```
!pip install tensorflow opencv-python-headless
```

```
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.15.0)
Requirement already satisfied: opencv-python-headless in /usr/local/lib/python3.10/dist-packages (4.10.0.84)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.3.25)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.9.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: ml-dtypes~=0.2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<2.0.0,>=1.23.5 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (24.1)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/pytho
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.37.
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.64.1)
Requirement already satisfied: tensorboard<2.16,>=2.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.2)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: keras<2.16,>=2.15.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.15.0)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (0.43
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorfl
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->t
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.16,>=2.15->tensorflow) (3
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboa
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboar
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.16,>
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<2,>=0.5->
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboa
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.16,>=2.
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.1
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.16,>=2
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-o
```

```
from google.colab import files

uploaded = files.upload()

# The uploaded file is now in the uploaded dictionary
# Let's assume you uploaded a file named 'example.zip'
zip_file_name = 'example.zip'
```

Choose Files newarchive.zip

- **newarchive.zip**(application/x-zip-compressed) - 52923344 bytes, last modified: 7/9/2024 - 100% done

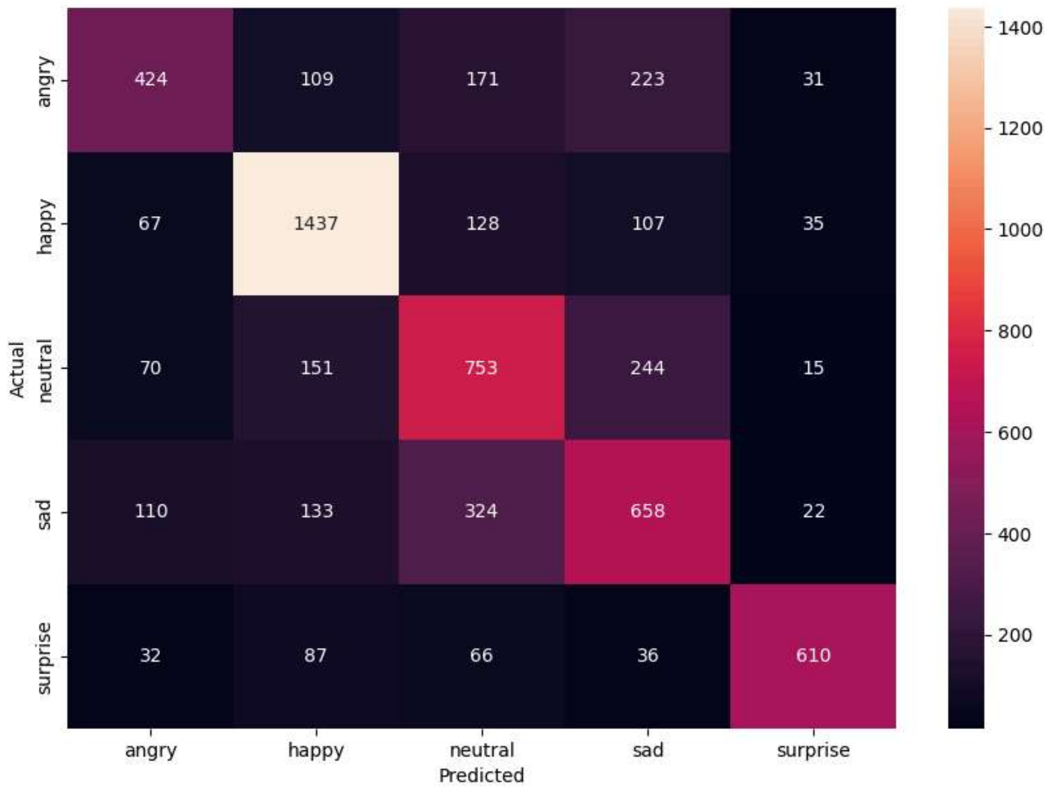
```
Saving newarchive.zip to newarchive.zip
```

```
import zipfile
import os

# Specify the name of the zip file
zip_file_name = 'newarchive.zip'

# Create a directory to extract the contents
extract_dir = 'extracted_files'
os.makedirs(extract_dir, exist_ok=True)

# Extract the contents
with zipfile.ZipFile(zip_file_name, 'r') as zip_ref:
    zip_ref.extractall(extract_dir)

print(f'Contents of the zip file extracted to: {extract_dir}')
```

```
Contents of the zip file extracted to: extracted_files
```

```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import numpy as np
import cv2
import zipfile
import os

# Path to the ZIP file
zip_file_path = '/content/drive/MyDrive/graph_projcet/archive.zip'
extracted_folder_path = '/content/extracted_files'

# Extract the ZIP file
#with zipfile.ZipFile(zip_file_path, 'r') as zip_ref:
#    zip_ref.extractall(extracted_folder_path)

# Paths to the extracted train and test directories
train_dataset_path = os.path.join(extracted_folder_path, 'train')
test_dataset_path = os.path.join(extracted_folder_path, 'test')

image_size = (48, 48)
batch_size = 32

# Data augmentation and rescaling
datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2
)

# Load training data
train_generator = datagen.flow_from_directory(
    train_dataset_path,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='training'
)

# Load validation data
validation_generator = datagen.flow_from_directory(
    train_dataset_path,
    target_size=image_size,
    batch_size=batch_size,
    class_mode='categorical',
    subset='validation'
)

# Define the model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_size[0], image_size[1], 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(len(train_generator.class_indices), activation='softmax')
])

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(train_generator, validation_data=validation_generator, epochs=15)

# Save the model
model.save('emotion_model.h5')

# Load the trained model
model = load_model('emotion_model.h5')

# Data generator for test data
test_datagen = ImageDataGenerator(rescale=1./255)

# Load test data
test_generator = test_datagen.flow_from_directory(
        test_dataset_path,
        target_size=image_size,
        batch_size=batch_size,
        class_mode='categorical',
        shuffle=False
    )

    # Evaluate the model
    test_loss, test_accuracy = model.evaluate(test_generator)
    print(f'Test Loss: {test_loss}')
    print(f'Test Accuracy: {test_accuracy}')

    # Function to preprocess image for prediction
    def preprocess_image(image):
        image = cv2.resize(image, (48, 48))
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        image = image.astype('float32') / 255
        image = np.expand_dims(image, axis=0)
        return image

    # Predict emotion from image
    def predict_emotion(image):
        preprocessed_image = preprocess_image(image)
        prediction = model.predict(preprocessed_image)
        return class_labels[np.argmax(prediction)]

    # Load class labels
    class_labels = list(test_generator.class_indices.keys())

    # Predict emotions for all images in the test set
    def predict_test_set(test_generator):
        predictions = model.predict(test_generator)
        predicted_classes = np.argmax(predictions, axis=1)
        true_classes = test_generator.classes
        class_labels = list(test_generator.class_indices.keys())
        return predicted_classes, true_classes, class_labels

    # Get predictions
    predicted_classes, true_classes, class_labels = predict_test_set(test_generator)

    # Print some results
    import pandas as pd
    results_df = pd.DataFrame({
        'Filename': test_generator.filenames,
        'Predicted': [class_labels[i] for i in predicted_classes],
        'True': [class_labels[i] for i in true_classes]
    })

    # Display some prediction results
    print(results_df.head())

    # Calculate and print confusion matrix
    from sklearn.metrics import confusion_matrix, classification_report
    import seaborn as sns
    import matplotlib.pyplot as plt

    conf_matrix = confusion_matrix(true_classes, predicted_classes)
    plt.figure(figsize=(10, 7))
    sns.heatmap(conf_matrix, annot=True, fmt="d", xticklabels=class_labels, yticklabels=class_labels)
    plt.ylabel('Actual')
    plt.xlabel('Predicted')
    plt.show()

    # Print classification report
    print(classification_report(true_classes, predicted_classes, target_names=class_labels))
```

```
:ound 19341 images belonging to 5 classes.
:ound 4835 images belonging to 5 classes.
:poch 1/15
;05/605 [==============================] - 77s 125ms/step - loss: 1.4731 - accuracy: 0.3557 - val_loss: 1.2942 - val_accuracy: 0.4602
:poch 2/15
;05/605 [==============================] - 74s 122ms/step - loss: 1.2173 - accuracy: 0.5017 - val_loss: 1.1098 - val_accuracy: 0.5574
:poch 3/15
;05/605 [==============================] - 63s 104ms/step - loss: 1.0827 - accuracy: 0.5663 - val_loss: 1.0400 - val_accuracy: 0.5944
:poch 4/15
;05/605 [==============================] - 64s 105ms/step - loss: 1.0009 - accuracy: 0.6056 - val_loss: 0.9752 - val_accuracy: 0.6199
:poch 5/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.9363 - accuracy: 0.6331 - val_loss: 0.9586 - val_accuracy: 0.6269
:poch 6/15
;05/605 [==============================] - 64s 107ms/step - loss: 0.8708 - accuracy: 0.6584 - val_loss: 0.9315 - val_accuracy: 0.6302
:poch 7/15
;05/605 [==============================] - 64s 106ms/step - loss: 0.8259 - accuracy: 0.6774 - val_loss: 0.9349 - val_accuracy: 0.6467
:poch 8/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.7808 - accuracy: 0.6971 - val_loss: 0.9141 - val_accuracy: 0.6459
:poch 9/15
;05/605 [==============================] - 63s 105ms/step - loss: 0.7295 - accuracy: 0.7190 - val_loss: 0.9389 - val_accuracy: 0.6418
:poch 10/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.6816 - accuracy: 0.7365 - val_loss: 0.9545 - val_accuracy: 0.6507
:poch 11/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.6356 - accuracy: 0.7541 - val_loss: 1.0074 - val_accuracy: 0.6430
:poch 12/15
;05/605 [==============================] - 62s 102ms/step - loss: 0.5857 - accuracy: 0.7700 - val_loss: 1.0205 - val_accuracy: 0.6472
:poch 13/15
;05/605 [==============================] - 62s 103ms/step - loss: 0.5464 - accuracy: 0.7887 - val_loss: 1.0480 - val_accuracy: 0.6465
:poch 14/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.5124 - accuracy: 0.7989 - val_loss: 1.1594 - val_accuracy: 0.6418
:poch 15/15
;05/605 [==============================] - 63s 104ms/step - loss: 0.4668 - accuracy: 0.8159 - val_loss: 1.1669 - val_accuracy: 0.6393
'usr/local/lib/python3.10/dist-packages/keras/src/engine/training.py:3103: UserWarning: You are saving your model as an HDF5 file via `m
  saving_api.save_model(
:ound 6043 images belonging to 5 classes.
.89/189 [==============================] - 7s 38ms/step - loss: 1.1423 - accuracy: 0.6424
'est Loss: 1.1422823667526245
'est Accuracy: 0.6423961520195007
.89/189 [==============================] - 6s 30ms/step
                      Filename Predicted   True
)  angry/PrivateTest_10131363.jpg     happy  angry
.  angry/PrivateTest_10304478.jpg  surprise  angry
!   angry/PrivateTest_1054527.jpg       sad  angry
;  angry/PrivateTest_10590091.jpg     angry  angry
;   angry/PrivateTest_1109992.jpg     angry  angry
```



```
              precision    recall  f1-score   support

       angry       0.60      0.44      0.51       958
       happy       0.75      0.81      0.78      1774
     neutral       0.52      0.61      0.56      1233
```