EXP 13 : EMOTION DETECTION USING FACIAL FEATURES

Aim: To detect emotions using facial features.

Algorithm:

1. **Import Required Libraries:**
    - Import necessary libraries like NumPy, TensorFlow/Keras, OpenCV, and Matplotlib.
2. **Data Preparation:**
    - Acquire a dataset of facial images with corresponding emotion labels.
    - Preprocess images (resize, grayscale, normalize).
3. **Model Creation:**
    - Build a Convolutional Neural Network (CNN) architecture.
    - Typical layers: Conv2D, MaxPooling2D, Flatten, Dense.
4. **Model Training:**
    - Train the CNN on the preprocessed dataset.
    - Use an optimizer (e.g., Adam) and loss function (e.g., categorical crossentropy).
5. **Prediction:**
    - Preprocess a new image.
    - Feed the image into the trained model.
    - Output the predicted emotion based on the model's output.

Program code:

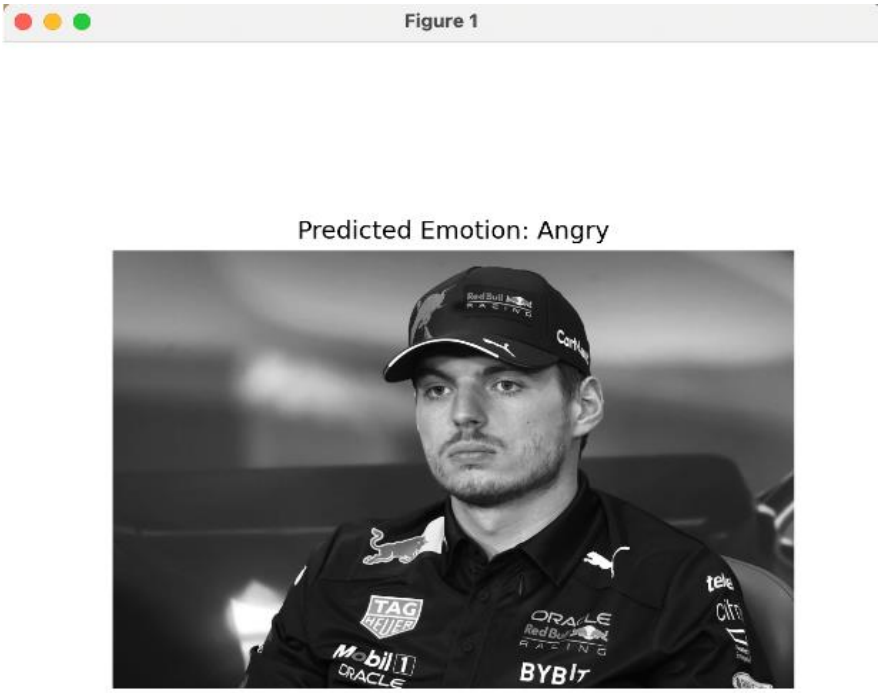
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
```

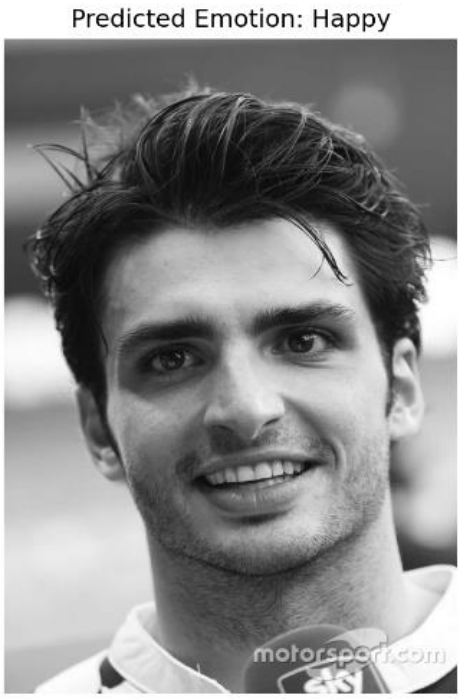
```
from PIL import Image

train_dir = '/Users/janesanjana/Downloads/fer2013/train'
val_dir = '/Users/janesanjana/Downloads/fer2013/test'

train_datagen = ImageDataGenerator(rescale=1.0/255.0,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True)
val_datagen = ImageDataGenerator(rescale=1.0/255.0)

train_generator = train_datagen.flow_from_directory(train_dir,
target_size=(48, 48),
batch_size=64,
color_mode='grayscale',
class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(val_dir,
target_size=(48, 48),
batch_size=64,
color_mode='grayscale',
class_mode='categorical')

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), activation='relu'))
```

```
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()

history = model.fit(train_generator,epochs=30,validation_data=validation_generator)

def preprocess_image(img_path):
img = image.load_img(img_path, color_mode='grayscale', target_size=(48, 48))
img = image.img_to_array(img)
img = np.expand_dims(img, axis=0)
img = img / 255.0
return img

img_paths = ['/Users/janesanjana/Downloads/happyface.jpg',
'/Users/janesanjana/Downloads/idkface.jpg']
sample_imgs = [preprocess_image(img_path) for img_path in img_paths]

predictions = [model.predict(sample_img) for sample_img in sample_imgs]
predicted_emotions = [np.argmax(prediction) for prediction in predictions]

emotion_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']
predicted_emotion_labels = [emotion_labels[predicted_emotion] for predicted_emotion in
predicted_emotions]

for i, img_path in enumerate(img_paths):
img = Image.open(img_path).convert('L')
```

```
plt.figure(figsize=(6, 6))
plt.imshow(img, cmap='gray')
plt.title(f'Predicted Emotion: {predicted_emotion_labels[i]}')
plt.axis('off')
plt.show()
```

Output:





Result: Thus emotion detection using facial features has been successfully implemented.