Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 07, July - 2025

Countenance Detection using CNN: An end-to-end Real-Time Emotion Recognition Web Application

Akki Eshwar, Shailesh Bhosekar
Department of Computer Science and Engineering
Keshav Memorial Institute of Technology (Autonomous), Hyderabad, India
Kaggle: https://www.kaggle.com/code/akkieshwar/notebook5c6bb4febe
GitHub: https://github.com/akkieshwar/emotion-detection

***Abstract*—In this paper, we present an educational, end-to-end implementation of a real-time facial emotion detection system using a Convolutional Neural Network (CNN), deployed as a browser-based web application. The system combines a Keras/TensorFlow model trained on a public facial expression dataset, a Flask backend, and a webcam-enabled frontend. Browser-captured images are sent to the backend for preprocessing and inference, with predictions returned in real time for live display. This project integrates open-source code and tutorials (with attribution), expands them with a custom-trained model, and documents the entire pipeline for educational reproducibility. The solution demonstrates practical integration of standard AI and web technologies, with applications in affective computing, education, and customer service.**

***Keywords*—Emotion detection, computer vision, CNN, Keras, TensorFlow, Flask, real-time web app, affective computing, educational implementation.**

I. INTRODUCTION

Accurately interpreting human emotions from facial expressions is an essential task in affective computing, with broad applications in mental health, education, customer service, and human-computer interaction. Deep learning, especially Convolutional Neural Networks (CNNs), has enabled reliable automated emotion recognition from images. This paper describes an end-to-end, real-time emotion detection system: a custom-trained CNN model is deployed as a Flask web service, accessed from a live webcam-enabled browser UI. The project demonstrates how to translate theory into a complete, usable AI application using open-source tools.

II. RELATED WORK

Numerous tutorials and open-source projects exist for facial emotion detection, often demonstrating isolated parts (e.g., only model training or only web deployment). Many systems are based on the FER-2013 dataset [1] and Keras/TensorFlow example models [2]. Our approach adapts and extends these resources, providing a complete, reproducible workflow from data to live application, as well as documenting engineering and deployment decisions for students and practitioners. Prior YouTube and GitHub tutorials are acknowledged for basic frontend/backend code structure.

III. SYSTEM OVERVIEW

A. Architecture

Frontend: HTML, CSS, JavaScript. Captures webcam video, sends image frames to the backend via HTTP POST, and displays predicted emotion live.

Backend: Flask (Python). Receives images, performs face detection and preprocessing (grayscale, resize, normalize), loads and runs the CNN model, and returns predictions.

Model: Keras/TensorFlow CNN trained on a public dataset to classify emotions (Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral).

Integration: End-to-end local deployment for live demo.

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 07, July - 2025

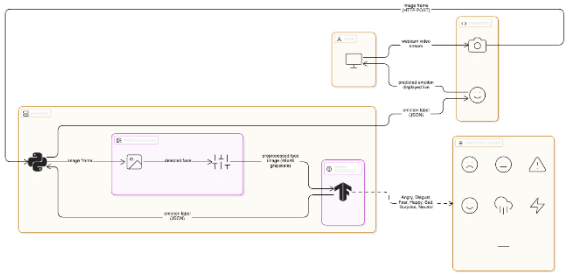

Fig. 1. Architecture of the Proposed Emotion Detection System

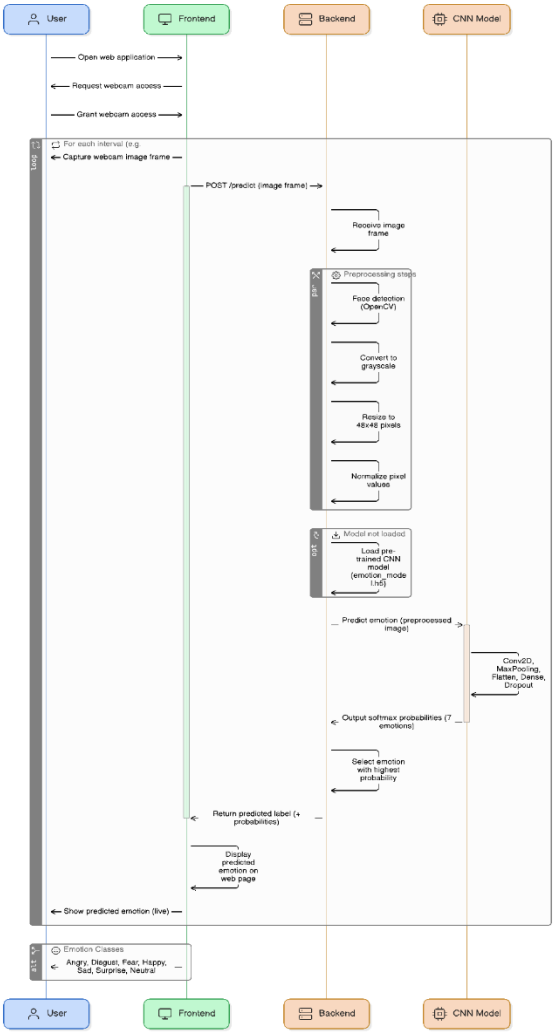

Fig. 2. Component flow design

IV. DATASET AND MODEL

A. Dataset

- Public facial expression dataset.
- Preprocessing: face detection, grayscale conversion, resize to 48x48, normalization (pixel values scaled to [0,1]), data augmentation for robustness.

B. Model

- Keras Sequential CNN with multiple Conv2D, MaxPooling, Flatten, Dense, Dropout layers.
- Activation: ReLU (hidden), Softmax (output).
- Trained with categorical cross-entropy loss and Adam optimizer.
- Achieved 63.5% accuracy on validation set.

V. IMPLEMENTATION DETAILS

A. Backend (Flask, Python)

- app.py handles image upload, loads emotion_model.h5, preprocesses images, predicts emotion, and returns result as JSON
- Uses OpenCV for face detection and image handling
- Fast API response times (<150 ms per image)

B. Frontend (HTML/JS/CSS)

- index.html, script.js, styles.css: accesses webcam with navigator.mediaDevices.getUserMedia, captures frames, sends to /predict endpoint via AJAX, receives and displays predicted emotion

VI. RESULTS AND DISCUSSION

The application delivers real-time emotion detection in a standard browser environment.

Softmax probabilities are returned for each class, with the label of highest probability shown live.

Performance: System predicts emotions within 100–150 ms per frame on a standard PC

Limitations: Accuracy depends on dataset quality and lighting; edge cases (occluded faces, low light) are more challenging; no transfer learning from larger face models (could be a future upgrade)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 14 Issue 07, July - 2025

Test Image	sad	neutral	surprise	fear	happy	angry	disgust	Predicted Label
1	0.0629	0.0079	0.0748	0.4764	0.2288	0.0903	0.0589	fear
2	0.1475	0.0035	0.4871	0.0059	0.0455	0.0485	0.2619	surprise
3	0.0177	0.0001	0.2047	0.0089	0.0055	0.0080	0.7550	disgust
4	0.0126	0.0002	0.0098	0.4952	0.4581	0.0199	0.0043	fear
5	0.0314	0.0002	0.3001	0.0015	0.0191	0.0164	0.6314	disgust

Table 1. Sample Prediction Outputs and Model Decision

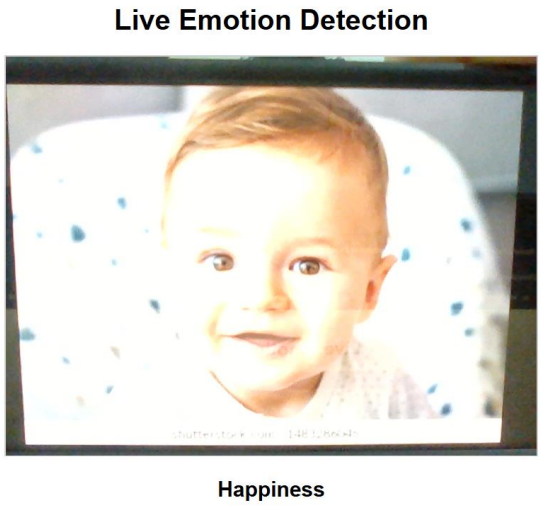

Image 1

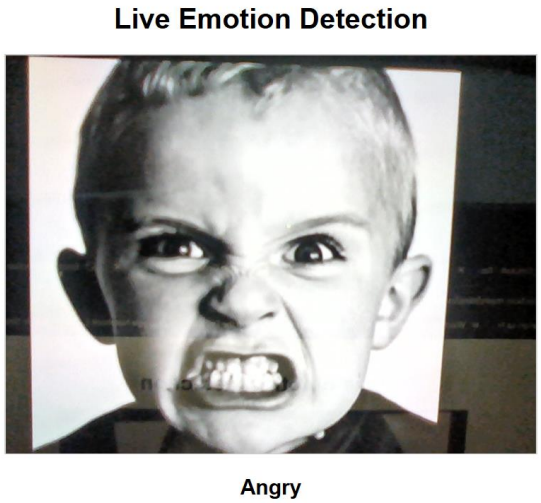

Image 2

VII. LIMITATIONS AND FUTURE WORK

Model Improvements: Upgrade to modern architectures (e.g., ResNet, EfficientNet), or use transfer learning for higher accuracy

Real-time Video: Streamline for higher FPS and better latency

Multimodal Emotion Recognition: Combine facial, vocal, and text cues

Deployment: Create a portable/mobile version or cloud deployment

VIII. CONCLUSION

This project demonstrates the practical, educational implementation of an end-to-end real-time facial emotion detection system using modern AI and web technologies. The codebase, model, and documentation are provided for reproducibility and as a teaching tool for students entering AI, computer vision, and full-stack engineering.

IX. REFERENCES

[1] I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. MIT Press, 2016.

[2] F. Chollet, Deep Learning with Python. Manning Publications, 2018.

[3] FER-2013 Kaggle Dataset, 2013. [Online]. Available: https://www.kaggle.com/datasets/msambare/fer2013

[4] TensorFlow Documentation. [Online]. Available: https://www.tensorflow.org

[5] OpenCV Documentation. [Online]. Available: https://opencv.org

[6] [YouTube Author], “Tutorial Title,” YouTube, [Online]. Available: [https://youtube.com/link-to-tutorial]

[7] [GitHub Author], “Repository Title,” GitHub, [Online]. Available: [https://github.com/link-to-repo].