```
# Required Libraries
import pandas as pd
import numpy as np
import os
from google.colab import files
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# For image processing
from tensorflow.keras.preprocessing.image import load_img, img_to_array

# For user interaction
import ipywidgets as widgets
from IPython.display import display, clear_output
```

```
# Step 1: Upload CSV datasets
uploaded_files = files.upload()

# Assuming the files are named 'yes_brain_tumor.csv' and 'no_brain_tumor.csv'
yes_brain_tumor_df = pd.read_csv('yes_brain_tumor.csv')
no_brain_tumor_df = pd.read_csv('no_brain_tumor.csv')

# Check the first few rows
print(yes_brain_tumor_df.head())
print(no_brain_tumor_df.head())
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving yes_brain_tumor.csv to yes_brain_tumor (6).csv
Saving no_brain_tumor.csv to no_brain_tumor (6).csv
  Filename  pixel_0  pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  \
0   Y1.jpg        3        3        3        3        3        3        2   
1   Y2.jpg        0        0        0        0        0        0        0   
2   Y3.jpg        0        0        0        0        0        0        0   
3   Y4.jpg      112       72       68       74       72       71       71   
4   Y6.jpg        1        1        1        0        1        1        1   

   pixel_7  pixel_8  ...  pixel_16374  pixel_16375  pixel_16376  pixel_16377  \
0        2        2  ...            2            2            2            2   
1        0        0  ...            0            0            0            0   
2        0        0  ...            0            0            0            0   
3       71       71  ...           29           29           29           29   
4        1        1  ...            1            1            1            1   

   pixel_16378  pixel_16379  pixel_16380  pixel_16381  pixel_16382  \
0            2            2            2            1            1   
1            0            0            0            0            0   
2            0            0            0            0            0   
3           29           29           29           27           25   
4            1            1            1            1            1   

   pixel_16383  
0            2  
1            0  
2            0  
3           85  
4            1  

[5 rows x 16385 columns]
    Filename  pixel_0  pixel_1  pixel_2  pixel_3  pixel_4  pixel_5  pixel_6  \
0  1 no.jpeg      0.0      0.0      0.0      0.0      0.0      0.0      0.0   
1  2 no.jpeg      0.0      0.0      0.0      0.0      0.0      0.0      0.0   
2   3 no.jpg      0.0      0.0      0.0      0.0      1.0      1.0      0.0   
3   4 no.jpg      2.0      2.0      2.0      2.0      2.0      2.0      2.0   
4   5 no.jpg     90.0     30.0     43.0     41.0     37.0     37.0     37.0   

   pixel_7  pixel_8  ...  pixel_16374  pixel_16375  pixel_16376  pixel_16377  \
0      0.0      0.0  ...          0.0          0.0          0.0          0.0   
1      0.0      0.0  ...          0.0          0.0          0.0          0.0   
2      2.0      7.0  ...         10.0          6.0          1.0          0.0   
3      2.0      2.0  ...          0.0          0.0          0.0          0.0   
4     37.0     37.0  ...         88.0         88.0         87.0         88.0   

   pixel_16378  pixel_16379  pixel_16380  pixel_16381  pixel_16382  \
0          0.0          0.0          0.0          0.0          0.0   
1          0.0          0.0          0.0          0.0          0.0   
2          1.0          0.0          0.0          0.0          0.0   
3          0.0          0.0          0.0          0.0          0.0   
4         88.0         86.0         85.0         86.0         86.0   

   pixel_16383  
0          0.0  
1          0.0  
2          0.0  
3          0.0  
4        141.0  

[5 rows x 16385 columns]
```

```
# Function to prepare images and labels from the dataset
def prepare_images_and_labels(dataframe, label):
    images = []
    labels = []
    for index, row in dataframe.iterrows():
        pixels = row[1:].values.astype('float32')  # All pixel values
        pixels = pixels.reshape(128, 128)  # Reshape to 128x128 image
        images.append(pixels)
        labels.append(label)
    return np.array(images), np.array(labels)

# Preparing images and labels for both datasets
yes_images, yes_labels = prepare_images_and_labels(yes_brain_tumor_df, 1)  # 1 for tumor
no_images, no_labels = prepare_images_and_labels(no_brain_tumor_df, 0)  # 0 for no tumor

# Combine the datasets
X = np.concatenate((yes_images, no_images), axis=0)
y = np.concatenate((yes_labels, no_labels), axis=0)

# Reshape images to include channel dimension (128, 128, 1)
X = X.reshape(X.shape[0], 128, 128, 1)

# Normalize pixel values (0-255) to range [0, 1]
X = X / 255.0



# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)



# Step 4: Build the CNN model
def create_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 1)),
        MaxPooling2D(pool_size=(2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D(pool_size=(2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dropout(0.5),
        Dense(1, activation='sigmoid')  # Binary classification
    ])
    model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
    return model

# Step 5: Train the model
model = create_model()
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_data=(X_test, y_test))
```

```
Epoch 1/10
/usr/local/lib/python3.10/dist-packages/keras/src/layers/convolutional/base_conv.py:107: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, pr
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
7/7 ━━━━━━━━━━━━━━━━━━━━ 8s 976ms/step - accuracy: 0.5833 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
```

```
Epoch 2/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 8s 593ms/step - accuracy: 0.4081 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 3/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 7s 929ms/step - accuracy: 0.4151 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 4/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 8s 594ms/step - accuracy: 0.4100 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 5/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 7s 924ms/step - accuracy: 0.4299 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 6/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 594ms/step - accuracy: 0.3901 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 7/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 4s 597ms/step - accuracy: 0.4214 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 8/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 6s 930ms/step - accuracy: 0.4309 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 9/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 8s 596ms/step - accuracy: 0.4037 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
Epoch 10/10
7/7 ━━━━━━━━━━━━━━━━━━━━ 6s 886ms/step - accuracy: 0.4421 - loss: nan - val_accuracy: 0.3269 - val_loss: nan
```

Start coding or generate with AI.

```
# Step 6: Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test)
print(f"Model Accuracy: {accuracy * 100:.2f}%")

# Save the trained model
model.save('brain_tumor_model.h5')

# Save the trained data
yes_brain_tumor_df.to_csv('yes_brain_tumor.csv', index=False)
no_brain_tumor_df.to_csv('no_brain_tumor.csv', index=False)
```

```
2/2 ━━━━━━━━━━━━━━━━━━━━ 0s 112ms/step - accuracy: 0.3325 - loss: nan
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is considered legacy. We recommend using instead the native Keras fo
Model Accuracy: 32.69%
```

```
import numpy as np
import pandas as pd
from keras.preprocessing.image import load_img, img_to_array
from google.colab import files
import ipywidgets as widgets
from IPython.display import display

# Load existing tumor and non-tumor datasets
try:
    yes_brain_tumor_df = pd.read_csv('yes_brain_tumor.csv')
except FileNotFoundError:
    yes_brain_tumor_df = pd.DataFrame(columns=['Filename'])

try:
    no_brain_tumor_df = pd.read_csv('no_brain_tumor.csv')
except FileNotFoundError:
    no_brain_tumor_df = pd.DataFrame(columns=['Filename'])
```

```
# Function to classify a new image
def classify_new_image(image_path):
    # Load and preprocess the new image
    image = load_img(image_path, target_size=(128, 128), color_mode='grayscale')
    image = img_to_array(image) / 255.0  # Normalize
    image = np.expand_dims(image, axis=0)  # Add batch dimension

    # Classify the image (assuming model is already loaded)
    prediction = model.predict(image)
    if prediction[0][0] > 0.5:
        print("Result: Tumor detected")
        return "tumor"
    else:
        print("Result: No tumor detected")
        return "no_tumor"

# Function to handle the image classification and addition to CSV files
def process_image():
    uploaded_test_image = files.upload()
    test_image_path = next(iter(uploaded_test_image))  # Get the filename

    # Classify the new image
    result = classify_new_image(test_image_path)

    # Add the new image to the respective CSV file
    new_row = pd.DataFrame([{'Filename': test_image_path}])  # Create a DataFrame with the new row

    if result == "tumor":
        # Save the tumor image if detected
        print("Saving the tumor image...")
        with open(f"tumor_{test_image_path}", "wb") as f:
            f.write(uploaded_test_image[test_image_path])

        # Add the new row to the yes_brain_tumor_df and save it
        global yes_brain_tumor_df  # Make it global to modify the original dataframe
        yes_brain_tumor_df = pd.concat([yes_brain_tumor_df, new_row], ignore_index=True)
        yes_brain_tumor_df.to_csv('yes_brain_tumor.csv', index=False)
    else:
        # Add the new row to the no_brain_tumor_df and save it
        global no_brain_tumor_df  # Make it global to modify the original dataframe
        no_brain_tumor_df = pd.concat([no_brain_tumor_df, new_row], ignore_index=True)
        no_brain_tumor_df.to_csv('no_brain_tumor.csv', index=False)

    print(f"The uploaded image has been classified as '{result}' and added to the respective CSV file.")

# Stop button callback function
def stop_upload(change):
    global continue_upload
    continue_upload = False
    print("Upload process stopped.")

# Function to repeatedly ask for new image uploads
def continuous_upload():
    global continue_upload
    continue_upload = True

    # Create a stop button
    # Create a stop button
    stop_button = widgets.Button(description="Stop Upload")
    stop_button.on_click(stop_upload)
    display(stop_button)

    # Keep uploading and classifying images until stop button is clicked
    while continue_upload:
        process_image()

# Start the continuous image classification
continuous_upload()
```

```
Stop Upload
Choose Files  No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Tr-no_1264.jpg to Tr-no_1264.jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 29ms/step
Result: No tumor detected
The uploaded image has been classified as 'no_tumor' and added to the respective CSV file.
Choose Files  No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving Tr-no_1273.jpg to Tr-no_1273 (2).jpg
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step
Result: No tumor detected
The uploaded image has been classified as 'no_tumor' and added to the respective CSV file.
Choose Files  No file chosen          Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
```

```
# Function to stop uploading and download the trained dataset
def stop_and_download():
    print("Stopping the upload process...")
    files.download('yes_brain_tumor.csv')
    files.download('no_brain_tumor.csv')

# Create a button to stop uploading and download datasets
stop_button = widgets.Button(description="Stop Upload and Download Data")
stop_button.on_click(lambda x: stop_and_download())

display(stop_button)
```

```
Stop Upload and D…
```

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.