WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

FACE EMOTION RECOGNITION USING PYTHON AND MACHINE LEARNING

Lokesh Yadav[1], Chandrapraksh Patel[2], Luminash Patel[3], Kaushal Chandrakar[4], Prof. Vijaya Tripathi Mam[5]

Roll no. (301411521038)
Roll no. (301411521039)
Roll no. (301411521040)
Roll no. (301411521001)
[1,2,3,4]UG Students Department of Computer Science and Engineering
[5] Guide, Department of Computer Science and Engineering , Shri Shankaracharya Technical Campus Bhilai,

ABSTRACT:

Emotion recognition is one of the emerging trends in computer vision and human-computer interaction. With increasing applications in healthcare, education, entertainment, surveillance, and customer service, face emotion recognition systems are now considered crucial for next-generation AI systems. This research presents a real-time face emotion recognition system developed using Python, OpenCV, TensorFlow, and machine learning techniques. The system detects a human face from live webcam input, processes it through a trained neural network, and classifies the emotion as one among five primary emotions: *Happy, Angry, Sad, Neutral,* and *Surprised*. The project demonstrates how deep learning and computer vision techniques can work together for effective emotion analysis, achieving practical results in real-time scenarios.

Introduction

Facial expressions are a natural way for humans to communicate emotions. Recognizing these expressions using a machine allows automation of tasks where emotional intelligence is required. The goal of our major project is to develop a system that recognizes emotions from facial expressions in real-time using machine learning and deep learning models. The project makes use of the OpenCV library for video frame capturing and face detection, and TensorFlow for implementing and running the deep learning model.

Problem Statement

Despite the progress in AI and computer vision, accurately identifying emotions from faces in real-time remains a challenge due to variations in lighting, face angles, occlusion, and subtle expression changes. The purpose of this project is to design a robust, real-time, emotion recognition model that operates efficiently on standard hardware and offers high accuracy in typical real-world conditions.

Objectives

- To build a system that detects human faces in real-time using OpenCV.
- To train a deep learning model using TensorFlow to classify emotions.
- To deploy the trained model for real-time inference.
- To provide a user-friendly output displaying the detected emotion label.
- To analyze the accuracy and performance of the system across different users.

Tools and Technologies

- **Programming Language:** Python

- **Libraries Used:** OpenCV, TensorFlow, NumPy, Keras, Matplotlib
- **Model Type:** Convolutional Neural Network (CNN)
- **Dataset Used:** FER-2013 (Facial Expression Recognition dataset)
- **Development Tools:** Visual Studio Code, Jupyter Notebook

- **Hardware Requirements:** Laptop with integrated camera, 8GB RAM minimum

Methodology

1. **Data Collection:**
   We used the publicly available FER-2013 dataset containing grayscale images of facial expressions labeled with emotion categories. The dataset was preprocessed to normalize image size and format.

2. **Model Design:**
   A Convolutional Neural Network (CNN) was designed and trained to classify the five target emotions. The model architecture includes multiple convolutional, pooling, dropout, and dense layers to optimize accuracy and reduce overfitting.

3. **Training and Validation:**
   The dataset was split into training and validation sets. The model was trained using categorical cross-entropy as the loss function and Adam optimizer. Accuracy and loss were monitored across epochs.

4. **Real-Time Detection:**
   A Python script using OpenCV captures video from the webcam. Faces are detected using Haar cascade classifiers and passed to the trained model for emotion prediction. The predicted emotion is displayed on the video feed.

5. **Model Deployment:**
   The trained model was saved in JSON and H5 format for easy loading and inference. The complete system runs locally without the need for cloud services.

Results and Observations

The trained model achieved an overall accuracy of approximately 75% on the validation dataset. Real-time performance was satisfactory with a detection speed of around 15-20 frames per second. Emotions such as "Happy" and "Neutral" were detected more accurately than "Sad" or "Surprised", due to clearer facial features. The system handled varied lighting and different users reasonably well, though performance dropped slightly in dim light or partial occlusion.

Applications

- **Healthcare:** Monitoring emotional wellbeing in patients.
- **Education:** Real-time feedback on student engagement.
- **Security:** Behavior analysis in surveillance systems.
- **Retail:** Customer mood analysis to improve service.
- **Robotics:** Enhancing human-robot interaction.

Conclusion and Future Work

This project successfully demonstrates a working model for facial emotion recognition using Python, OpenCV, and TensorFlow. The system achieves realtime emotion classification with reliable accuracy and performance. In the future, we plan to expand the model to recognize more emotions, improve robustness under diverse conditions, and integrate speech sentiment for multimodal emotion recognition.

9. REFERENCES / SOURCES

• FER-2013 Dataset: Used for training and testing the CNN model for emotion recognition. Source – Kaggle.
https://www.kaggle.com/datasets/msambare/fer2013

• Python Programming Language: Used as the primary language for implementing the emotion recognition system. Installed and managed via VS Code terminal. https://www.python.org

• TensorFlow Library: Used for building and training the deep learning (CNN) model. Installed using pip install tensorflow in Visual Studio Code terminal.

• OpenCV Library: Used for real-time face detection and webcam-based image capture. Installed using pip install opencv-python in Visual Studio Code terminal.
• NumPy Library: Used for performing matrix operations and numerical tasks. Installed using pip in the Python environment.

• Visual Studio Code (VS Code): Used as the main IDE for writing, testing, and executing Python code. https://code.visualstudio.com

• Jupyter Notebook: Used for testing model logic, training processes, and displaying results during development. https://jupyter.org

• ChatGPT (OpenAI): Used for guidance, technical explanation, code understanding, report writing assistance, and overall conceptual support. https://chat.openai.com