2025

Facial Expression Recognition System

COMPUTER VISION

LEADER: NIMRA (BIT21253)

University of the Punjab
Gujranwala Campus
Department of Information Technology

CERTIFICATE

This is to certify that the project titled
"Facial Expression Recognition System"

has been completed by the following students:

Group Leader: Nimra Shahid

Project Members:

Name:	**Roll no:**
Nimra	BIT21253
Isha	BIT21238
Hina	BIT21251

of the Seventh Semester, Bachelor of information technology in the year 2025 in partial fulfillment of the requirement to the award of course **"COMPUTER VISION"**

Project submitted to:
Ms. Fouqia Zafeer
Place : University of the Punjab, Gujranwala campus
Date : Februray 17,2025

Table of Contents

Project Overview

Project Title:

Facial Expression Recognition System

Objective

This project aims to develop a system capable of recognizing human facial expressions from both images and real-time video streams. It first identifies faces using the Haar Cascade classifier and then classifies the expressions using a pre-trained deep learning model built with TensorFlow and Keras.

Scope

The system is designed to:

Detect and classify seven distinct facial expressions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.

Support both static images and real-time video feeds for expression detection.

Utilize Python, TensorFlow, Keras, and OpenCV for implementation.

System Architecture

Components

1. Face Detection

Technology Used: OpenCV Haar Cascade Classifier

Description: The system leverages Haar Cascades to locate human faces in images or video frames.

2. Expression Recognition

Technology Used: TensorFlow, Keras, OpenCV

Description: Detected faces are processed and passed to a deep learning model, which classifies them into one of the seven predefined expressions.

Workflow

1. Input: The system receives an image or a live video feed.
2. Face Detection: The Haar Cascade classifier detects faces within the input frame.
3. Preprocessing: The detected faces are resized and normalized for the model.
4. Prediction: The processed face is fed into the trained deep learning model, which predicts the facial expression.
5. Output: The detected expression is displayed as a label on the image or video.

Setup Instructions

Environment Setup

Ensure Python and the required libraries are installed:

Pip install tensorflow opencv-python numpy keras

Project Directory Structure

Project-root/

```
│
├── model/
│   └── model.h5  # Trained Keras model
│
├── haarcascades/
│   └── haarcascade_frontalface_default.xml  # Haar Cascade for face detection
│
├── static_test.py   # Script for testing with static images
└── video_test.py    # Script for real-time video processing
```

Model Training

Dataset

The deep learning model is trained on a dataset of labeled facial expressions. Before training, images are converted to grayscale and resized to 48x48 pixels to maintain uniform input dimensions.

Model Architecture

The convolutional neural network (CNN) consists of multiple layers for feature extraction and classification:

```
From keras.models import Sequential

From keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout

Model = Sequential()

# Convolutional and Pooling Layers

Model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(48, 48, 1)))

Model.add(MaxPooling2D(pool_size=(2, 2)))

Model.add(Dropout(0.25))

Model.add(Conv2D(64, (3, 3), activation='relu'))

Model.add(MaxPooling2D(pool_size=(2, 2)))

Model.add(Dropout(0.25))

# Fully Connected Layers

Model.add(Flatten())

Model.add(Dense(128, activation='relu'))

Model.add(Dropout(0.5))

Model.add(Dense(7, activation='softmax'))  # 7 classes for 7 expressions

# Compile the Model

Model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])
```

Training Process

The model is trained using image generators that preprocess and augment the dataset:

```
Model.fit(train_generator, validation_data=validation_generator,
    Epochs=100, steps_per_epoch=224, validation_steps=56)
Model.save('model.h5')
```

Code Explanation

Static Image Testing (static_test.py)

This script processes a static image to detect and classify facial expressions.

Key Steps:

1. Load the pre-trained model (model.h5) and Haar Cascade classifier.
2. Convert the image to grayscale.
3. Detect faces using Haar Cascade.
4. Resize and normalize detected faces.
5. Pass the processed face to the deep learning model for prediction.
6. Display the image with a bounding box and label.

```
Import cv2
Import numpy as np
From keras.models import load_model
# Load pre-trained model
Model = load_model('model.h5')
# Load Haar Cascade
faceDetect = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
# Expression labels
Label_dict = {0:'Angry', 1:'Disgust', 2:'Fear', 3:'Happy', 4:'Neutral', 5:'Sad', 6:'Surprise'}
```

```
# Read and preprocess image
Input_img = cv2.imread('face2.jpg')
Gray = cv2.cvtColor(input_img, cv2.COLOR_BGR2GRAY)
Faces = faceDetect.detectMultiScale(gray, 1.3, 1)

# Process detected faces
For x, y, w, h in faces:
   Face = gray[y:y+h, x:x+w]
   Resized = cv2.resize(face, (48, 48)) / 255.0
   Reshaped = np.reshape(resized, (1, 48, 48, 1))

   # Predict expression
   Result = model.predict(reshaped)
   Label = np.argmax(result)

   # Draw bounding box and label
   Cv2.rectangle(input_img, (x, y), (x+w, y+h), (0, 0, 255), 1)
   Cv2.putText(input_img, label_dict[label], (x, y-10),
         Cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)

# Display output
Cv2.imshow("Frame", input_img)
Cv2.waitKey(0)
Cv2.destroyAllWindows()
```

Real-time Video Testing (video_test.py)

This script applies the same detection and classification process to live webcam video.

Key Steps:

1. Capture video frames from the webcam.
2. Convert frames to grayscale.
3. Detect faces and preprocess them.
4. Predict facial expressions and display the results in real-time.

```
Import cv2

Import numpy as np

From keras.models import load_model


# Load model and Haar Cascade

Model = load_model('model.h5')

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


# Label dictionary

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


# Start video capture

Video = cv2.VideoCapture(0)


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:
        Face = gray[y:y+h, x:x+w]
        Resized = cv2.resize(face, (48, 48)) / 255.0
        Reshaped = np.reshape(resized, (1, 48, 48, 1))

        Result = model.predict(reshaped)
        Label = np.argmax(result)

        Cv2.rectangle(frame, (x, y), (x+w, y+h), (0, 0, 255), 1)
        Cv2.putText(frame, label_dict[label], (x, y-10),
                Cv2.FONT_HERSHEY_SIMPLEX, 0.8, (255, 255, 255), 2)

    Cv2.imshow("Frame", frame)
    If cv2.waitKey(1) & 0xFF == ord('q'):
        Break

Video.release()
Cv2.destroyAllWindows()
```

Usage Instructions

1. Static Image Detection

Run the following command to analyze a static image:

Python static_test.py

Detects faces, predicts expressions, and displays results with bounding boxes.

2. Real-time Video Detection

Run the following command to analyze facial expressions in real-time:

Python video_test.py

Uses a webcam to detect and classify expressions live.

Press ‘q’ to exit.

3. Training a New Model (Optional)

To train a custom model:

Python train.py

Saves the trained model as model.h5.

4. Adjusting Face Detection

For better face detection, replace the Haar Cascade file in haarcascades/ and update scripts:

faceDetect = cv2.CascadeClassifier('haarcascades/haarcascade_frontalface_alt.xml')

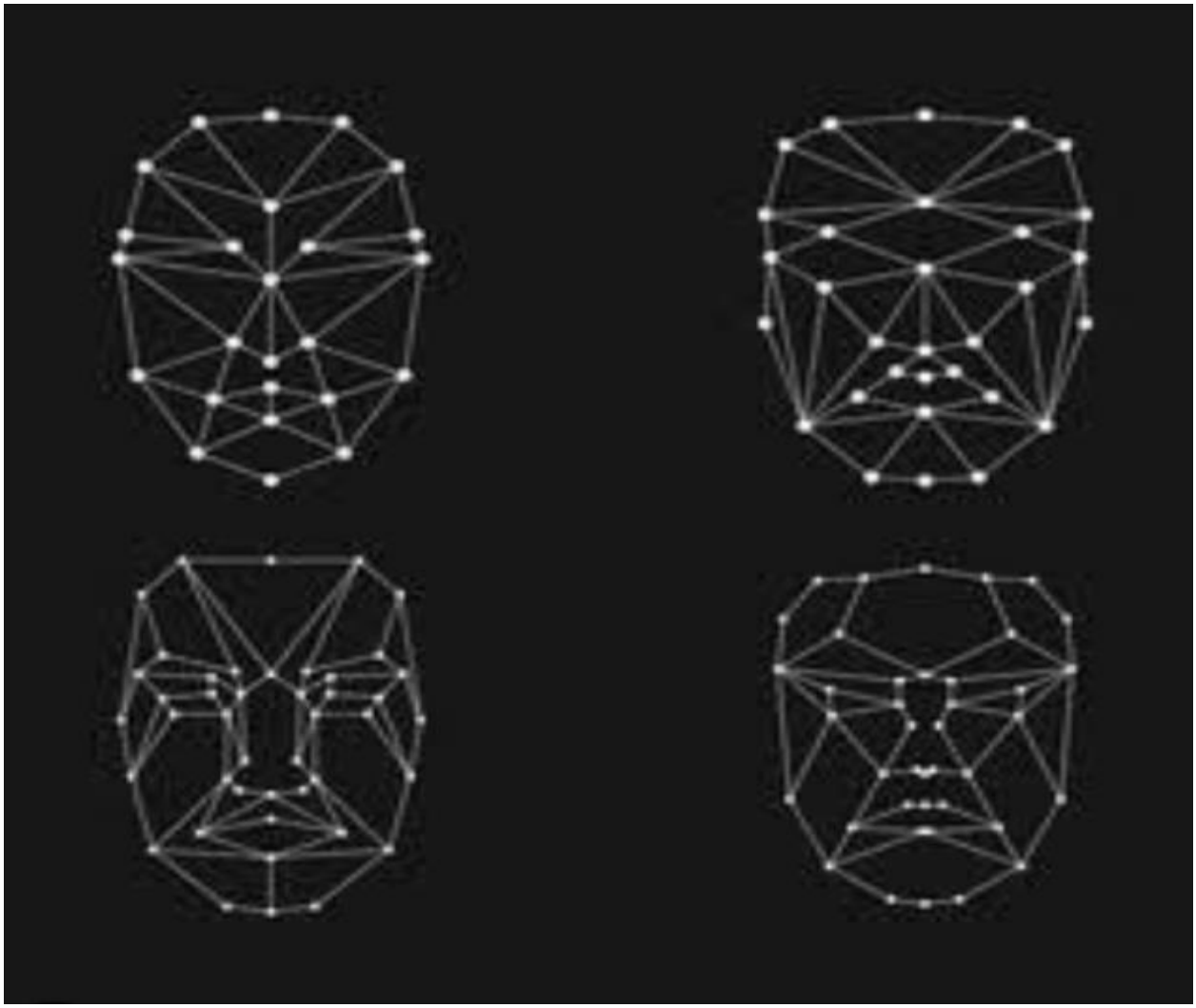
7. Model Logical Structure

7.1 Landmark Grids:

This image shows how the facial expression recognition model detects facial landmarks (key points on the face) and overlays a mesh grid on these points. The grid helps in analyzing the movement and position of facial features to determine expressions like "Laughing" and "Surprise".

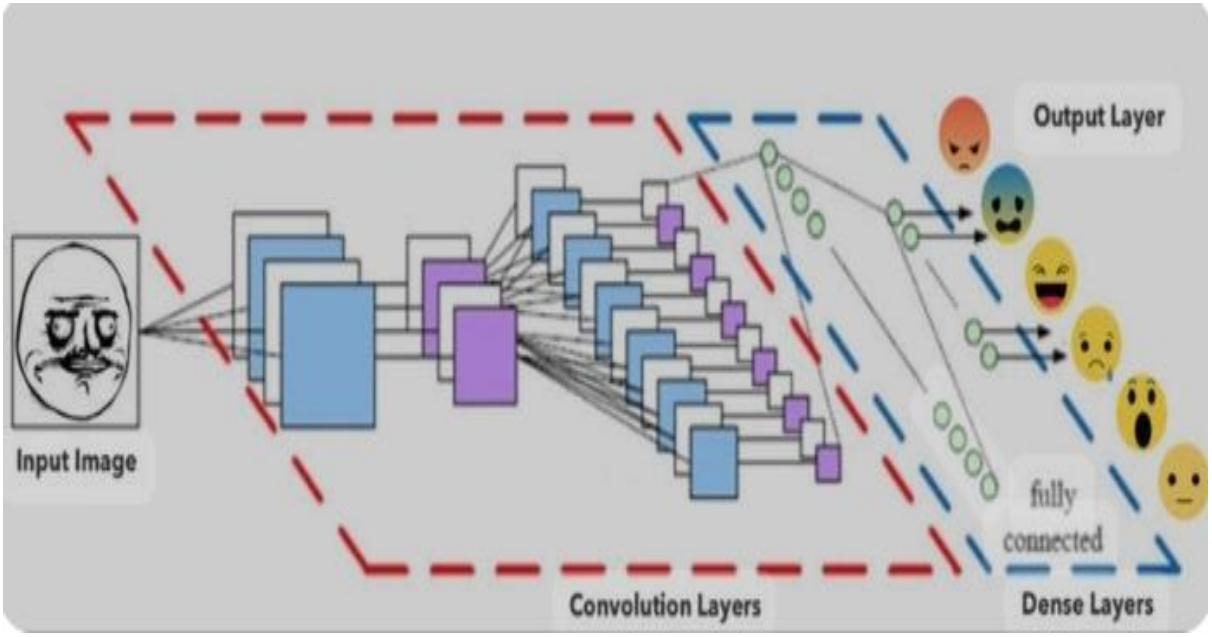
7.2 Mesh Grids:

This image displays multiple faces with overlaid mesh grids, representing the facial landmarks detected by the model. These grids are used to understand the structure and movements of the facial features, which are critical for identifying different expressions.



7.3 CNN Architecture

This diagram effectively represents the flow of data through a CNN designed for facial expression recognition, from an initial face image to final predictions.



Conclusion

This project successfully implements a Facial Expression Recognition System using deep learning and computer vision. The system can detect faces and classify expressions in both static images and real-time video. Future improvements could involve enhancing model accuracy, adding more expressions, or integrating the system into an interactive application.