YASH AGGARWAL

HMR INSTITUTE OF TECHNOLOGY AND MANAGEMENT

AFFILIATED TO GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY

B.Tech CSE, 3rd Year

21st MAY 2024

# 5. Develop a real-time emotion detection system that operates on streaming video data and identifies the predominant emotion in each frame

## ✞ INTRODUCTION

Emotion detection from video data is a significant research area in computer vision and affective computing. Real-time emotion detection systems have numerous applications, including human-computer interaction, surveillance, healthcare, and entertainment. This report presents a comprehensive approach to developing a real-time emotion detection system that operates on streaming video data and identifies the predominant emotion in each frame.

## ✞ SYSTEM ARCHITECTURE

The system architecture consists of the following components:

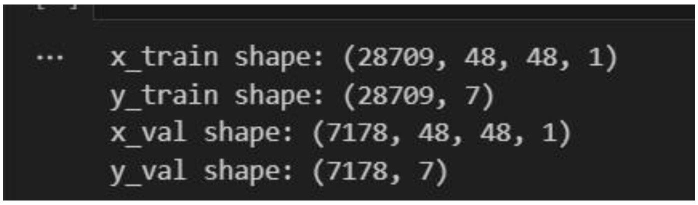
1. **Video Capture Module:** Captures streaming video frames from a camera or video file.
2. **Preprocessing Module:** Processes each frame to enhance quality and normalize the data.
3. **Emotion Detection Module:** Utilizes a trained machine learning model to predict emotions.
4. **Display Module:** Visualizes the detected emotions on the video frames in real-time.

## ✞ TECHNICAL REQUIREMENTS

1. **Hardware:** A computer with a webcam or a pre-recorded video, GPU for model inference.
2. **Software:** Python, OpenCV, TensorFlow/Keras, Dlib.

## ✞ DATA COLLECTION AND PREPROCESSING

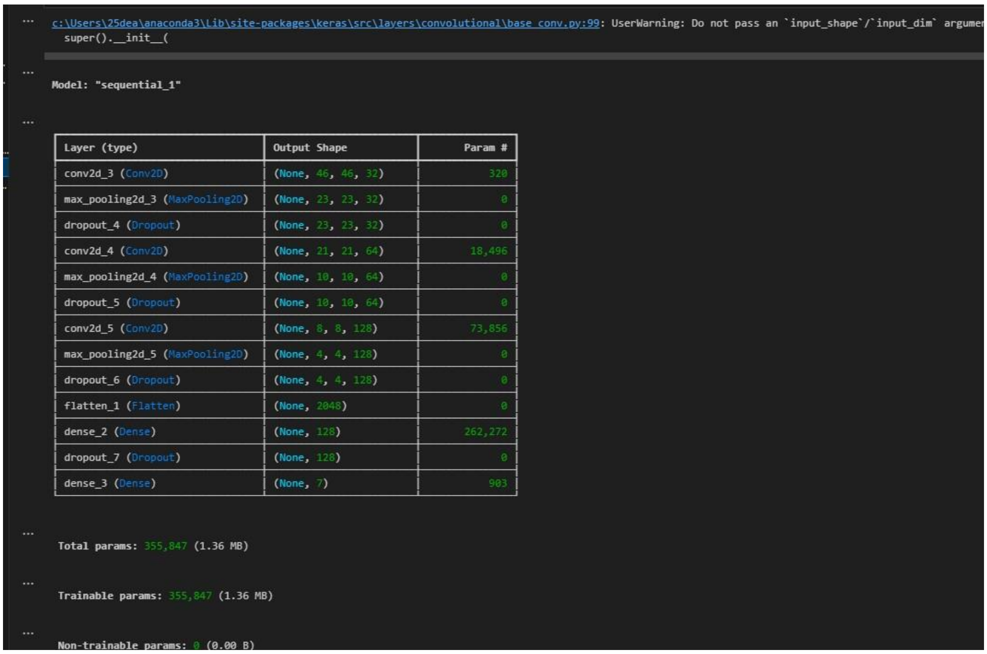
1. **Data Collection:** Use publicly available datasets like FER-2013, CK+, or custom video recordings.
2. **Preprocessing:** A. Convert frames to grayscale
   B. Resize frames to a fixed size(eg: 48X48 pixels). C. Normalize pixel values.



```
···   x_train shape: (28709, 48, 48, 1)
      y_train shape: (28709, 7)
      x_val shape: (7178, 48, 48, 1)
      y_val shape: (7178, 7)
```

## ✞ EMOTION DETECTION MODEL

1. **Model Selection:** Convolutional Neural Networks (CNNs) are effective for image-based emotion detection.
2. **Training:** Use labeled datasets with emotions such as happy, sad, angry, etc. Train the model with techniques like data augmentation to improve robustness.
3. **Performance Metrics:** Accuracy, F1-score, precision, and recall.



```
c:\Users\25dea\anaconda3\Lib\site-packages\keras\src\layers\convolutional\base_conv.py:99: UserWarning: Do not pass an `input_shape`/`input_dim` argumen
  super().__init__(

Model: "sequential_1"

┌─────────────────────────────┬───────────────────────┬───────────┐
│ Layer (type)                │ Output Shape          │   Param # │
├─────────────────────────────┼───────────────────────┼───────────┤
│ conv2d_3 (Conv2D)           │ (None, 46, 46, 32)    │       320 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ max_pooling2d_3 (MaxPooling2D) │ (None, 23, 23, 32) │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dropout_4 (Dropout)         │ (None, 23, 23, 32)    │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ conv2d_4 (Conv2D)           │ (None, 21, 21, 64)    │    18,496 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ max_pooling2d_4 (MaxPooling2D) │ (None, 10, 10, 64) │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dropout_5 (Dropout)         │ (None, 10, 10, 64)    │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ conv2d_5 (Conv2D)           │ (None, 8, 8, 128)     │    73,856 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ max_pooling2d_5 (MaxPooling2D) │ (None, 4, 4, 128)  │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dropout_6 (Dropout)         │ (None, 4, 4, 128)     │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ flatten_1 (Flatten)         │ (None, 2048)          │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dense_2 (Dense)             │ (None, 128)           │   262,272 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dropout_7 (Dropout)         │ (None, 128)           │         0 │
├─────────────────────────────┼───────────────────────┼───────────┤
│ dense_3 (Dense)             │ (None, 7)             │       903 │
└─────────────────────────────┴───────────────────────┴───────────┘

 Total params: 355,847 (1.36 MB)

 Trainable params: 355,847 (1.36 MB)

 Non-trainable params: 0 (0.00 B)
```

## ✞ REAL TIME PROCESSING

1. **Video Streaming Handling:** Use OpenCV to read video frames in real-time.
2. **Integration:** Apply the trained model on each frame to predict emotions.
3. **Optimization:** Use techniques like frame skipping or model quantization to enhance performance.

## ✞ CHALLENGES AND SOLUTIONS

1. **Challenge:** Handling different lighting conditions and facial occlusions.

   **Solution:** Use data augmentation during training and advanced preprocessing techniques.

2. **Challenge:** Ensuring real-time performance.

   **Solution:** Optimize the model and use efficient video processing techniques.

## ✞ TESTING AND EVALUATION

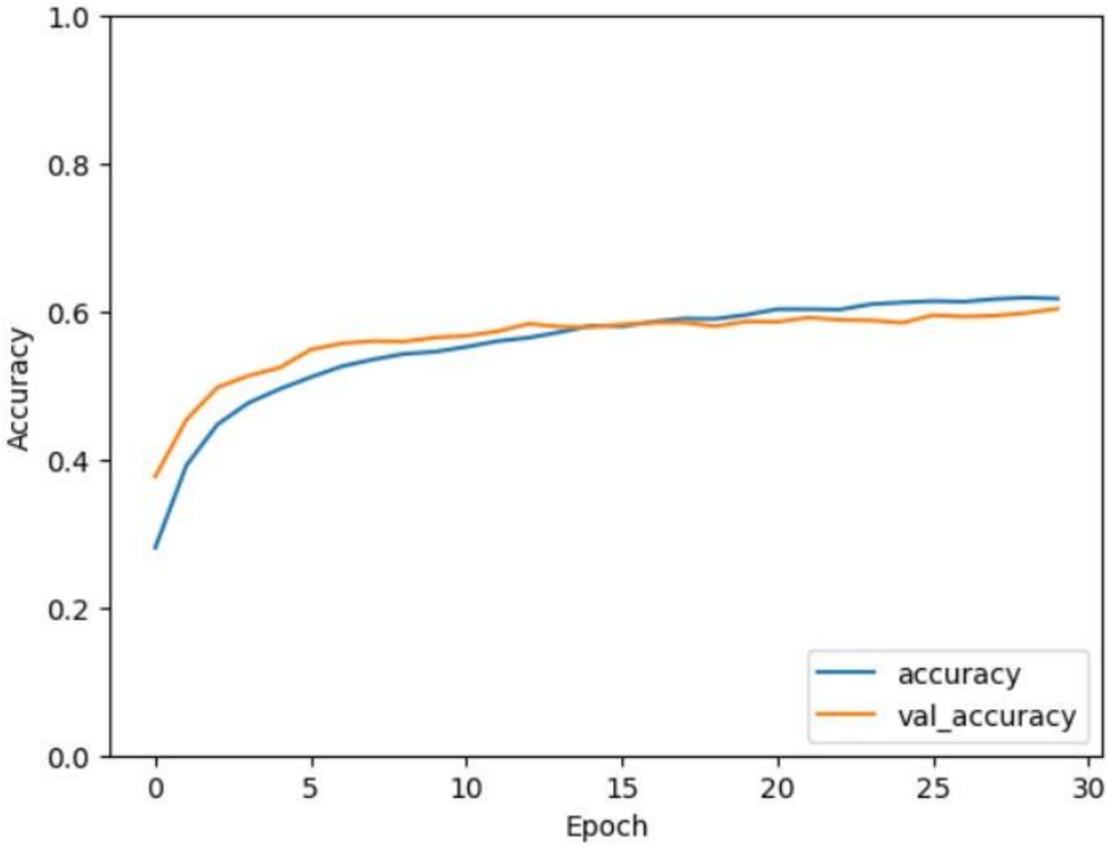
1. **Testing:** Conduct tests using different video streams with various subjects and lighting conditions.

2. **Evaluation:** Measure the accuracy and response time of the system. Ensure that the model generalizes well to new data.

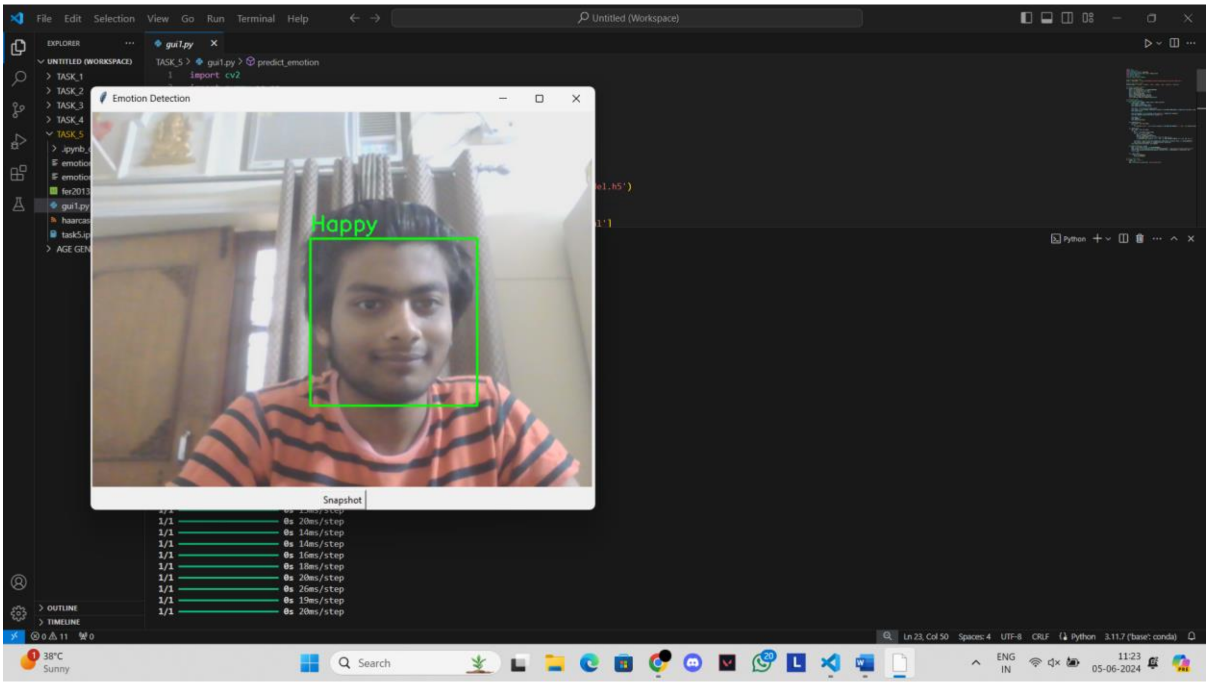
## ✞ CONCLUSION AND FUTURE WORK

This report outlines the development of a real-time emotion detection system. The system captures video frames, preprocesses them, and uses a CNN model to detect emotions. Future work includes improving model accuracy, handling more complex scenarios, and integrating additional features like multi-face detection and emotion tracking over time.

### MODEL ACCURACY:

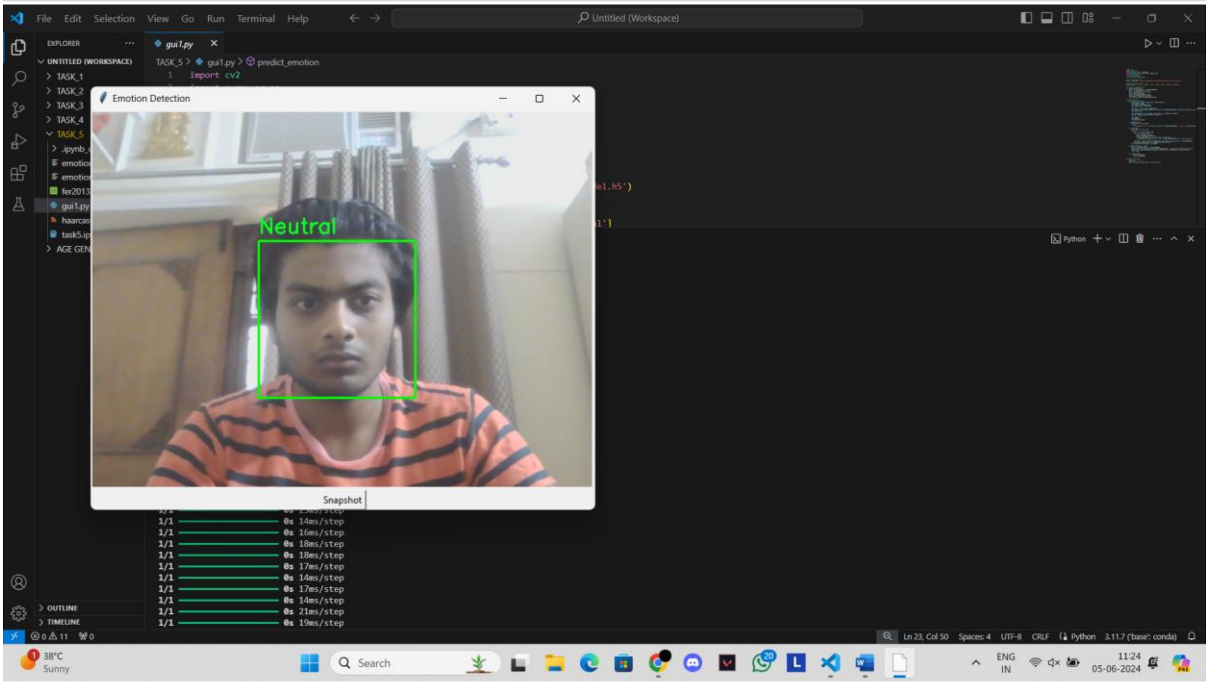


# <u>OUTPUT OF MY CODE FOR TASK 5:</u>



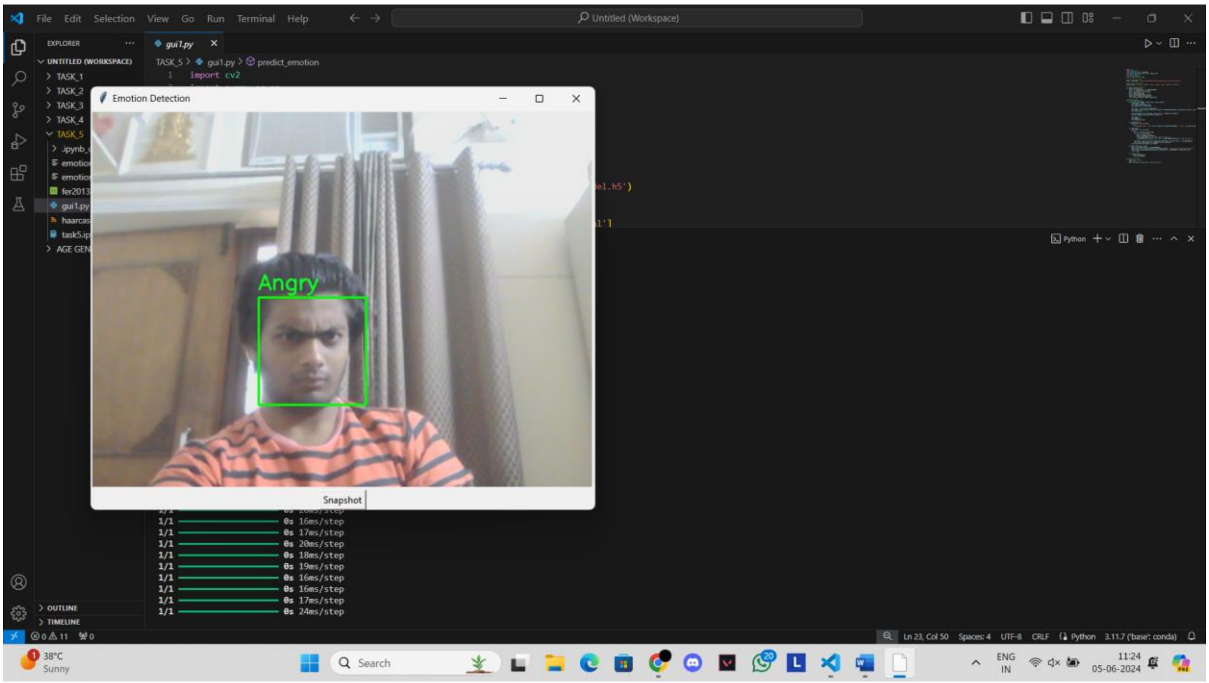
```
import cv2
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
```



```
import cv2
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 14ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 21ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
```



```
import cv2
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 20ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 18ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 19ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 16ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 17ms/step
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step
```