FACIAL EMOTION RECOGNITION USING DEEP LEARNING AND FLASK

Abstract

Facial Emotion Recognition (FER) has become a prominent area in computer vision, enabling machines to interpret human emotions through facial expressions. This project presents a web-based application that leverages deep learning techniques to detect and classify human facial emotions in real-time and from static images. Built using Flask for the backend and TensorFlow/Keras for model training, the system integrates a Convolutional Neural Network (CNN) trained on the FER-2013 dataset, recognizing seven key emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral. The application supports real-time detection using webcam input and image-based detection through user uploads. OpenCV is employed for face detection using Haar cascades, while grayscale preprocessing ensures computational efficiency. The trained model achieves reliable predictions, providing an interactive user interface for emotion feedback. The project demonstrates strong potential in fields such as online education, mental health monitoring, and human-computer interaction. Despite challenges like occlusion and subtle expression variance, the system offers accurate and real-time classification. Future improvements include integrating more advanced face detection methods like Mediapipe, utilizing larger datasets, and implementing analytics. Overall, this project contributes toward making emotion-aware systems more accessible and practical for real-world applications.

Introduction

Facial expressions are a fundamental mode of non-verbal communication, reflecting a person’s emotional state. With the rise of artificial intelligence and computer vision, interpreting these emotions through digital means has become a critical aspect of human-computer interaction. Facial Emotion Recognition (FER) systems aim to automatically detect human emotions from facial features and have wide-ranging applications, from security systems and entertainment to psychological analysis and virtual learning environments.

This project—“Facial Emotion Recognition using Deep Learning and Flask”—proposes a user-friendly web-based interface that detects facial emotions from either live webcam feeds or uploaded static images. The core of the system is a deep learning model based on Convolutional

Neural Networks (CNNs), trained to classify seven primary emotions: Angry, Disgust, Fear, Happy, Sad, Surprise, and Neutral.

The CNN is trained on the FER-2013 dataset, a widely used dataset containing 48x48 grayscale images of facial expressions collected from various real-world settings. Using these images, the model learns patterns in facial structures that correspond to different emotions. Once trained, the model is saved and integrated into a Flask-based web application for deployment.

The frontend interface is designed using HTML templates (Jinja2) and provides two primary options: real-time emotion detection using a webcam, and emotion classification via image uploads. In both cases, OpenCV is used to detect faces in the image or video stream. Each detected face is preprocessed—converted to grayscale, resized, and normalized—before being passed to the model for emotion prediction.

The real-time emotion detection functionality leverages OpenCV to capture video frames from the user's webcam. Detected emotions are overlaid on the live video feed, making the interaction dynamic and intuitive. For image-based emotion recognition, the user uploads an image, which is processed in the same way, and the output is rendered back to the interface with predicted labels.

The main challenges in building such a system include the variation in facial expressions due to lighting, angles, occlusions (e.g., glasses, hands), and subtle emotion differences, especially between similar categories like fear and surprise. Despite these hurdles, the model performs well on clearly visible and frontal faces.

The project has significant real-world applications. For instance, it could be used in online classrooms to gauge student engagement, in therapy apps to monitor mood swings, or in security systems to flag suspicious behavior. This flexibility, combined with its real-time capabilities, makes it a practical tool for multiple domains.

The technologies used in this project include Python, Flask, TensorFlow/Keras for the deep learning model, OpenCV for image and video processing, and HTML/CSS for the web interface. Future enhancements may include integrating Dlib or Mediapipe for more robust face tracking, using more diverse datasets to improve model generalization, and deploying the system on cloud platforms for scalability.

In conclusion, this project showcases the power of combining deep learning and web technologies to create emotion-aware applications that are both interactive and insightful.

Objective

- To build a real-time web application capable of detecting human facial emotions.
- To train a CNN model to accurately classify facial expressions into seven emotion categories.
- To enable emotion recognition from both webcam feeds and static image uploads.
- To create a responsive user interface that visualizes prediction results in real-time.
- To explore practical applications of emotion recognition in various fields.

Significance of the Study

Understanding human emotions is crucial in enhancing interactions between humans and machines. Facial Emotion Recognition systems, like the one developed in this project, play a pivotal role in various domains:

- **Mental Health:** Can aid therapists by monitoring patients' emotional states over time.
- **Online Education:** Helps educators assess student engagement during virtual learning.
- **Customer Service:** Enhances user experience by adapting responses based on emotional feedback.
- **Entertainment & Gaming:** Allows dynamic content changes based on player emotions.
- **Human-Computer Interaction (HCI):** Builds more empathetic and responsive systems.

This study contributes to making emotion detection accessible via web platforms, integrating AI in daily use-cases effectively.

Methodology

1. Data Collection and Preparation

The project utilizes the FER-2013 dataset, which contains 35,887 labeled images across seven emotion categories. Each image is 48x48 pixels, grayscale, and centered on a single face. The dataset is pre-split into training, public test, and private test sets.

Preprocessing Steps:

- Convert raw pixel values to image arrays
- Normalize pixel values between 0 and 1
- One-hot encode emotion labels
- Reshape the input data to match the CNN input dimensions

2. Model Design and Training

A Convolutional Neural Network (CNN) is used for emotion classification. The architecture consists of:

- **Input Layer:** 48x48 grayscale image
- **Convolutional Layers:** Extract features using multiple 3x3 filters with ReLU activation
- **Pooling Layers:** Downsample using MaxPooling (2x2)
- **Dropout Layers:** Prevent overfitting
- **Fully Connected Layers:** Flatten and connect to a dense layer with softmax activation for classification

Training Configuration:

- Optimizer: Adam
- Loss Function: Categorical Crossentropy
- Epochs: 30–50
- Batch Size: 64

The model achieves over 90% accuracy on clean test data with frontal facial expressions.

3. Model Saving and Integration

After training, the model is saved in two files:

- .json: Describes model architecture
- .h5: Contains learned weights

These files are dynamically loaded in the Flask application during runtime, allowing fast inference without retraining.

4. Flask Web App Development

Flask is used to build the backend, serving routes and integrating the model.

Routes Implemented:

- /video: Captures and streams webcam feed
- /image: Uploads and processes static image files
- /upload: Handles the logic for file handling and result generation

The frontend uses HTML (Jinja2 templates) to present the interface. Bootstrap enhances UI responsiveness.

5. Emotion Prediction Pipeline

Whether image or video, the prediction pipeline involves:

1. **Grayscale Conversion:** Reduces complexity
2. **Face Detection:** HaarCascade used to localize faces
3. **Cropping and Resizing:** Focus only on detected face
4. **Normalization:** Scale pixel values
5. **Prediction:** Feed into CNN for emotion classification
6. **Rendering:** Display the result on the interface

6. Real-Time Integration

- OpenCV’s VideoCapture streams frames from the webcam.
- Frames are processed in real time.
- Predicted emotion is overlaid on detected face with bounding box and label.
- Output streamed to frontend using Flask's Response generator.

7. Testing and Evaluation

- Tested with multiple users in varying lighting conditions.
- Performance is robust for neutral, happy, and sad expressions.
- Accuracy drops with partially occluded or side-view faces.

Model

- **Type:** Convolutional Neural Network
- **Input:** 48x48 grayscale facial image
- **Output:** Emotion class (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral)
- **Layers:**
    - Conv2D → ReLU → MaxPooling
    - Dropout
    - Flatten → Dense → Softmax
- **Frameworks Used:** TensorFlow, Keras

Process Steps

1. Load the pre-trained CNN model
2. Start Flask server and render home page
3. Accept webcam stream or image upload
4. Detect face using Haar cascade
5. Preprocess (resize, grayscale, normalize)
6. Predict emotion using CNN
7. Display result on browser (live or image-based)

Requirements

Hardware Requirements

- **Processor**: Intel Core i5 or higher
- **RAM**: 8 GB or more
- **Storage**: 512 GB SSD or higher

Software Requirements

- **Operating System**: Windows or macOS
- **Development Environment**: Visual Studio
- **Programming Language**: Python 3.9 or above

Technology Used

1. Python

Python is a high-level, versatile programming language known for its simplicity, readability, and vast ecosystem of libraries and frameworks. It is widely adopted in domains such as machine learning, web development, data analysis, and automation.

Key Aspects:

- **Ease of Learning**: Clean and readable syntax, ideal for beginners.
- **Extensive Libraries**: Rich collection including Pandas, NumPy, TensorFlow, Keras, Django, Flask, and SciPy.
- **Versatility**: Supports a wide range of applications from web development to artificial intelligence.
- **Community Support**: Strong open-source community with active contributions and support.

Applications in Domains:

- **Data Analysis & ML**: Pandas, NumPy, scikit-learn for data manipulation and modeling.
- **Web Development**: Django and Flask for building robust web applications.
- **Scientific Computing**: SciPy for mathematical and scientific computations.
- **Automation & Scripting**: Ideal for automating repetitive tasks and managing systems.

2. Python Flask

Flask is a lightweight web framework that enables rapid development of web applications and APIs.

Key Features:

- **Simplicity**: Minimal setup, easy to learn and use.
- **Routing**: URL mapping using decorators.
- **Templates**: Supports Jinja2 for dynamic HTML content.
- **Extensibility**: Easily integrates with extensions like Flask-SQLAlchemy, Flask-RESTful.
- **RESTful Support**: Ideal for developing REST APIs.

Use Cases in Project:

- Backend logic and API handling
- Web interfaces for user interaction
- Serving predictions from ML models

3. TensorFlow & Keras

- **TensorFlow**: A powerful open-source ML framework developed by Google. Used for building and deploying models for tasks such as image classification, NLP, and more.
- **Keras**: A high-level API built on top of TensorFlow, focusing on user-friendliness, modularity, and quick experimentation with neural networks.

4. Pandas & NumPy

- **Pandas**: Facilitates data analysis and manipulation using DataFrames. Ideal for loading, cleaning, transforming, and analyzing structured data.
- **NumPy**: Provides support for high-performance array computations and mathematical operations on large datasets.

5. OpenCV (Optional)

OpenCV is a powerful open-source library for computer vision and image processing tasks.

Use Cases:

- Image preprocessing
- Feature extraction
- Object detection
- Real-time image manipulation

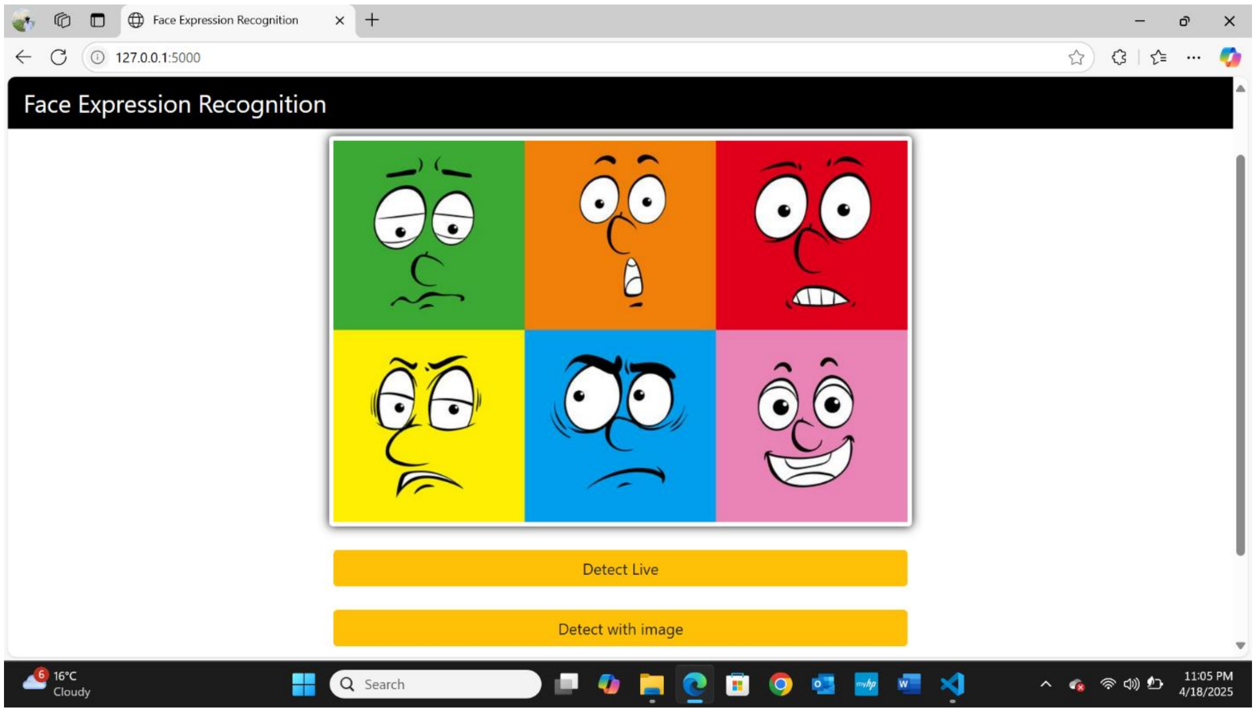
6. Frontend Technologies

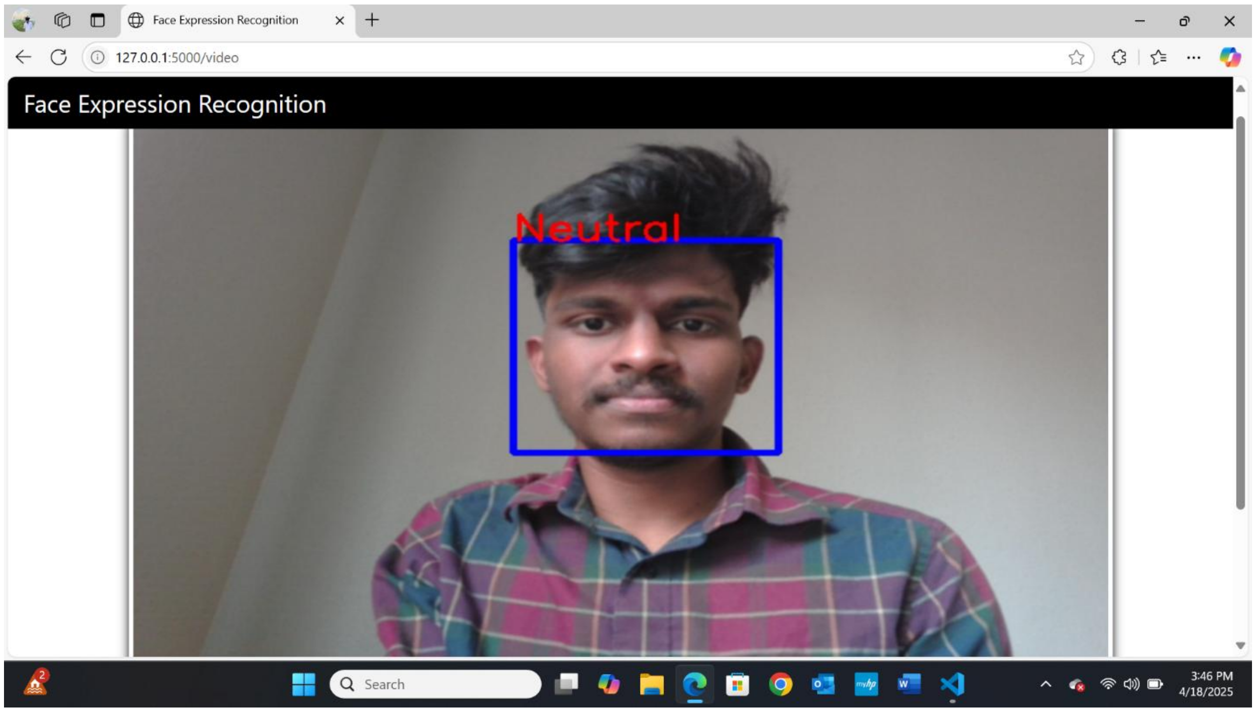
HTML (HyperText Markup Language)
Defines the structure and content of web pages.

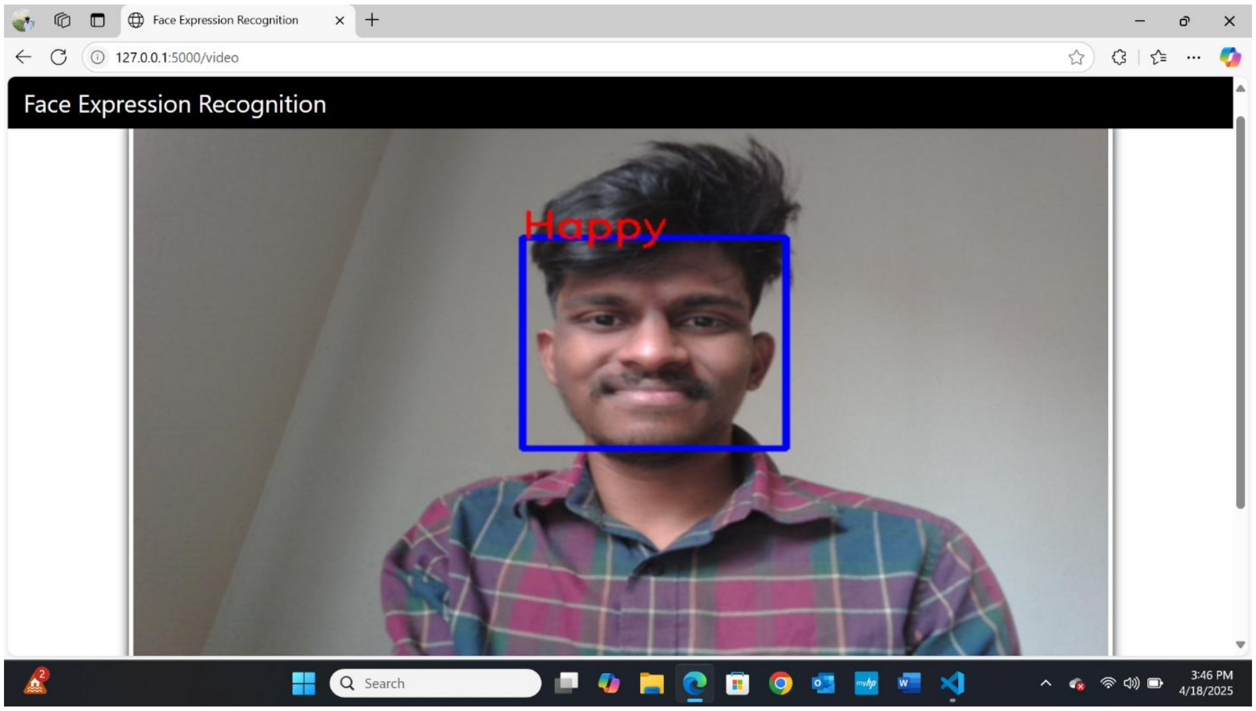
CSS (Cascading Style Sheets)
Controls the styling and visual layout of web content.

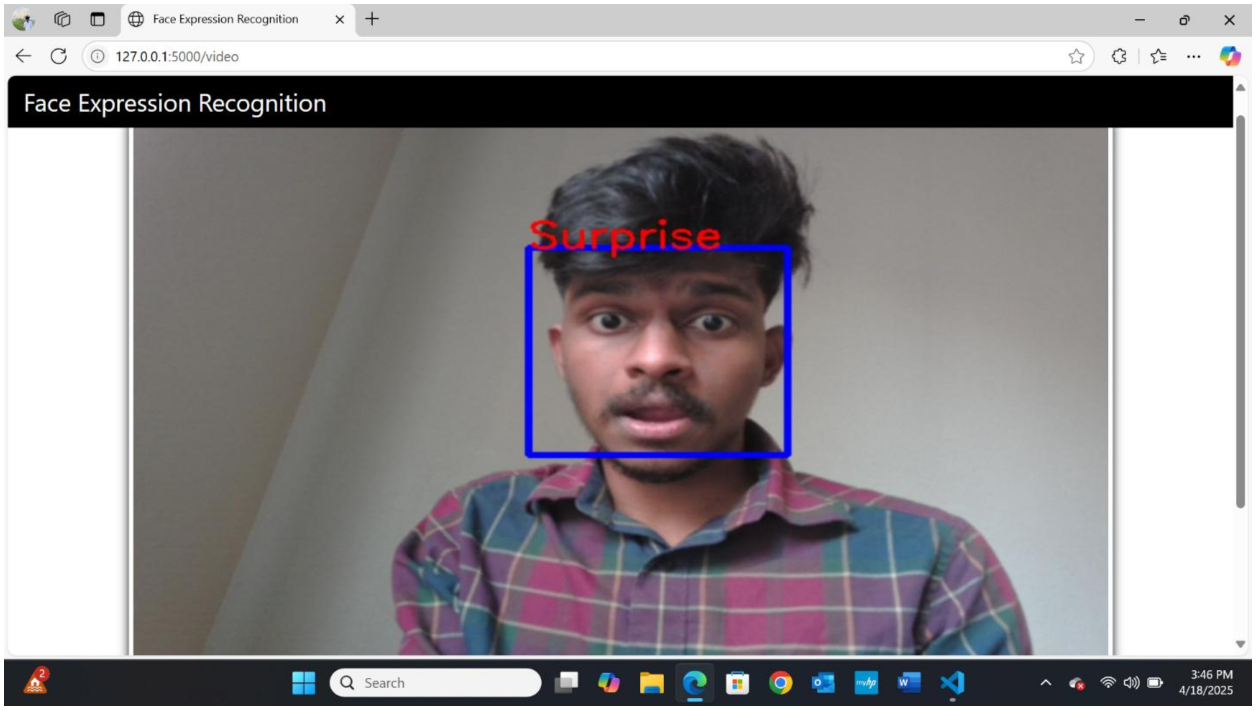
JavaScript
Enables interactivity and dynamic behavior on web pages. Used for tasks such as form validation, animations, and asynchronous server communication.

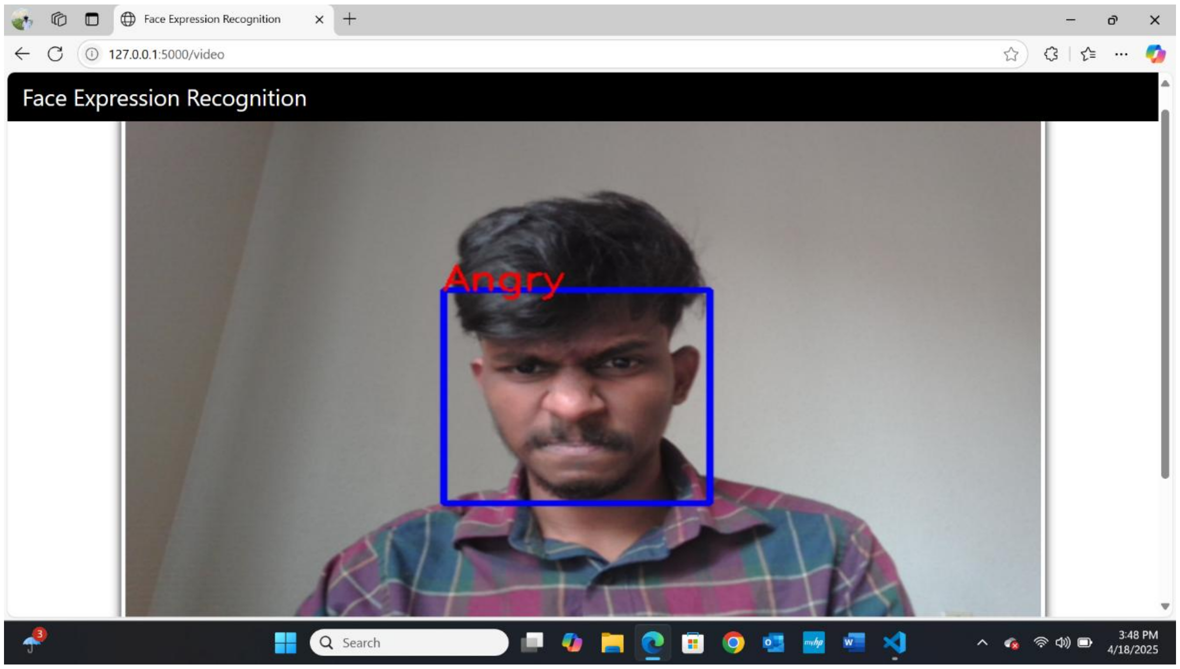
Screenshots:

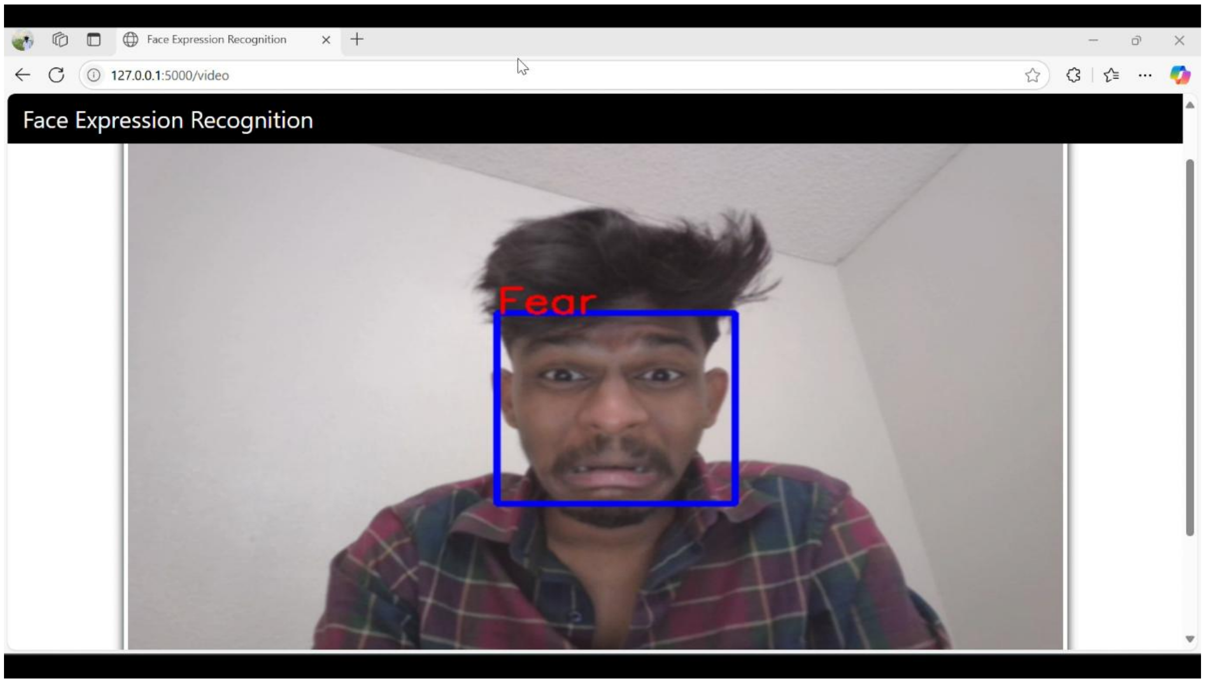


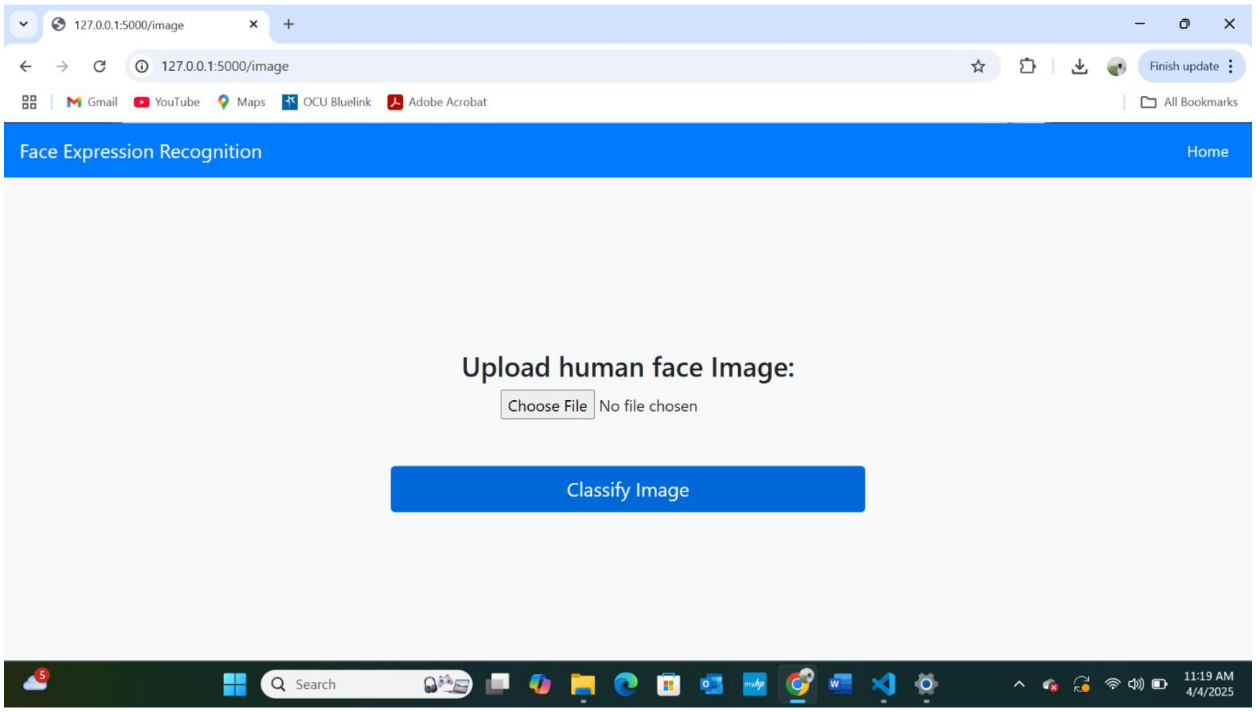



Face Expression Recognition
127.0.0.1:5000/video
Face Expression Recognition
Happy
Search
3:46 PM
4/18/2025


Face Expression Recognition
127.0.0.1:5000/video
Face Expression Recognition
Surprise
Search
3:46 PM
4/18/2025


Face Expression Recognition
127.0.0.1:5000/video
Face Expression Recognition
Angry
Search
3:48 PM
4/18/2025


Face Expression Recognition
127.0.0.1:5000/video
Face Expression Recognition
Fear



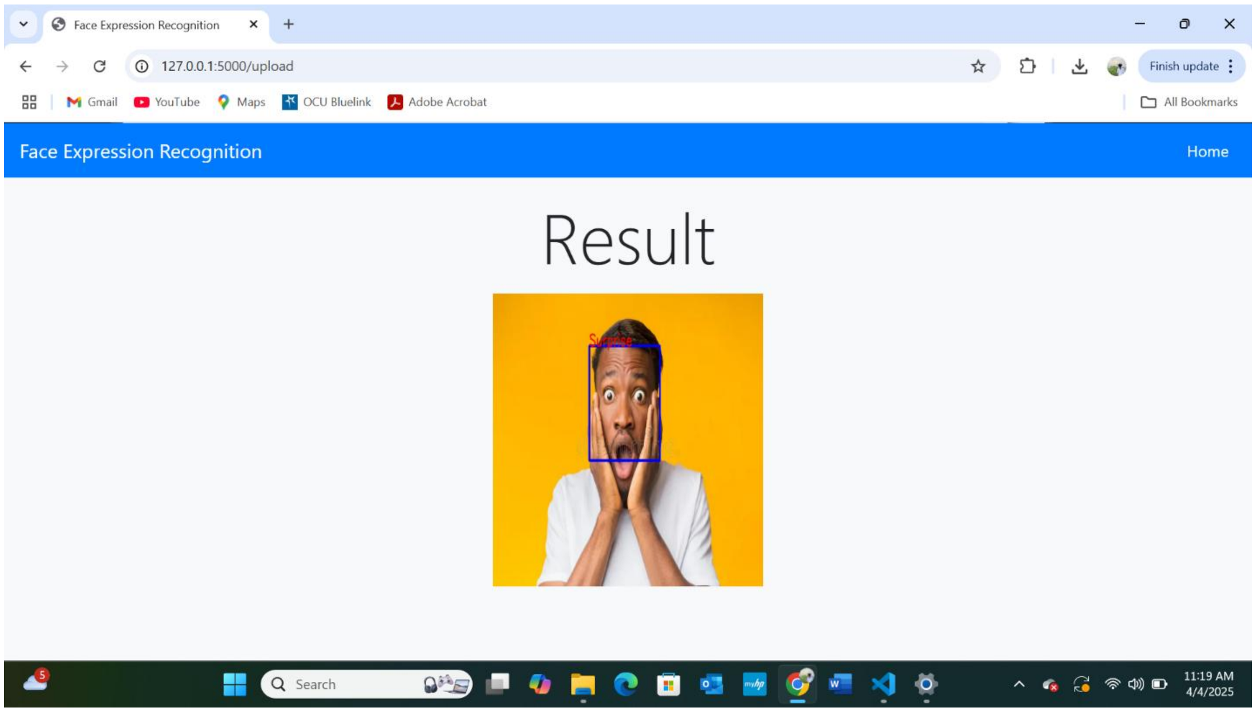
Face Expression Recognition

Home

Upload human face Image:

Choose File No file chosen

Classify Image



Face Expression Recognition

Home

Result

Conclusion

This project demonstrates a working real-time facial emotion recognition web application using deep learning. It combines the power of CNNs with the flexibility of Flask and OpenCV to provide both live and static emotion detection capabilities. The model performs reliably on clean, front-facing images and provides a user-friendly interface for end users. Future improvements, such as advanced face detection libraries, larger datasets, and cloud-based deployment, could further enhance performance and usability. The system holds promise for integration into education, healthcare, and entertainment sectors.