uploaded MRI Image", use_column_width=True)

    # Preprocess the image
    img = image.load_img(uploaded_file, target_size=(150, 150))
    img_array = image.img_to_array(img)
    img_array = np.expand_dims(img_array, axis=0) / 255.0  # normalize

    # Predict
    prediction = model.predict(img_array)
    predicted_class = labels[np.argmax(prediction)]
    confidence = np.max(prediction)

    # Display results
    st.markdown(f"### 🧩 Predicted Tumor Type: **{predicted_class}**")
    st.markdown(f"### 🔹 Confidence: **{confidence:.2f}**")
```

```
Requirement already satisfied: streamlit in c:\users\mahes\appdata\local\programs\python\p
ython313\lib\site-packages (1.49.1)
Requirement already satisfied: altair!=5.4.0,!=5.4.1,<6,>=4.0 in c:\users\mahes\appdata\lo
cal\programs\python\python313\lib\site-packages (from streamlit) (5.5.0)
Requirement already satisfied: blinker<2,>=1.5.0 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (1.9.0)
Requirement already satisfied: cachetools<7,>=4.0 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (6.2.0)
Requirement already satisfied: click<9,>=7.0 in c:\users\mahes\appdata\local\programs\pyth
on\python313\lib\site-packages (from streamlit) (8.1.8)
Requirement already satisfied: numpy<3,>=1.23 in c:\users\mahes\appdata\local\programs\pyt
hon\python313\lib\site-packages (from streamlit) (2.2.6)
Requirement already satisfied: packaging<26,>=20 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (25.0)
Requirement already satisfied: pandas<3,>=1.4.0 in c:\users\mahes\appdata\local\programs\p
ython\python313\lib\site-packages (from streamlit) (2.3.3)
Requirement already satisfied: pillow<12,>=7.1.0 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (11.3.0)
Requirement already satisfied: protobuf<7,>=3.20 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (6.33.0)
Requirement already satisfied: pyarrow>=7.0 in c:\users\mahes\appdata\local\programs\pytho
n\python313\lib\site-packages (from streamlit) (21.0.0)
Requirement already satisfied: requests<3,>=2.27 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (2.32.5)
Requirement already satisfied: tenacity<10,>=8.1.0 in c:\users\mahes\appdata\local\program
s\python\python313\lib\site-packages (from streamlit) (9.1.2)
Requirement already satisfied: toml<2,>=0.10.1 in c:\users\mahes\appdata\local\programs\py
thon\python313\lib\site-packages (from streamlit) (0.10.2)
Requirement already satisfied: typing-extensions<5,>=4.4.0 in c:\users\mahes\appdata\local
\programs\python\python313\lib\site-packages (from streamlit) (4.15.0)
Requirement already satisfied: watchdog<7,>=2.1.5 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (6.0.0)
Requirement already satisfied: gitpython!=3.1.19,<4,>=3.0.7 in c:\users\mahes\appdata\loca
l\programs\python\python313\lib\site-packages (from streamlit) (3.1.45)
Requirement already satisfied: pydeck<1,>=0.8.0b4 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from streamlit) (0.9.1)
Requirement already satisfied: tornado!=6.5.0,<7,>=6.0.3 in c:\users\mahes\appdata\local\p
rograms\python\python313\lib\site-packages (from streamlit) (6.5.2)
Requirement already satisfied: jinja2 in c:\users\mahes\appdata\local\programs\python\pyth
on313\lib\site-packages (from altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (3.1.6)
Requirement already satisfied: jsonschema>=3.0 in c:\users\mahes\appdata\local\programs\py
thon\python313\lib\site-packages (from altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (4.25.1)
Requirement already satisfied: narwhals>=1.14.2 in c:\users\mahes\appdata\local\programs\p
ython\python313\lib\site-packages (from altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit) (2.3.0)
Requirement already satisfied: colorama in c:\users\mahes\appdata\local\programs\python\py
thon313\lib\site-packages (from click<9,>=7.0->streamlit) (0.4.6)
Requirement already satisfied: gitdb<5,>=4.0.1 in c:\users\mahes\appdata\local\programs\py
thon\python313\lib\site-packages (from gitpython!=3.1.19,<4,>=3.0.7->streamlit) (4.0.12)
Requirement already satisfied: smmap<6,>=3.0.1 in c:\users\mahes\appdata\local\programs\py
thon\python313\lib\site-packages (from gitdb<5,>=4.0.1->gitpython!=3.1.19,<4,>=3.0.7->stre
amlit) (5.0.2)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\mahes\appdata\local\prog
rams\python\python313\lib\site-packages (from pandas<3,>=1.4.0->streamlit) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in c:\users\mahes\appdata\local\programs\pytho
n\python313\lib\site-packages (from pandas<3,>=1.4.0->streamlit) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in c:\users\mahes\appdata\local\programs\pyt
hon\python313\lib\site-packages (from pandas<3,>=1.4.0->streamlit) (2025.2)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\mahes\appdata\local\pr
ograms\python\python313\lib\site-packages (from requests<3,>=2.27->streamlit) (3.4.3)
Requirement already satisfied: idna<4,>=2.5 in c:\users\mahes\appdata\local\programs\pytho
n\python313\lib\site-packages (from requests<3,>=2.27->streamlit) (3.10)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from requests<3,>=2.27->streamlit) (2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\mahes\appdata\local\programs
\python\python313\lib\site-packages (from requests<3,>=2.27->streamlit) (2025.8.3)
Requirement already satisfied: MarkupSafe>=2.0 in c:\users\mahes\appdata\local\programs\py
thon\python313\lib\site-packages (from jinja2->altair!=5.4.0,!=5.4.1,<6,>=4.0->streamlit)
(3.0.2)
Requirement already satisfied: attrs>=22.2.0 in c:\users\mahes\appdata\local\programs\pyth
on\python313\lib\site-packages (from jsonschema>=3.0->altair!=5.4.0,!=5.4.1,<6,>=4.0->stre
amlit) (25.3.0)
Requirement already satisfied: jsonschema-specifications>=2023.03.6 in c:\users\mahes\appd
ata\local\programs\python\python313\lib\site-packages (from jsonschema>=3.0->altair!=5.4.
0,!=5.4.1,<6,>=4.0->streamlit) (2025.4.1)
Requirement already satisfied: referencing>=0.28.4 in c:\users\mahes\appdata\local\program
s\python\python313\lib\site-packages (from jsonschema>=3.0->altair!=5.4.0,!=5.4.1,<6,>=4.0
->streamlit) (0.36.2)
Requirement already satisfied: rpds-py>=0.7.1 in c:\users\mahes\appdata\local\programs\pyt
hon\python313\lib\site-packages (from jsonschema>=3.0->altair!=5.4.0,!=5.4.1,<6,>=4.0->str
eamlit) (0.27.1)
Requirement already satisfied: six>=1.5 in c:\users\mahes\appdata\local\programs\python\py
thon313\lib\site-packages (from python-dateutil>=2.8.2->pandas<3,>=1.4.0->streamlit) (1.1
7.0)
Note: you may need to restart the kernel to use updated packages.
```

```
[notice] A new release of pip is available: 25.2 -> 25.3
[notice] To update, run: python.exe -m pip install --upgrade pip
2025-11-09 17:12:38.057 WARNING streamlit.runtime.scriptrunner_utils.script_run_context: T
hread 'MainThread': missing ScriptRunContext! This warning can be ignored when running in
bare mode.
2025-11-09 17:12:38.401
  Warning: to view this Streamlit app on a browser, run it with the following
  command:

    streamlit run C:\Users\mahes\AppData\Roaming\Python\Python313\site-packages\ipykernel_
launcher.py [ARGUMENTS]
2025-11-09 17:12:38.402 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.403 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.403 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.404 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.404 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.409 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.410 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.410 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.410 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
c:\Users\mahes\AppData\Local\Programs\Python\Python313\Lib\site-packages\keras\src\saving
\saving_lib.py:797: UserWarning: Skipping variable loading for optimizer 'rmsprop', becaus
e it has 26 variables whereas the saved optimizer has 50 variables.
  saveable.load_own_variables(weights_store.get(inner_path))
2025-11-09 17:12:38.860 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.861 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.861 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.862 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.862 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.862 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.863 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.863 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
2025-11-09 17:12:38.863 Thread 'MainThread': missing ScriptRunContext! This warning can be
ignored when running in bare mode.
```