Brain tumor CNN & DL

```
from google.colab import drive
drive.mount('/content/drive')
```

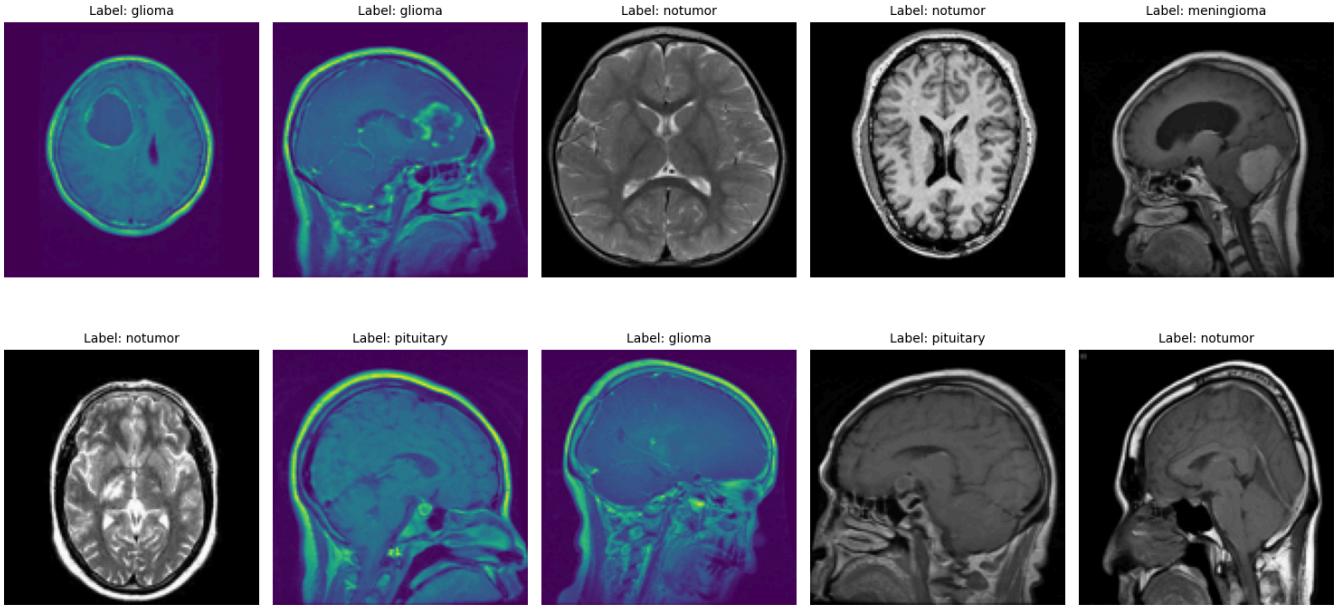
```
Mounted at /content/drive
```

```
# Install required packages
!pip install pillow tensorflow scikit-learn seaborn

# Import libraries
import os
import numpy as np
import random
from PIL import Image, ImageEnhance
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten, Dropout, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.applications import VGG16
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from sklearn.utils import shuffle
from sklearn.metrics import confusion_matrix, classification_report
import seaborn as sns

# Set random seeds for reproducibility
np.random.seed(42)
tf.random.set_seed(42)
random.seed(42)

# Data paths (update these with your actual paths)
train_dir = '/content/drive/MyDrive/MRI Image/Training'
test_dir = '/content/drive/MyDrive/MRI Image/Testing'

# Load and shuffle training data
train_paths = []
train_labels = []

for label in os.listdir(train_dir):
    for image in os.listdir(os.path.join(train_dir, label)):
        train_paths.append(os.path.join(train_dir, label, image))
        train_labels.append(label)

train_paths, train_labels = shuffle(train_paths, train_labels, random_state=42)

# Load and shuffle testing data
test_paths = []
test_labels = []

for label in os.listdir(test_dir):
    for image in os.listdir(os.path.join(test_dir, label)):
        test_paths.append(os.path.join(test_dir, label, image))
        test_labels.append(label)

test_paths, test_labels = shuffle(test_paths, test_labels, random_state=42)

# Display sample images
def show_samples(paths, labels, n_samples=10):
    plt.figure(figsize=(15, 8))
    for i in range(n_samples):
        idx = random.randint(0, len(paths)-1)
        img = Image.open(paths[idx])
        img = img.resize((128, 128))

        plt.subplot(2, 5, i+1)
        plt.imshow(img)
        plt.title(f"Label: {labels[idx]}", fontsize=10)
        plt.axis('off')
    plt.tight_layout()
    plt.show()

print("Training samples:")
show_samples(train_paths, train_labels)

print("\nTesting samples:")
show_samples(test_paths, test_labels)

# Image processing and augmentation
```

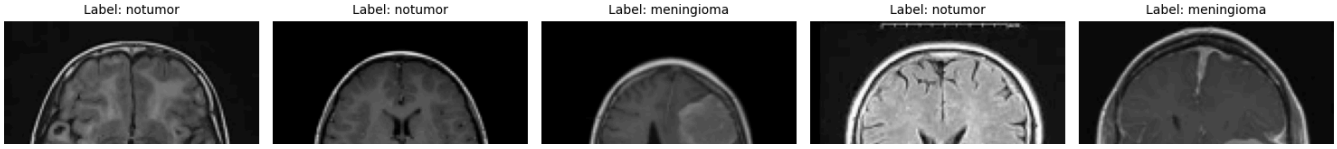
```
# Image processing and augmentation
def process_image(image_path, augment=False):
    img = load_img(image_path, target_size=(128, 128))
    img = img_to_array(img) / 255.0  # Normalize to [0,1]

    if augment:
        # Random brightness
        img = ImageEnhance.Brightness(Image.fromarray((img * 255).astype('uint8'))).enhance(random.uniform(0.8, 1.2))
        # Random contrast
        img = ImageEnhance.Contrast(img).enhance(random.uniform(0.8, 1.2))
        img = np.array(img) / 255.0

    return img

# Encode labels
class_names = sorted(os.listdir(train_dir))
label_to_idx = {name: i for i, name in enumerate(class_names)}

def encode_labels(labels):
    return np.array([label_to_idx[label] for label in labels])

# Data generator
def data_generator(paths, labels, batch_size=32, augment=False):
    while True:
        for i in range(0, len(paths), batch_size):
            batch_paths = paths[i:i+batch_size]
            batch_images = np.array([process_image(p, augment) for p in batch_paths])
            batch_labels = encode_labels(labels[i:i+batch_size])
            yield batch_images, batch_labels

# Model architecture
IMAGE_SIZE = 128
base_model = VGG16(input_shape=(IMAGE_SIZE, IMAGE_SIZE, 3), include_top=False, weights='imagenet')

# Freeze base model layers
for layer in base_model.layers:
    layer.trainable = False

# Unfreeze last few layers
for layer in base_model.layers[-4:]:
    layer.trainable = True

model = Sequential([
    base_model,
    Flatten(),
    Dropout(0.5),
    Dense(256, activation='relu'),
    Dropout(0.3),
    Dense(len(class_names), activation='softmax')
])

model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

model.summary()

# Training parameters
batch_size = 20
steps_per_epoch = len(train_paths) // batch_size
validation_steps = len(test_paths) // batch_size

# Train the model
history = model.fit(
    data_generator(train_paths, train_labels, batch_size=batch_size, augment=True),
    steps_per_epoch=steps_per_epoch,
    epochs=5,
    validation_data=data_generator(test_paths, test_labels, batch_size=batch_size),
    validation_steps=validation_steps
)

# Plot training history
plt.figure(figsize=(12, 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('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

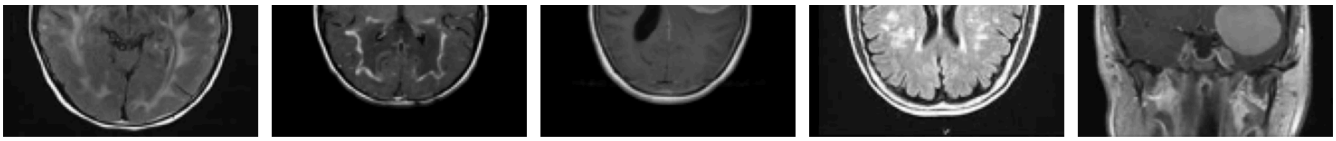
```
    plt.subplot(1, 2, 2)
    plt.plot(history.history['loss'], label='Training Loss')
    plt.plot(history.history['val_loss'], label='Validation Loss')
    plt.title('Model Loss')
    plt.xlabel('Epoch')
    plt.ylabel('Loss')
    plt.legend()
    plt.show()

    # Evaluation
    def evaluate_model(model, test_paths, test_labels):
        # Prepare test data
        X_test = np.array([process_image(p) for p in test_paths])
        y_test = encode_labels(test_labels)

        # Predictions
        y_pred = np.argmax(model.predict(X_test), axis=1)

        # Classification report
        print("\nClassification Report:")
        print(classification_report(y_test, y_pred, target_names=class_names))

        # Confusion matrix
        cm = confusion_matrix(y_test, y_pred)
        plt.figure(figsize=(8, 6))
        sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',
                    xticklabels=class_names,
                    yticklabels=class_names)
        plt.title('Confusion Matrix')
        plt.xlabel('Predicted')
        plt.ylabel('Actual')
        plt.show()

    evaluate_model(model, test_paths, test_labels)

    # Prediction function
    def predict_tumor(image_path, model):
        try:
            # Process image
            img = process_image(image_path)
            img = np.expand_dims(img, axis=0)

            # Make prediction
            preds = model.predict(img)[0]
            pred_class = np.argmax(preds)
            confidence = np.max(preds)

            # Display results
            plt.figure(figsize=(8, 6))
            plt.imshow(Image.open(image_path))
            plt.axis('off')

            result = "No tumor" if class_names[pred_class] == 'no_tumor' else f"Tumor: {class_names[pred_class].replace('_', ' ')}"
            plt.title(f"{result}\nConfidence: {confidence*100:.2f}%", fontsize=12, pad=20)
            plt.show()

            print("\nDetailed probabilities:")
            for name, prob in zip(class_names, preds):
                print(f"{name.replace('_', ' '):<20}: {prob*100:.2f}%")

        except Exception as e:
            print(f"Error: {e}")

    # Test prediction
    test_image = '/content/drive/MyDrive/MRI Image/Testing/meningioma/Te-meTr_0003.jpg'
    predict_tumor(test_image, model)

    # Save model
    model.save('brain_tumor_classifier.h5')
    print("Model saved as 'brain_tumor_classifier.h5'")
```

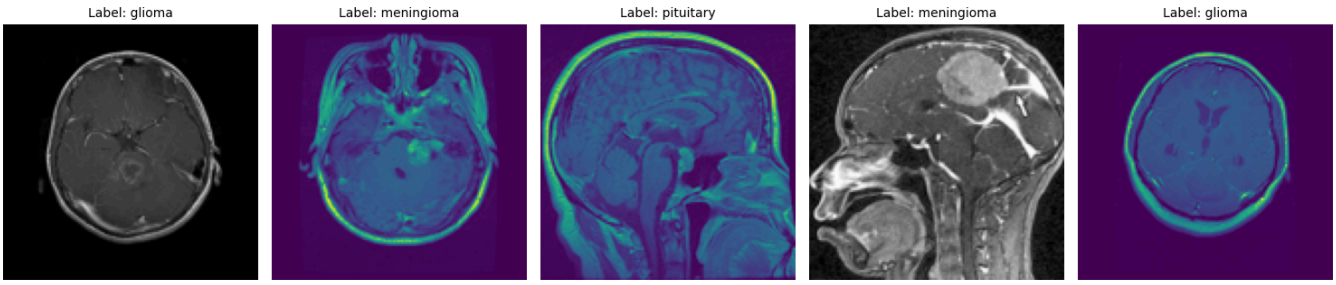
```
Requirement already satisfied: pillow in /usr/local/lib/python3.11/dist-packages (11.1.0)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.11/dist-packages (1.6.1)
Requirement already satisfied: seaborn in /usr/local/lib/python3.11/dist-packages (0.13.2)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/l
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.71.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.0.2)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.13.0)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflo
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.14.1)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn) (3.6.0)
Requirement already satisfied: pandas>=1.2 in /usr/local/lib/python3.11/dist-packages (from seaborn) (2.2.2)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /usr/local/lib/python3.11/dist-packages (from seaborn) (3.10.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorfl
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.0.8)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (0.14.1)
Requirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seabo
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seaborn)
Requirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seal
Requirement already satisfied: kiwisolver>=1.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seal
Requirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->seabo
Requirement already satisfied: python-dateutil>=2.7 in /usr/local/lib/python3.11/dist-packages (from matplotlib!=3.6.1,>=3.4->s
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.2->seaborn) (2025.1)
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->t
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensorflow)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/dist-packages (from requests<3,>=2.21.0->tensor
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tenso
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.11/dist-packages (from tensorboa
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.11/dist-packages (from tensorboard<2.19,>=2.18->tenso
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.11/dist-packages (from werkzeug>=1.0.1->tensorboard
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->tens
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras>=3.5.0->te
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras>=
Training samples:
```



Testing samples:



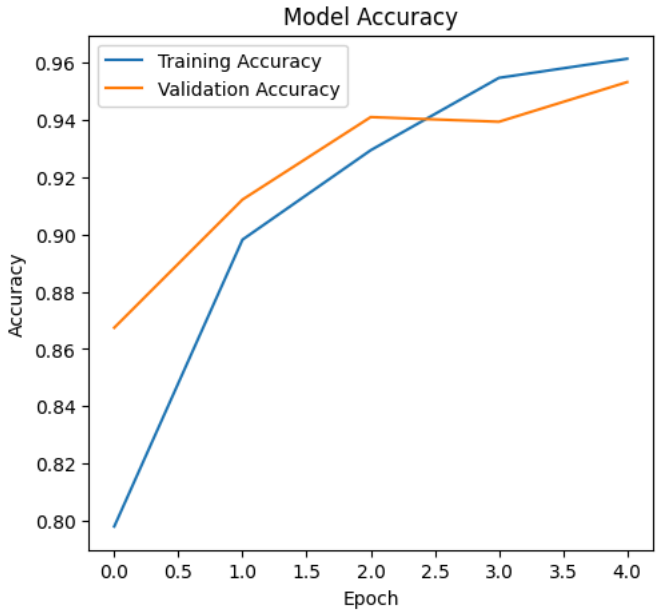


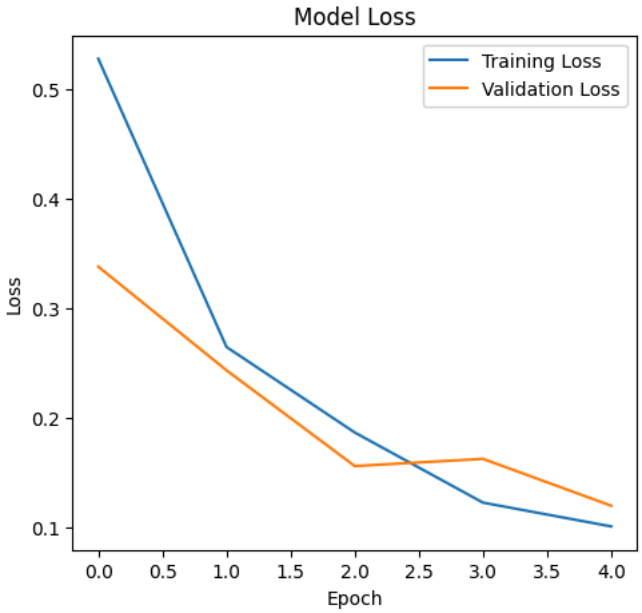



```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kern
58889256/58889256 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Model: "sequential"
```

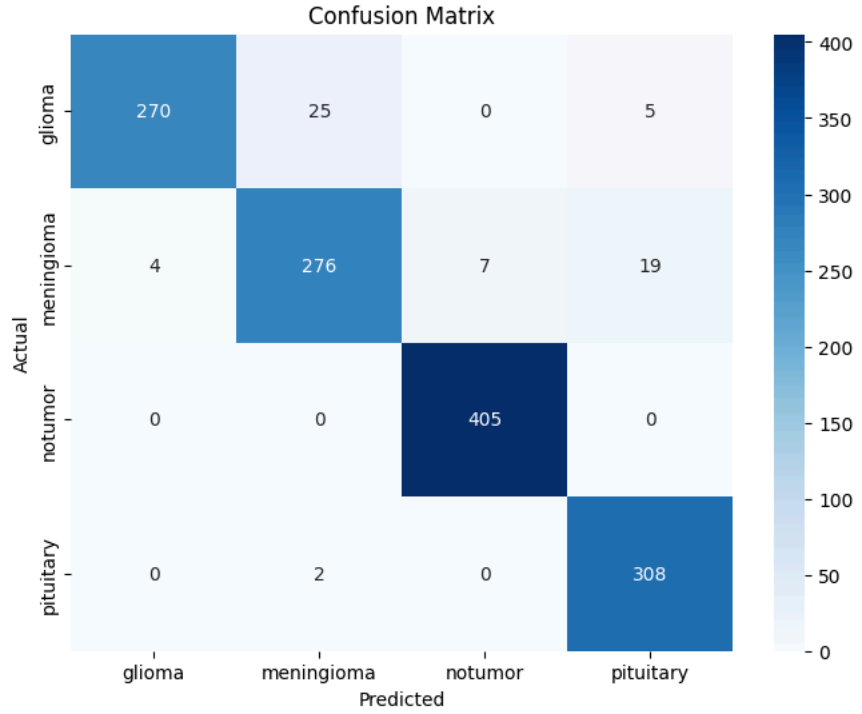
Layer (type)	Output Shape	Param #
vgg16 (Functional)	(None, 4, 4, 512)	14,714,688
flatten (Flatten)	(None, 8192)	0
dropout (Dropout)	(None, 8192)	0
dense (Dense)	(None, 256)	2,097,408
dropout_1 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 4)	1,028

```
Total params: 16,813,124 (64.14 MB)
Trainable params: 9,177,860 (35.01 MB)
Non-trainable params: 7,635,264 (29.13 MB)
Epoch 1/5
286/286 ━━━━━━━━━━━━━━━━━━━━ 2431s 8s/step - accuracy: 0.7001 - loss: 0.7673 - val_accuracy: 0.8674 - val_loss: 0.3381
Epoch 2/5
286/286 ━━━━━━━━━━━━━━━━━━━━ 85s 283ms/step - accuracy: 0.8881 - loss: 0.2927 - val_accuracy: 0.9121 - val_loss: 0.2437
Epoch 3/5
286/286 ━━━━━━━━━━━━━━━━━━━━ 41s 144ms/step - accuracy: 0.9198 - loss: 0.1989 - val_accuracy: 0.9409 - val_loss: 0.1563
Epoch 4/5
286/286 ━━━━━━━━━━━━━━━━━━━━ 43s 150ms/step - accuracy: 0.9551 - loss: 0.1224 - val_accuracy: 0.9393 - val_loss: 0.1629
Epoch 5/5
286/286 ━━━━━━━━━━━━━━━━━━━━ 37s 130ms/step - accuracy: 0.9628 - loss: 0.0956 - val_accuracy: 0.9531 - val_loss: 0.1202
```



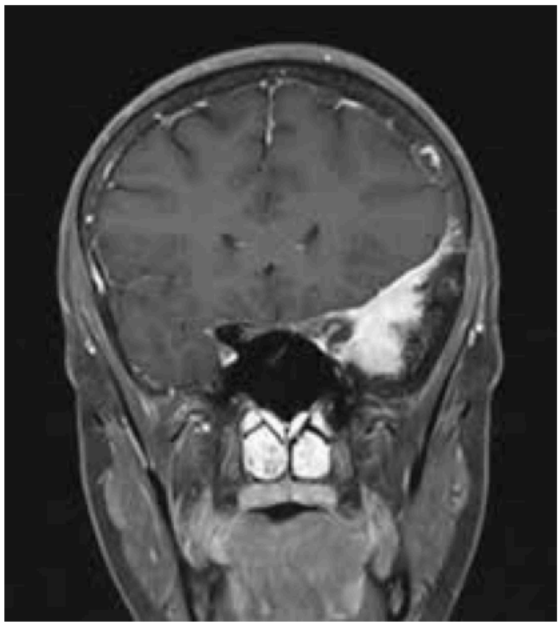



```
42/42 ━━━━━━━━━━━━━━━━━━━━ 11s 130ms/step

Classification Report:
              precision    recall  f1-score   support

      glioma       0.99      0.90      0.94       300
  meningioma       0.91      0.90      0.91       306
     notumor       0.98      1.00      0.99       405
   pituitary       0.93      0.99      0.96       310

    accuracy                           0.95      1321
   macro avg       0.95      0.95      0.95      1321
weighted avg       0.95      0.95      0.95      1321
```




```
1/1 ━━━━━━━━━━━━━━━━━━━━ 1s 648ms/step
```

Tumor: meningioma
Confidence: 99.75%



```
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format

Detailed probabilities:
glioma              : 0.04%
meningioma          : 99.75%
notumor             : 0.04%
pituitary           : 0.16%
Model saved as 'brain_tumor_classifier.h5'
```