Real-time Facial Emotions Classifier

Team Members: Dharna Mittal (12212143), Ashok (12212133)

Computer Engineering Department – National Institute of Technology, Kurukshetra

Abstract:

This project explores the creation of a real-time facial emotion recognition system using deep learning and computer vision. The aim is to build a system that can understand and classify human emotions—like happiness, sadness, anger, surprise, fear, and neutrality—just by analyzing facial expressions from a live video feed. Using a webcam, the system detects faces, processes the images, and then predicts the emotion using a trained Convolutional Neural Network (CNN) based on the FER-2013 dataset.

The system is designed with four main components: live video capture, face detection using OpenCV, image preprocessing, and emotion classification using a deep learning model built with TensorFlow and Keras. We've focused on ensuring the system is responsive and works in real-time, even under varying lighting conditions and different facial orientations.

This kind of technology has many real-world uses. It could help teachers understand student engagement in online classes, support mental health tools by identifying signs of distress, or make customer service bots more empathetic and responsive. By combining AI with live video analysis, this project shows how machines can take a step closer to understanding human emotions, ultimately leading to smarter, more emotionally aware systems that improve everyday digital interactions.

Introduction:

Background

Facial expressions are one of the most natural and powerful ways humans communicate emotions. With advancements in artificial intelligence and computer vision, machines are now capable of interpreting these non-verbal cues. Facial emotion recognition systems aim to bridge the gap between human emotional intelligence and machine perception, enabling more natural and empathetic human-computer interactions. Deep learning, particularly Convolutional Neural Networks (CNNs), has revolutionized this domain by significantly improving the accuracy of image-based classification tasks.

Problem Statement

Despite recent advancements, real-time facial emotion recognition remains challenging due to variations in lighting, facial orientation, occlusions, and the subtle nature of some emotions. Many existing systems either lack real-time capability or are not reliable in dynamic environments. There is a need for a system that can accurately detect and classify facial emotions in real time, while being robust, efficient, and easy to integrate into practical applications.

Objectives

- To develop a real-time facial emotion recognition system using deep learning.
- To classify emotions such as happiness, sadness, anger, surprise, fear, and neutrality from live video input.
- To build a responsive and user-friendly interface for visualizing results.
- To ensure reliable performance across varied conditions.

Motivation

The ability for machines to understand human emotions has enormous potential across sectors like education, mental health, customer service, and entertainment. Emotion-aware systems can lead to more personalized user experiences, timely mental health interventions, and smarter digital assistants. This project is driven by the vision of creating an AI system that not only sees but understands.

Overview of the Report

This report begins with a detailed review of related work and the technology stack used. It then explains the dataset and preprocessing steps followed by the model architecture and training process. The implementation and real-time testing methods are presented next. Finally, we discuss the results, challenges faced, and future improvements, concluding with the overall impact and potential applications of the system.

Related work:

Mollahosseini et al. [1] proposed a deep neural network approach for facial expression recognition using the FER-2013 dataset. Their CNN-based model demonstrated significant performance improvements over traditional methods, achieving an accuracy of 66.4% on the test set. Their work highlighted the importance of deeper architectures and large annotated datasets in emotion recognition tasks.

Barsoum et al. [2] introduced an emotion recognition system using deep CNNs trained on FER-2013 and further fine-tuned using a private dataset. They applied data augmentation and dropout techniques to improve generalization and achieved 69.4% test accuracy, outperforming conventional handcrafted-feature-based methods.

Khan et al. [3] developed a real-time facial emotion detection system using a lightweight CNN architecture optimized for embedded devices. They integrated OpenCV for face detection and used TensorFlow Lite for deployment. Their model showed reliable performance with real-time inference speed, emphasizing the feasibility of deep learning models in low-resource settings.

AUTHOR	Dataset	YEAR	Method	Model	Accuracy	Research Gaps
Barsoum et al.	FER-2013 + Private	2016	CNN with augmentation	CNN + Dropout	69.4%	Overfitting due to limited public data
Mollahosseini et al.	FER-2013	2016	Deep CNN	Deep Neural Network	66.4%	Limited to static images, lacks real-time capability
Khan et al.	FER-2013	2020	Lightweight CNN + OpenCV	Optimized CNN	~68%	Limited emotion categories, real-time latency issues

Methodology Used:

The methodology followed in this project involved a systematic process comprising data collection, preprocessing, model development, and evaluation. Each step was designed to ensure the successful development of a robust image classification system.

1. Data Collection

We used the **FER-2013** (Facial Expression Recognition) dataset for this project. It is a widely used benchmark dataset for emotion classification tasks and contains grayscale facial images of size 48x48 pixels. The dataset was already divided into training and test sets:

- The **training set** consisted of **28,709 images**.
- The **test (validation) set** had **7,178 images**.

All images were categorized into seven different emotion classes:
Angry, Disgust, Fear, Happy, Sad, Surprise, and **Neutral**.
Each image belonged to one of these folders, which made it easier to load them using directory-based image generators.

2. Data Preprocessing

To prepare the data for training and evaluation, the following preprocessing steps were carried out:

- **Rescaling**: All pixel values were normalized by dividing by 255 to bring them into the [0,1] range. This helps the model train more efficiently and prevents large gradient updates.
- **Image Loading**: We used Keras's ImageDataGenerator to load and preprocess the images from directories. Two generators were created — one for training and one for validation:
    - The **training generator** was set to shuffle the images and feed them in batches of 64.
    - The **validation generator** was also created in the same way, but with shuffle = False to maintain the order during evaluation.
- **Color Mode**: Since the dataset contains grayscale images, the generators were set to color_mode = 'grayscale'.
- **Target Size**: All images were resized to (48, 48) to ensure consistency in input shape throughout the model.
- **One-Hot Encoding**: The labels were converted into one-hot vectors using class_mode = 'categorical', so the model could treat it as a multi-class classification problem.

As a sanity check, one sample image from each emotion class was visualized using matplotlib to ensure the data was correctly loaded and labeled.

3. Model Development

A Convolutional Neural Network (CNN) was designed from scratch using the Keras Sequential API. The architecture consisted of four convolutional blocks followed by fully connected dense layers.

- Each convolutional block included one or two Conv2D layers followed by **Batch Normalization** and **ReLU** activation.
- **MaxPooling2D** was applied to downsample feature maps and reduce computation.
- **Dropout** was used after each block to prevent overfitting.

The final layers include:

- A Flatten() layer to convert feature maps into a 1D vector.
- Two dense layers (with 256 and 512 units) followed by batch normalization and dropout.
- An output layer with 7 neurons and softmax activation to classify images into one of the seven emotion categories.

The model was compiled using the **Adam optimizer** with a learning rate of 0.0001, and the loss function used was **categorical cross entropy** since this is a multi-class classification task.

The training was performed over **35 epochs**, using the previously defined train and validation datasets. Accuracy was used as the evaluation metric.

4. Model Details

Layer (type)	Output Shape	Param #
conv2d_5 (Conv2D)	(None, 48, 48, 32)	320
conv2d_6 (Conv2D)	(None, 48, 48, 64)	18,496
batch_normalization_6 (BatchNormalization)	(None, 48, 48, 64)	256
max_pooling2d_4 (MaxPooling2D)	(None, 24, 24, 64)	0
dropout_6 (Dropout)	(None, 24, 24, 64)	0
conv2d_7 (Conv2D)	(None, 24, 24, 128)	204,928
batch_normalization_7 (BatchNormalization)	(None, 24, 24, 128)	512
max_pooling2d_5 (MaxPooling2D)	(None, 12, 12, 128)	0
dropout_7 (Dropout)	(None, 12, 12, 128)	0
conv2d_8 (Conv2D)	(None, 12, 12, 512)	590,336
batch_normalization_8 (BatchNormalization)	(None, 12, 12, 512)	2,048

max_pooling2d_6 (MaxPooling2D)	(None, 6, 6, 512)	0
dropout_8 (Dropout)	(None, 6, 6, 512)	0
conv2d_9 (Conv2D)	(None, 6, 6, 256)	1,179,904
batch_normalization_9 (BatchNormalization)	(None, 6, 6, 256)	1,024
max_pooling2d_7 (MaxPooling2D)	(None, 3, 3, 256)	0
dropout_9 (Dropout)	(None, 3, 3, 256)	0
flatten_1 (Flatten)	(None, 2304)	0

dense_3 (Dense)	(None, 256)	590,080
batch_normalization_10 (BatchNormalization)	(None, 256)	1,024
dropout_10 (Dropout)	(None, 256)	0
dense_4 (Dense)	(None, 512)	131,584
batch_normalization_11 (BatchNormalization)	(None, 512)	2,048
dropout_11 (Dropout)	(None, 512)	0
dense_5 (Dense)	(None, 7)	3,591

Total params: 2,726,151 (10.40 MB)

Trainable params: 2,722,695 (10.39 MB)

Non-trainable params: 3,456 (13.50 KB)

5. Real-Time Face Detection and Preprocessing

For real-time emotion detection, OpenCV was used to process the video feed and detect faces. The app captures the webcam stream and analyzes each frame to find faces using OpenCV's Haar cascade classifier. When a face is found, the face region is cropped, turned into grayscale, and resized to 48x48 pixels to match the input requirements of the CNN model. The preprocessed image is then passed to the emotion recognition model for real-time emotion prediction. This process ensures the system works efficiently and accurately while processing video frames with minimal delay.

6. User Interface

The user interface was built using Streamlit, a lightweight Python framework designed for creating web apps with machine learning. It allows users to interact with the emotion recognition system directly from their browser. The UI displays the live webcam feed, shows the predicted emotion in real-time, and includes features like toggling the camera, viewing confidence scores, and updating predictions dynamically. Streamlit made the development process easy and fast, providing a simple yet effective frontend for the application without the complexity of traditional web development.

Experimental Results:

1. Performance Metrics

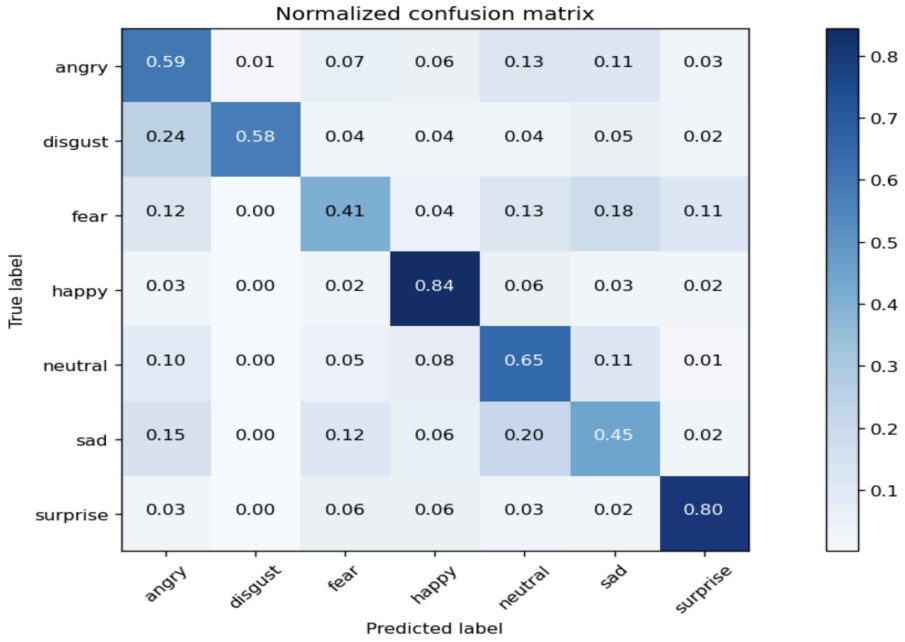
The model was trained on the FER-2013 dataset with grayscale 48x48 pixel facial images across 7 emotion classes. The training process produced the following key results:

Metric	Value
Training Accuracy	80%
Validation Accuracy	65%

These values suggest that the model learned well on the training data but shows some signs of **overfitting**, as there is a considerable drop in accuracy on the validation set.

2. Confusion Matrix

To gain deeper insights into model performance across different emotion classes, a **confusion matrix** was generated. This helped visualize where the model was making the most classification errors.

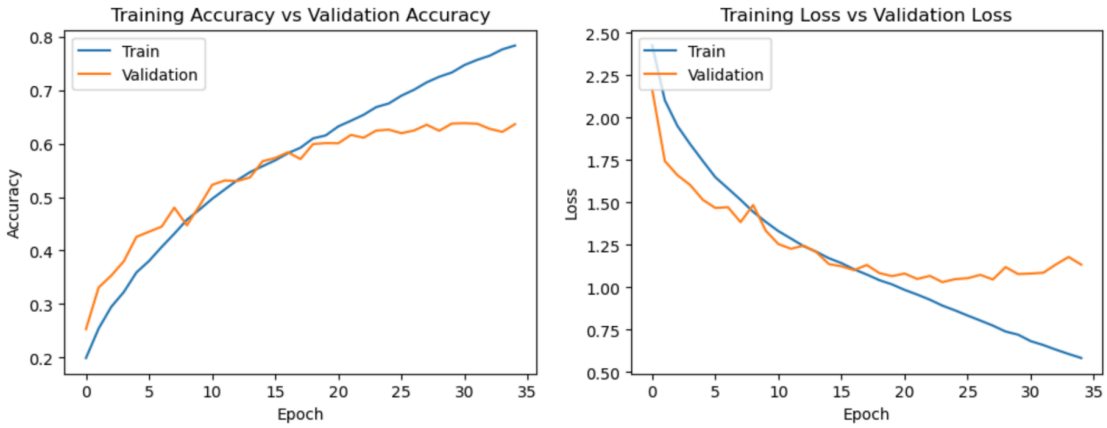


Typical observations from the matrix include:

- High accuracy in detecting "happy" and "surprise" emotions.
- Frequent misclassifications between "fear" and "sad", and between "disgust" and "angry".
- Poor performance in detecting "fear" and “sad”, likely due to limited samples in the dataset.

3. Loss and Accuracy Curves

The training and validation accuracy/loss curves help visualize the learning trend over epochs:



Key observations:

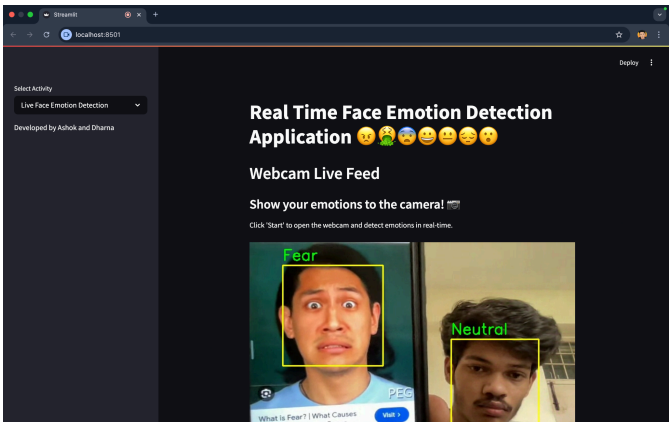
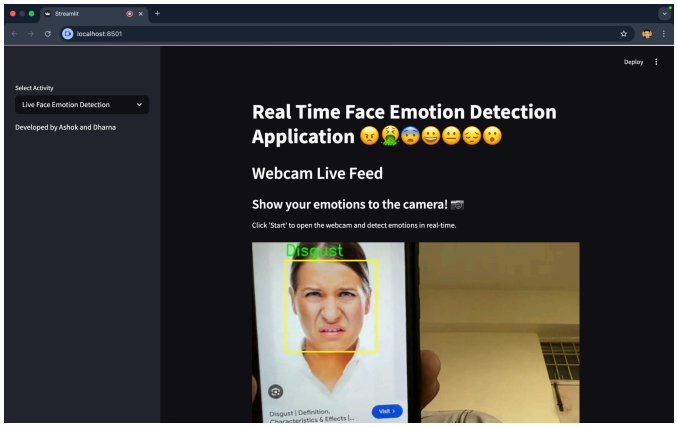
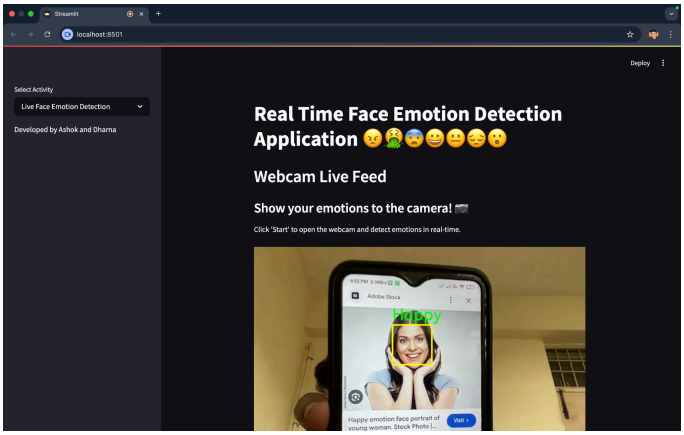
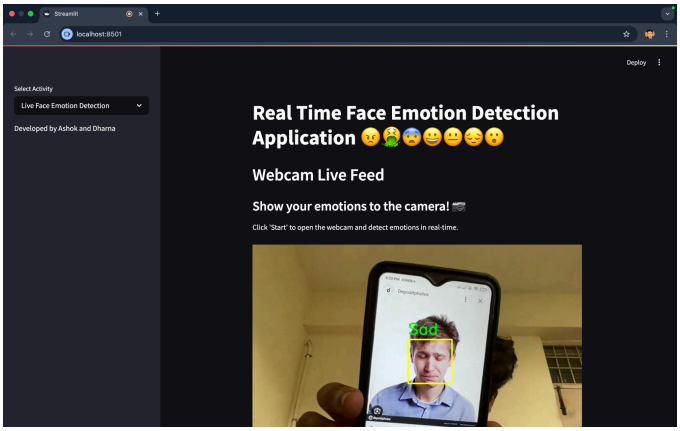
- The training accuracy steadily increased and plateaued around 80%.
- Validation accuracy peaked at around 65%, with occasional fluctuations indicating overfitting.
- Loss curves also diverged after some epochs, supporting the overfitting observation.

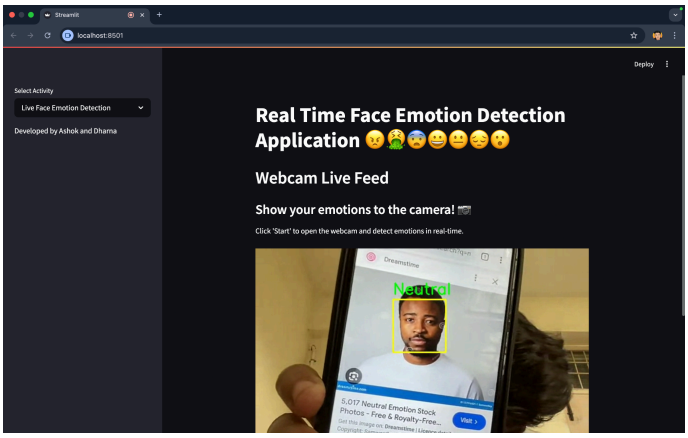
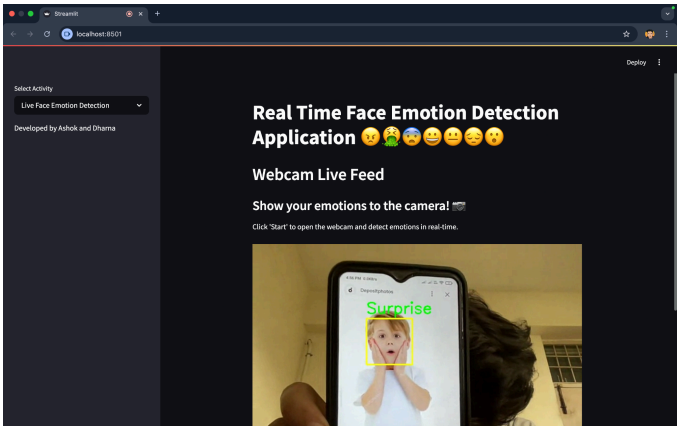
Conclusion:

In this project, we developed a real-time emotion recognition system using OpenCV for face detection and a CNN model for emotion classification. By integrating OpenCV with a webcam feed, we enabled the system to detect and preprocess faces in real-time, providing accurate emotion predictions. The user interface, built with Streamlit, made the system easy to use and interact with, allowing for live webcam interaction and dynamic emotion display. Overall, the project successfully demonstrates how computer vision and machine learning can work together to recognize emotions from video streams in real-time, offering a simple yet effective solution for emotion detection.

The complete source code for this project can be found on Github: https://github.com/ashokdhiman/EmotionClassifierAnd the project is deployed here: https://real-time-facial-emotion-classifier-by-ashok-and-dharnamittal.streamlit.app/

These are some screenshots of the project:

1. Fear	
2. Disgust	
3. Happy	
4. Sad	

5. Neutral	
6. Surprised	
7. Angry