Emotion Recognition

1. Project Overview:-

This project focuses on developing a deep learning-based facial emotion recognition system that can accurately classify human facial expressions into distinct emotional states. Facial expression analysis has become an important area of research in fields such as affective computing, human-computer interaction, and psychology. Automatic emotion recognition from facial expressions helps build responsive and empathetic AI systems.

The main objective of this project is to create a robust and efficient model that can recognize seven fundamental human emotions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise. To achieve this, the system leverages Convolutional Neural Networks (CNNs) which are well-suited for image classification tasks. The project uses the FER-2013 dataset, a widely accepted benchmark in the emotion recognition domain, and applies image preprocessing techniques such as grayscale conversion, normalization, and data augmentation to enhance model generalization.

To address common challenges like class imbalance and overfitting, the training procedure includes the use of class weighting and EarlyStopping callbacks. These enhancements ensure that the model not only learns dominant classes like Happy and Neutral but also performs well on underrepresented classes like Disgust and Fear. Finally, the model is tested in real-time using webcam input and Haar cascades for face detection, enabling live emotion recognition from facial expressions.

This project demonstrates a complete pipeline from dataset preprocessing and deep learning model construction to model evaluation and real-time deployment, serving as a foundational step toward building intelligent and emotionally aware applications.

2. Key Components:-

- **Image Preprocessing:**
  - **Grayscale conversion**
  - **Resizing to 48x48 pixels**
  - **Data augmentation (rotation, zoom, flip, shear)**
- **Model Architecture:**
  - **Input: 48x48 grayscale images**
  - **4 Convolutional layers with ReLU activation**
  - **MaxPooling and Dropout layers for regularization**
  - **Dense layer with 512 neurons**
  - **Output: Softmax layer for 7 emotion classes**
- **Training Strategy:**
  - **Optimizer: Adam**
  - **Loss: Categorical Crossentropy**

- Metric: Accuracy
- EarlyStopping: Patience = 5, monitored on validation accuracy
- Class Weights: Computed using sklearn.utils.class_weight to handle imbalanced emotion classes

3. Dataset Details:-

- Dataset: FER-2013
- Total Training Images: ~28,709
- Total Testing Images: ~7,178
- Emotion Classes: Angry, Disgust, Fear, Happy, Neutral, Sad, Surprise
- Class Distribution: Highly imbalanced (Disgust has very few samples)

4. Model Training Summary:-

- Epochs Targeted: 101
- Early Stopped At: Typically around 36-45 epochs (based on val_accuracy stagnation)
- Final Training Accuracy: ~59% (variable)
- Final Validation Accuracy: ~56% (variable)
- Model Saved As: model_file.h5
- Training History Logged In: training_history.txt

5. Testing and Evaluation:-

- Test Data: FER-2013 test split
- Real-Time Test: Implemented using test.py with OpenCV and Haar Cascade for face detection
- Live Prediction: Emotion label is shown on detected faces in real-time webcam feed

6. Challenges Faced:-

- Handling class imbalance
- Preventing overfitting on dominant classes
- Training performance variation due to data shuffling
- Ensuring proper environment setup (e.g., Pillow, OpenCV, TensorFlow)

7. Solutions Applied:-

- Used EarlyStopping to avoid unnecessary training after validation stagnates
- Used class_weight to fairly learn underrepresented classes like Disgust and Surprise
- Enabled training history logging for later performance analysis

8. Future Enhancements:-

- Switch to more advanced datasets (e.g., AffectNet, RAF-DB)
- Try pretrained models (e.g., MobileNetV2 or EfficientNet)
- Use RGB instead of grayscale to improve accuracy
- Deploy as a web app or mobile app

9. Conclusion:-

This project successfully implements a real-time facial emotion recognition system using deep learning techniques. Through careful preprocessing, class balancing, and regularization, the CNN model is able to distinguish between seven human emotions with reasonable accuracy. The use of EarlyStopping prevents overfitting, while class weighting ensures that even less-represented emotions like Disgust and Fear are learned.

Overall, this project demonstrates the potential of deep learning in interpreting human emotions from facial expressions. It also highlights the importance of handling real-world challenges such as dataset imbalance and performance stability. With further improvements, such as using richer datasets and advanced architectures, this system could be integrated into real-world applications like virtual assistants, smart surveillance, and emotion-aware games. The knowledge and framework established here provide a solid foundation for more sophisticated and socially intelligent AI systems in the future.

10. Code:-

- **Main Code**

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPooling2D
from tensorflow.keras.callbacks import EarlyStopping
import os

# Paths
train_data_dir = 'data/train/'
validation_data_dir = 'data/test/'

# Image preprocessing
train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    shear_range=0.3,
    zoom_range=0.3,
    horizontal_flip=True,
    fill_mode='nearest'
)

validation_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    color_mode='grayscale',
    target_size=(48, 48),
    batch_size=32,
    class_mode='categorical',
    shuffle=True
)

validation_generator = validation_datagen.flow_from_directory(
    validation_data_dir,
    color_mode='grayscale',
    target_size=(48, 48),
    batch_size=32,
    class_mode='categorical',
```

```
model.add(Flatten())
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(7, activation='softmax'))

model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
print(model.summary())

# Count images
num_train_imgs = sum(len(files) for _, _, files in os.walk(train_data_dir))
num_test_imgs = sum(len(files) for _, _, files in os.walk(validation_data_dir))

print(f"Train images: {num_train_imgs}")
print(f"Test images: {num_test_imgs}")

# Callbacks
early_stop = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights=True)

# Train
epochs = 101
history = model.fit(
    train_generator,
    steps_per_epoch=num_train_imgs // 32,
    epochs=epochs,
    validation_data=validation_generator,
    validation_steps=num_test_imgs // 32,
    callbacks=[early_stop]
)

# Save model
model.save('model_file.h5')

# Save training history
with open("training_history.txt", "w") as f:
    for key in history.history:
```

- **Test Code**

```
import cv2
import numpy as np
from tensorflow.keras.models import load_model

model=load_model('model_file.h5')

video=cv2.VideoCapture(0)

faceDetect=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

labels_dict={0:'Angry',1:'Disgust', 2:'Fear', 3:'Happy',4:'Neutral',5:'Sad',6:'Surprise'}

while True:
    ret,frame=video.read()
    gray=cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces= faceDetect.detectMultiScale(gray, 1.3, 3)
    for x,y,w,h in faces:
        sub_face_img=gray[y:y+h, x:x+w]
        resized=cv2.resize(sub_face_img,(48,48))
        normalize=resized/255.0
        reshaped=np.reshape(normalize, (1, 48, 48, 1))
        result=model.predict(reshaped)
        label=np.argmax(result, axis=1)[0]
        print(label)
        cv2.rectangle(frame, (x,y), (x+w, y+h), (0,0,255), 1)
        cv2.rectangle(frame,(x,y),(x+w,y+h),(50,50,255),2)
        cv2.rectangle(frame,(x,y-40),(x+w,y),(50,50,255),-1)
        cv2.putText(frame, labels_dict[label], (x, y-10),cv2.FONT_HERSHEY_SIMPLEX,0.8,(255,255,255),2)

    cv2.imshow("Frame",frame)
    k=cv2.waitKey(1)
    if k==ord('q'):
        break

video.release()
cv2.destroyAllWindows()
```

Submitted By:- **Abhinav S Bisht**

Submitted To:- **Nandita Singla**