### An Industry Oriented Mini Project (CS704PC)

#### on

### “EMOTION DETECTION IN VIDEOS USING IMAGE PROCESSING TECHNIQUES”

*Submitted*

*in partial fulfillment of the requirements for the award of the degree of* **Bachelor of Technology**

in

##### Computer Science & Engineering

by

##### Ms. B. SANDHYA (18261A0507)

Under the Guidance of

##### Dr. A. NAGESH

Professor



Department of Computer Science and Engineering **MAHATMA GANDHI INSTITUTE OF TECHNOLOGY** GANDIPET, HYDERABAD - 500 075, INDIA.

2021-2022

**MAHATMA GANDHI INSTITUTE OF TECHNOLOGY**

(Affiliated to Jawaharlal Nehru Technological University, Hyderabad) GANDIPET, HYDERABAD - 500 075, Telangana.

**CERTIFICATE**



This is to certify that the project entitled **Emotion Detection in Videos Using Image Processing Techniques** is being submitted by **Ms. B. SANDHYA** bearing Roll No. 18261A0507 in partial fulfillment for the award of **B. Tech in Computer Science and Engineering** to **Jawaharlal Nehru Technological University, Hyderabad** is a record of bonafide work carried out by him under our guidance and supervision.

The results embodied in the project have not been submitted to any other University or Institute for the award of any degree or diploma.

Supervisor and coordinator Head of Department, CSE

##### Dr. A. NAGESH Dr. C. R.K Reddy

Professor Professor

##### External Examiner

**DECLARATION**

This is to certify that the work reported in this project titled **“Emotion Detection in Videos Using Image Processing Techniques”** is a record of work done by me in the Department of **Computer Science & Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.**

No part of the work is copied from books/journals/internet and whatever the portion is taken, the same has been duly referred to in the text. The report is based on the work done entirely by me and not copied from any other source.

##### B. SANDHYA (18261A0507)

i

**ACKNOWLEDGEMENT**

I would like to express my sincere thanks to **Dr. K. Jaya Shankar, Principal, MGIT,** for providing the working facilities in college.

I would like to express my sincere thanks and gratitude to **Dr. C. R. K. Reddy, Professor and HOD,** Department of CSE, MGIT, for all the timely support and valuable suggestions during the period of project.

I am extremely thankful to **Dr. C. R. K. Reddy, Professor and HOD,** and **Ms. B. Prashanthi, Assistant Professor,** and **Ms. D. Deepika, Assistant Professor,** Department of CSE, MGIT, mini project coordinators for their encouragement and support throughout the project.

I am extremely thankful and indebted to my Internal guide **Dr. A. Nagesh Professor,** Department of CSE, for her constant guidance, encouragement and moral support throughout the project.

Finally, I would also like to thank all the faculty and staff of CSE Department who helped us directly or indirectly, for completing this project.

##### B. SANDHYA (18261A0507)

**List of Figures**

|  |  |  |
| --- | --- | --- |
| Figure 3.1 | Flow of logic for processing videos | 7 |
| Figure 3.2 | System architecture for emotion detection | 7 |
| Figure 3.3 | Conversion of Image to Grayscale | 8 |
| Figure 3.4 | Classification of Emotions | 9 |
| Figure 3.5 | The seven types of emotions | 9 |
| Figure 3.6 | Classification of images | 12 |
| Figure 4.1 | Data Flow Diagram | 14 |
| Figure 4.2 | Use Case Diagram | 15 |
| Figure 4.3 | Activity Diagram | 16 |
| Figure 4.4 | Sequence Diagram | 17 |
| Figure 4.5 | Class Diagram | 18 |
| Figure 5.1 | Training(acc) and validation Accuracy(v-acc) | 19 |
| Figure 5.2 | Training(loss) and ValidationLosses(v-loss) | 20 |
| Figure 5.3 | Homepage of Anaconda Navigator | 21 |
| Figure 5.4 | Dataset location | 21 |
| Figure 5.5 | Fear Emotion | 22 |
| Figure 5.6 | Angry Emotion | 22 |
| Figure 5.7 | Neutral Emotion | 23 |
| Figure 5.8 | Sad Emotion | 23 |
| Figure 5.9 | Surprised Emotion | 24 |
| Figure 5.10 | Happy Emotion | 24 |
| Figure 5.11 | Angry | 25 |
| Figure 5.12 | Neutral | 25 |
| Figure 5.13 | Happy | 26 |
| Figure 5.14 | Sad | 26 |
| Figure 5.15 | Surprised | 27 |

**List of Tables**

Table 2.1 Literature Survey 4

**ABSTRACT**

Facial Expressions are considered as one of the channels that convey human emotions. The task of emotion recognition often involves the analysis of human expressions in multi-modal forms such as images or video. The detection of emotion of a person using a camera is useful for various research and analytics purposes.

In this Project, the user's emotion using its facial expressions will be detected. These expressions can be derived from the live feed via the system's camera or any pre-existing image available in the memory. Emotions possessed by humans can be recognized and has a vast scope of study in the computer vision industry upon which several researchers have already been done. The work has been implemented using Python, Open-Source Computer Vision Library (OpenCV) and NumPy. The scanned image is being compared to a training dataset and thus emotion is predicted. The objective of this project is to develop a system which can analyze the image and predict the expression of the person.

Keywords: openCV, Python, Emotion Recognition.

**List of Contents**

|  |  |
| --- | --- |
| Certificate | i |
| Declaration | ii |
| Acknowledgement | iii |
| List of Figures | iv |
| List of Tables | v |
| Abstract | vi |
| **1. Introduction** | **1** |
| 1.1Motivation | 1 |
| 1.2 Problem Statement | 1 |
| 1.3 Existing System | 1 |
| 1.4 Proposed System | 1 |
| 1.5 Requirement Specifications | 2 |
| 1.5.1 Hardware Requirements | 2 |
| 1.5.2 Software Requirements | 2 |
| **2. Literature Survey** | **3** |
| **3. Emotion Detection in Videos Using Image processing Methodology** | **5** |
| 3.1 Computer Vision | 6 |
| 3.1.1 Introduction to Image Processing | 6 |
| 3.2 System Design | 6 |
| 3.3 System Architecture | 7 |
| 3.4 Methodology | 8 |
| 3.5 Modules Description | 9 |
| 3.5.1 Dataset | 9 |
| 3.5.2 Importing Libraries | 10 |
| 3.5.3 Face Detection Using Haar Cascade Classifier | 10 |
| 3.6 Deep Learning Technique | 11 |
| 3.6.1 Classification of Images using MobileNetV2 | 11 |
| **4. UML Diagrams** | **13** |
| 4.1 Data Flow Diagram | 13 |
| 4.2 Use Case Diagram | 14 |
| 4.3 Activity Diagram | 15 |

* 1. [Sequence Diagram 16](#_TOC_250007)
  2. [Class Diagram 17](#_TOC_250006)

1. [Testing and Results 19](#_TOC_250005)
   1. [Modules 19](#_TOC_250004)
   2. [Output screens 21](#_TOC_250003)
   3. [Real time video output screens 25](#_TOC_250002)
2. Conclusion and Future Scope 28

[Bibliography 29](#_TOC_250001)

[Appendix 31](#_TOC_250000)

vi

**1. Introduction**

* 1. **Motivation**

In today’s networked world the need to maintain security of information or physical property is becoming both increasingly important and increasingly difficult. In countries like Nepal the rate of crimes are increasing day by day. No automatic systems are there that can track a person's activity. If we will be able to track Facial expressions of persons automatically then we can find the criminal easily since facial expressions change while doing different activities. So we decided to make a Facial Expression Recognition System.We are interested in this project after we went through a few papers in this area. The papers were published as per their system creation and way of creating the system for accurate and reliable facial expression recognition system.As a result we are highly motivated to develop a system that recognizes facial expression and track one person’s activity.

* 1. **Problem Statement**
     + The growing need for efficiently processing and analyzing the information contained in digital images, videos is a continuous challenge in order to apply image processing and computer vision technologies.
     + According to traditional techniques, manual selection from so many features is impossible.
     + Digital images, videos are commonly processed in a brute force style, by analyzing all the pixels contained in the images, videos, no matter how big and redundant they are.
  2. **Existing System**

A manual examination can be error-prone due to the level of complexities involved in identification of emotions and their features.

* 1. **Proposed System**

So, to resolve such kinds of limitations, computer vision technology is much needed. Computer Vision is a portion of Artificial Intelligence that deals with visual data. With the advent of machine learning and deep learning models, computer systems today can work with digital images and videos to understand and emotionally identify the characteristics of the video’s contents.

### Objectives

1. To develop a facial expression recognition system.
2. To experiment with machine learning algorithms in computer vision fields.
3. To detect emotion thus facilitating Intelligent Human-Computer Interaction.
   1. **Requirements Specification:**

### Hardware Requirements:

* + - * Windows 10 64-bit OS.
      * 12 GB DDR4 RAM.
      * 16 GB NVidia Tesla P100 GPU.
      * 40 GB of Non-Persistent Storage.

### Software Requirements:

* + - * Python 3 is required.
      * Anaconda Navigator.
      * Jupyter Notebook.
  1. **Literature Survey**

Literature [survey](http://www.blurtit.com/q876299.html) is the most important step in the software development process. Before developing the tool, it is necessary to determine the time factor, economy and company strength. Once these things are satisfied, then the next step is to determine which operating system and language can be used for developing the tool. Once the [programmers](http://www.blurtit.com/q876299.html) start building the tool the programmers need a lot of external support. This support can be obtained from senior programmers, from [book](http://www.blurtit.com/q876299.html)s or from websites. Before building the system, the above considerations are taken into account for developing the proposed system.

A literature review is a body of text that aims to review the critical points of current knowledge including substantive findings as well as theoretical and methodological contributions to a particular topic. Literature reviews are [secondary sources](http://en.wikipedia.org/wiki/Secondary_sources), and as such, do not report any new or original experimental work. Also, a literature review can be interpreted as a review of an abstract accomplishment. Most often associated with academic-oriented literature, such as a [thesis](http://en.wikipedia.org/wiki/Thesis), a literature review usually precedes a research proposal and results section. Its main goal is to situate the current study within the body of literature and to provide context for the particular reader.

The given table 2.1 gives details regarding the Emotion Detection Algorithms that were collected from various research papers. It gives information regarding Algorithms used, Name of the author, their merits and demerits and the year published.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.N O** | **NAME OF PAPER** | **AUTHOR** | **METHODOLOGY** | **MERITS** | **DEMERITS** |
| 1. | Emotion Recognition From Facial Expressions and Its Control Using Fuzzy Logic  **YEAR: 2017** | Aruna Chakraborty, Amit Konar  ,Uday Kumar Chakraborty, and Amita Chatterjee. | Fuzzy classifier | It is a robust system where no precise inputs are required. | Not efficient as it is dependent on human intelligence. |
| 2. | Overview on Emotion Recognition System.  **YEAR :2018** | Ashwini Ann Verghese, Jacob P Cherian & Jubilant J Kizhakke Thottam | Active Appearance Model (AAM) | utilizes statistical model | It only uses shape constraints |
| 3 | Emotion Detection Algorithm Using Frontal Face Image.  **YEAR: 2019** | Kim & Joo, Young Hoon & Bae Park, Jin. | Histogram Segmentation Algorithm | Simple and has good antinoise capabilities. | Not applicable for if the object area is not known. |
| 4 | Research on Facial Expression Recognition Based on Neural  Networks. | Zhang and M. Li. | Convolutional neural Network | Very high accuracy in image recognition problems. | Lots of data required |
|  | **YEAR: 2020** |  |  |  |  |

##### TABLE 2.1: Literature Survey

* 1. **Emotion Detection in Videos Using Image Processing Techniques Methodology**
  2. **Computer Vision**

Computer vision provides the ability for the computer to see as humans see. It is the part of computer science that is focused on replicating the intricate parts of the human visual system. It helps identify and process the objects in images through the computer.

Deep learning has delivered superhuman accuracy for image classification, object detection, image restoration, and image segmentation. It uses enormous neural networks to teach machines how to automate the tasks performed by human visual systems. It is a field that aims to gain a deep understanding through digital images or videos.

### Introduction to Image Processing :

It is the core part of computer vision which plays a crucial role in many real-world examples like robotics, self-driving cars, and object detection. Image processing allows us to transform and manipulate thousands of images at a time and extract useful insights from them. It has a wide range of applications in almost every field.

Python is one of the widely used programming languages for this purpose. Its amazing libraries and tools help in achieving the task of image processing very efficiently.

It is a method to perform some important operations on an image. In order to get an enhanced high quality image or to extract the most useful information from that .In this processing input is an image and output may be image or characteristics/features associated with that image.

This is a technique similar to signal processing, in which the input is given as a 2D image, which is a collection of numbers ranging from 0 to 255 which denotes the corresponding pixel value.The method involves converting an image into a 2D Matrix.

It consists of three basic steps:

1. **Scanning the image:** a raw image is acquired which has to be processed. It can be expressed in the form of pixels as stated above. The aim of this step is to extract information which is suitable for computing.
2. **Processing and Enhancing it:** the image is converted into digital form by using a digitizer which samples and quantizes the input signals. The rate of sampling should be high for good resolution and high quantization level for human perception of different shades using different using gray-scale
3. The obtained result describes the property of the image and further classifies the image.
   1. **System Design**

The major requirement for implementing this project using python programming language along with Deep learning and also with python libraries.. The program starts by taking into input the image or video that needs analysis. The FER() constructor is initialized by giving it a face detection classifier (either Opencv Haarcascade).

We then call this constructor to detect emotions function by passing the input object (image or video) to it. The result achieved is an array of emotions with a value mentioned against each. Finally, the ‘top\_emotion’ function can seclude the highest valued emotion of the object and return it. So essentially, the working of any algorithm for both videos and images is the same. The only additional step in processing videos is splitting the video into all of its individual frames and then applying the image processing algorithms to it.

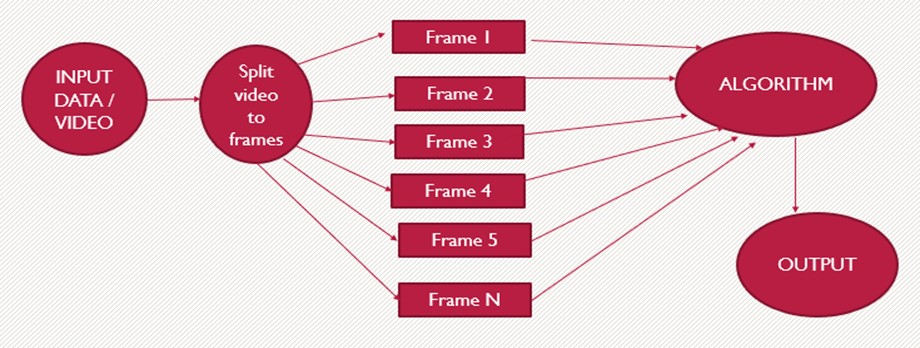
### The flow of logic in Videos:

Although the underlying algorithm is similar for both images and videos ,there are a few key changes that we will follow for videos.

##### Video analyze():

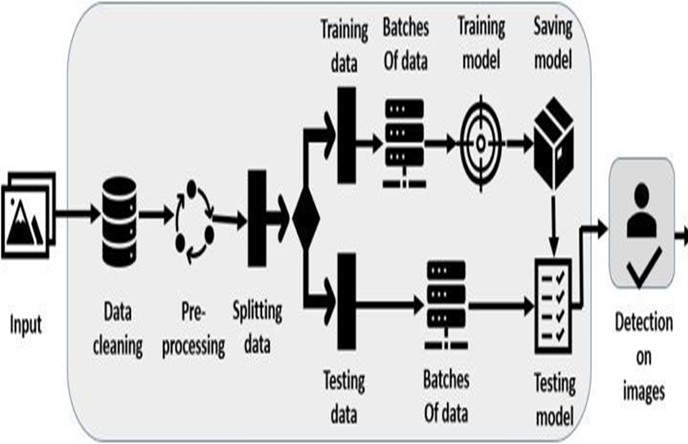
This function is responsible for extracting the individual image frames from a video and then analyzing those independently. Each frame analyzed by this function is stored as a separate image by the algorithm in the root directory folder where the code is running. Also, this function later creates a replica of the original video by placing a box around the face and showing live emotions within the video. We then create a Pandas DataFrame from these analyzed values and plot this dataframe using matplotlib. In this plot, we can see every emotion plotted against time. We can further analyze this dataframe by taking

individual emotion values that were recognized by the model and finding which sentiment was dominant across the entire video. This way, we can work on videos by extracting individual image frames and analyzing them. This process is displayed in the figure 3.1 below that shows how an additional step gets added for processing videos.



##### Figure 3.1 Flow of logic for processing videos.

* 1. **System Architecture**



##### Figure 3.2 System architecture for emotion detection.

As shown in the above figure 3.2 , the first data is retrieved from datasets. The data has to be prepared for training. So, the data is undergone into a pre-processing step. Now, the model is constructed with deep

learning algorithms. Now, the model is trained with a dataset. The model is validated and tuned to improve the classification accuracy of the model.

* 1. **Methodology**

For implementing this project four basic steps are required to be performed.

1. **Preprocessing :** Image pre-processing includes the removal of noise and normalization against the variation of pixel position or brightness. Most preprocessing steps that are implemented are – Reduce noise, Convert The Image To Binary/Grayscale, Pixel Brightness Transformation. and Geometric Transformation.



##### Figure 3.3 conversion of Image to Grayscale.

The above figure 3.3 shows the images are converted into different variations to remove the noise and each are varying in the pixel positions.

1. **Facial Feature Extraction :** Selection of the feature vector is the most important part in a pattern classification problem. The image of the face after preprocessing is then used for extracting the importantfeatures. The inherent problems related to image classification include the scale, pose translation and variations in illumination level.
2. **Face Detection and Classification :** Face Detection is useful in detection of facial image. Face Detection is carried out in a training dataset using a Haar classifier and implemented through Opencv. Haar-like features encode the difference in average intensity indifferent parts of the image and consist of black and white connected rectangles in which the value of the feature is the difference of sum of pixel values in black and white region.

**Emotion Classification** : In this step of classification, the algorithm attempts to classify the given faces portraying one of the seven basic emotions.



##### Fig 3.4 Classification of Emotions.

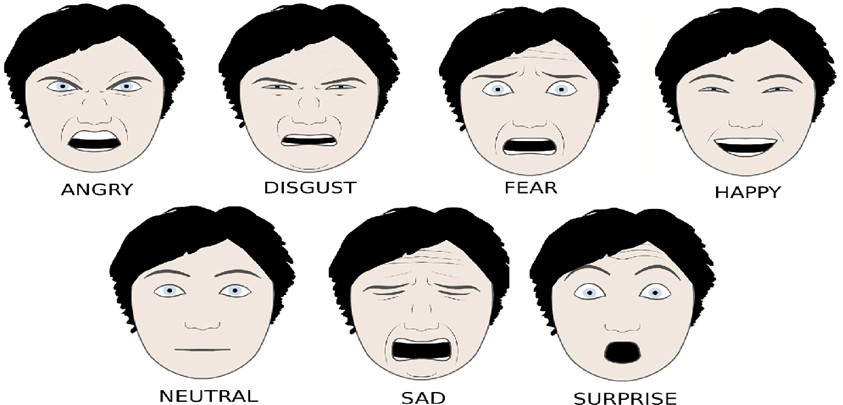
The above Figure 3.4 shows the different types of emotion a human can have in their real life.

* 1. **Modules Description :**

### Dataset

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image.

The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral) which is shown in Figure 3.5 The training set consists of 9,000 examples and the public test set consists of 2585 examples. These are going to be trained and validated against results.



##### Figure 3.5 The seven types of emotions

* + 1. **Importing Libraries**

**OpenCV** There are some predefined packages and libraries in python as part of Computer Vision which can make our life quite simple and OpenCV is one of them. It helps us develop a system that can process images and real-time video using computer vision. OpenCV (Open-Source Computer Vision Library) is an open-source computer vision and machine learning software library which is easy to import in Python. We will be using the HaarCascade algorithm in the model. It is a machine learning-based approach where a cascade function is trained using a whole of positive and negative images. It is then used to detect objects in other images.

**Numpy** Numpy is the fundamental package for scientific computing with Python. As it is used to divide the given image from a construction site into n dimensional objects such that it can be easily compared with the dataset.

**Matplotlib** Matplotlib is a Python 2D plotting library which is used to plot a given image in a number of small images and it will check with the dataset and it will produce publication quality figures in a variety of hard copy formats and interactive environments across platforms

**Keras** Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Keras contains numerous implementations of commonly used neural network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel. Keras allows users to productize deep models on smartphones (iOS and Android), on the web, or on the Java Virtual Machine. It also allows use of distributed training of deep learning models on clusters of Graphics Processing Units (GPU).

**TensorFlow** TensorFlow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of TensorFlow.

* + 1. **Face detection using Haar Cascade Classifier:**

Haar classifiers are the classifiers that were used in the first real-time face detector. It is a machine learning object detection program that identifies objects in an image and video.Haar Cascade classifier is

used to detect faces and classify the emotions in this project. It is significantly faster in speed and high-accuracy object detection algorithm.

* 1. **Deep Learning Technique**

### MobileNetV2

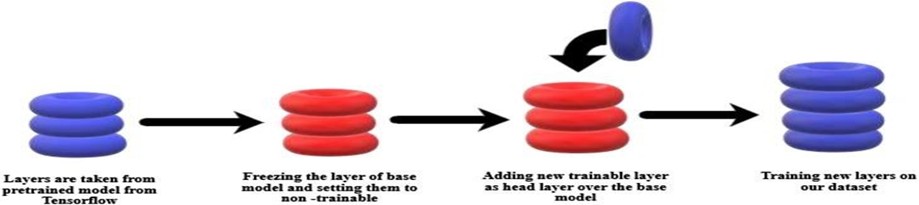
MobileNets are a family of neural network architectures released by Google to be used on machines with limited computing power, like mobile devices. They strive to provide state of the art accuracy, while requiring as little memory and computing power as possible. This makes them a very fast family of networks to use for image processing.

MobileNets achieve this performance by reducing dramatically the number of learnable parameters, which also makes them faster and easier to train compared to more traditional networks.It is based on an inverted residual structure where the residual connections are between the bottleneck layers.

The intermediate expansion layer uses lightweight depth-wise convolutions to filter features as a source of non-linearity. As a whole, the architecture of MobileNetV2 contains the initial fully convolution layer with 32 filters, followed by 19 residual bottleneck layers.The typical MobilenetV2 architecture has as many layers listed below, In Pytorch we can use the models library in TorchVision to create the MobileNetV2 model instead of defining/building our own model. The weights of each layer in the model are predefined based on the ImageNet dataset. The weights indicate the padding, strides, kernel size, input channels and output channels.Based on ImageNet dataset MobileNetV2 outperforms MobileNet and ShuffleNet with comparable model size and computational cost. And also it will perform well for the smaller dataset.

* + 1. **Classification of images using MobileNetV2**

MobileNetV2 is a Deep Neural Network that has been deployed for the classification problem. Pre-trained weights of ImageNet were loaded from TensorFlow. Then the base layers are frozen to avoid impairment of already learned features. Then new trainable layers are added, and these layers are trained on the collected dataset so that it can determine the features to classify a face wearing a mask from a face not wearing a mask. Then the model is fine-tuned, and then the weights are saved. Using pre- trained models helps avoid unnecessary computational costs and helps in taking advantage of already biased weights without losing already learned features.This plan of action is shown in figure 3.6 .



##### Figure 3.6 Classification of images.

1. **UML DIAGRAMS**

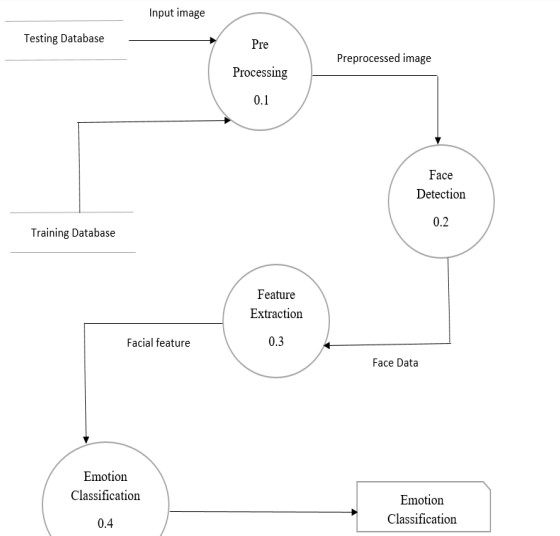
Unified Modeling Language is a standard language for writing software blueprints. It can be used to visualize, specify, construct, and document the artifacts of a software-intensive system. Modeling is a proven and well-accepted engineering technique.

* 1. **Data flow diagram**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an overview of the system without going into great detail, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design). A DFD shows what kind of information will be input to and output from the system, how the data will advance through the system, and where the data will be stored. It does not show information about process timing or whether processes will operate in sequence or in parallel, unlike a traditional structured flowchart which focuses on control flow, or a UML activity workflow diagram, which presents both control and data flows as a unified model.

Data flow diagrams are also known as bubble charts. DFD is a designing tool used in the top down approach to Systems Design.

Figure 4.1 shows that whenever the input image is given, it is preprocessed to remove the noise and change the pixel's position and by using a classifier it detects the features of the image. Then those features are extracted. By feature extraction one can classify the type of emotion by using Emotion classification algorithms.The detected emotion will be shown as the output.It maintains Training and testing Dataset. Training Database is used for training the model and Testing Database for Validating the inputs.



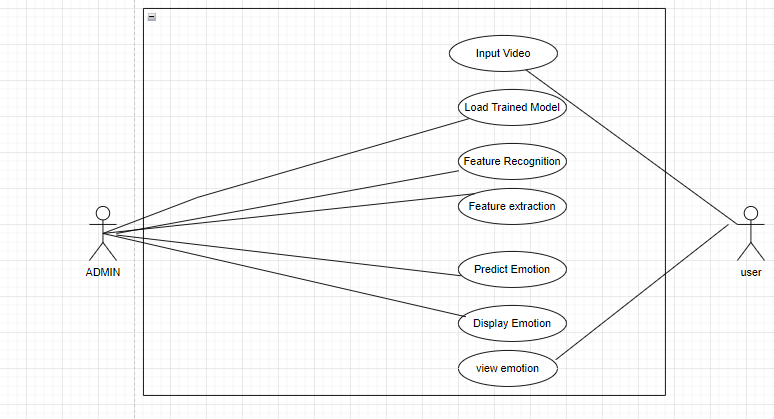
##### Figure 4.1 Data flow diagram for emotion detection

* 1. **Use Case Diagram :**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted. A use case diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. It models the tasks, services, and functions required by a system/subsystem of an application. It depicts the high-level functionality of a system and also tells how the user handles a system.

As shown in Figure 4.2 the role of the system administrator is to predict the output. The data set used should allow operations like Loading model, features recognition, features extraction. The other actor is the user who gives the input and sees the output.

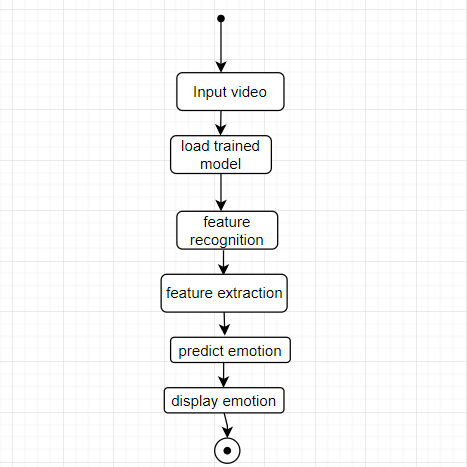
The administrator divides the data set into training and testing phases and modeling is performed iteratively in order to get the best model for deployment.



##### Figure 4.2 Use Case Diagram

* 1. **Activity Diagram**

Activity diagram is another important behavioral diagram in UML diagram to describe dynamic aspects of the systems. Activity diagram is essentially an advanced version of the flow chart that is modeling the flow from one activity to another. It is used to show the steps of each method and also show the activity of the project. It can be used to represent the path of execution of a project. As shown in figure 4.3 we load the Data, then training and loading the model is done followed by feature recognition and feature extraction. Finally based on those outcomes, the emotion is classified and predicted.

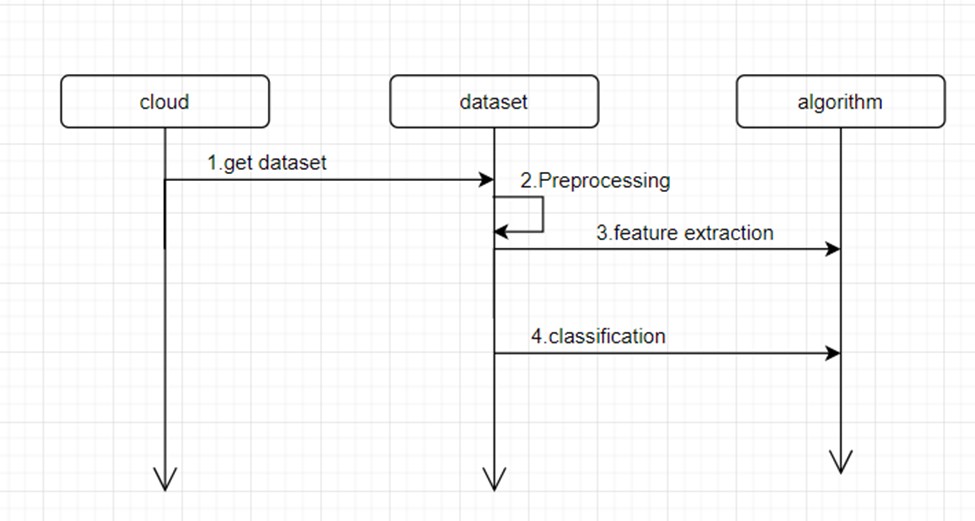


##### Figure 4.3 Activity Diagram

## Sequence Diagram

A sequence diagram or system sequence diagram shows object interactions arranged in time sequence in the field of software engineering. It depicts the objects involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. The Figure

4.3 shows the flow of execution of various processes wiz. acquiring datasets, preprocessing, feature extraction, classification, predicting results and accuracy.

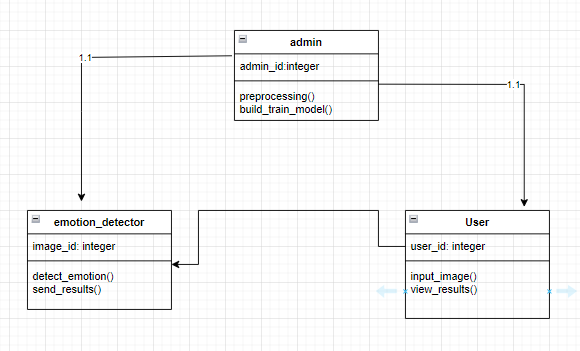


##### Figure 4.4 Sequence diagram

## 4.5 Class Diagram

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes.

Figure 4.4 shows each class contains a set of attributes and functions, there are three main classes to implement this algorithm. Out of which User class and Admin class would provide the access to the respective person.



##### Figure 4.5 : class diagram

# Testing and Results

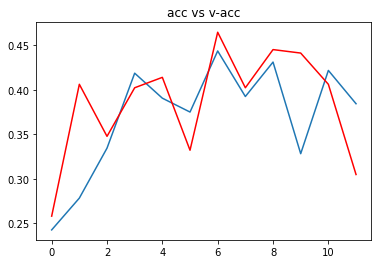
Testing is a very important module in software development to verify, validate and provide quality and service for different components of software. It is used to minimize the risks by efficient use of resources in the development life cycle. This module can be employed at any point of the development process. It is efficient for the testing phase to be implemented at initial level to lower down the risks of defects and failures. Software is tested and implemented in various conditions and environments to examine different aspects of software.

## Modules

* + - **emotion\_classifier.ipynb-** It contains all the steps, processes and results of training, validating and testing the classifier.
* **realtime\_video.py-** it classifies the emotions in a live video.

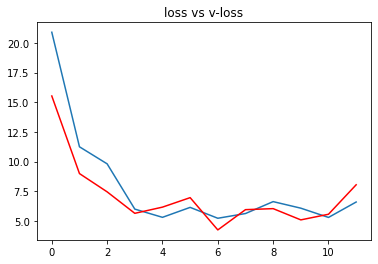
**Running the classifier** Use the realtime\_video.py for running the script in Terminal, Powershell or Command prompt and the command used is **python realtime\_video.py**

Inorder to test a dataset with algorithms we have to test the dataset with the same algorithms and check the accuracy of each algorithm and predicted values of each algorithm. The algorithm which gives high accuracy and best predicted value in testing is the best algorithm which should be used further in testing.



##### Figure 5.1 : Training(acc) and validation Accuracy(v-acc)

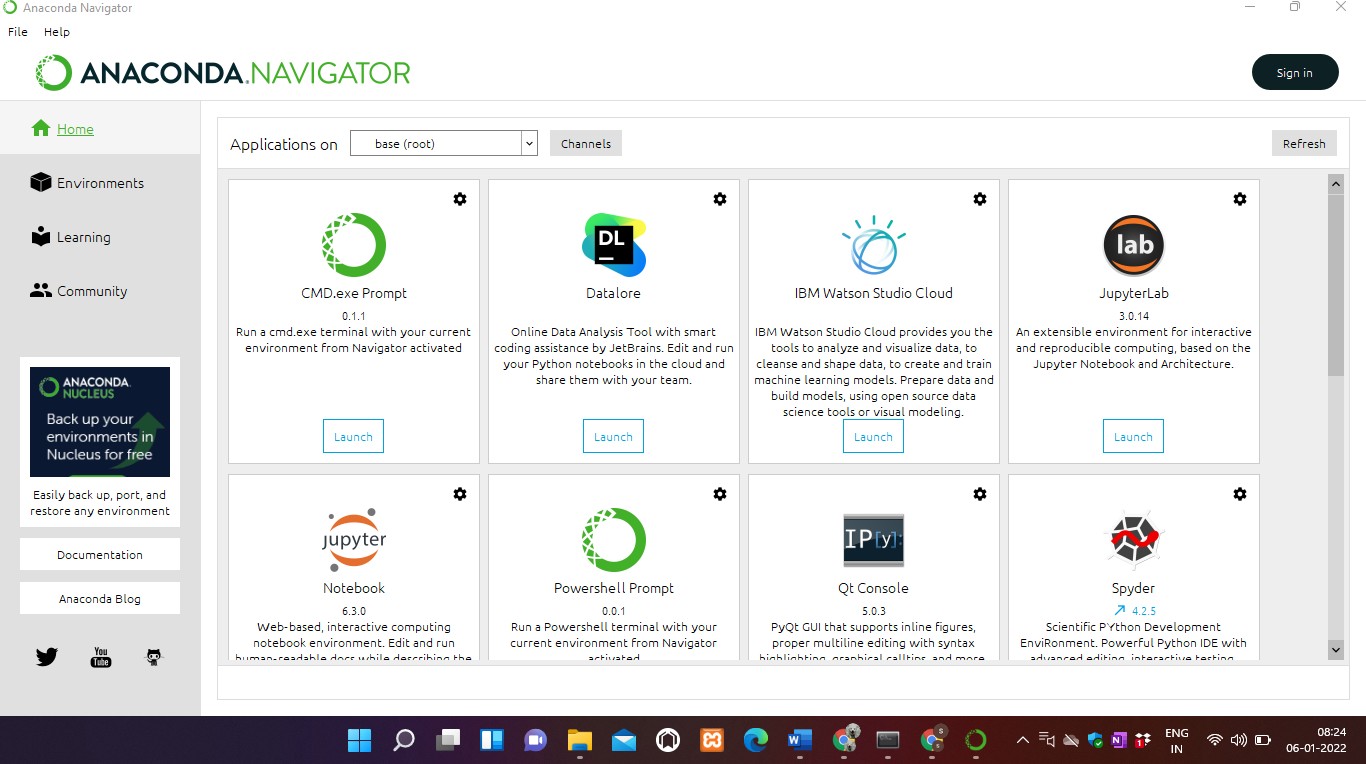
Figure 5.1 shows the Training accuracy indicated in blue line and Validation accuracy in red line. It shows values for each epoch from 0 to 10. Here, the x-axis represents epochs while the y-axis represents accuracy.



##### Figure 5.2 Training(loss) and Validation Losses(v-loss)

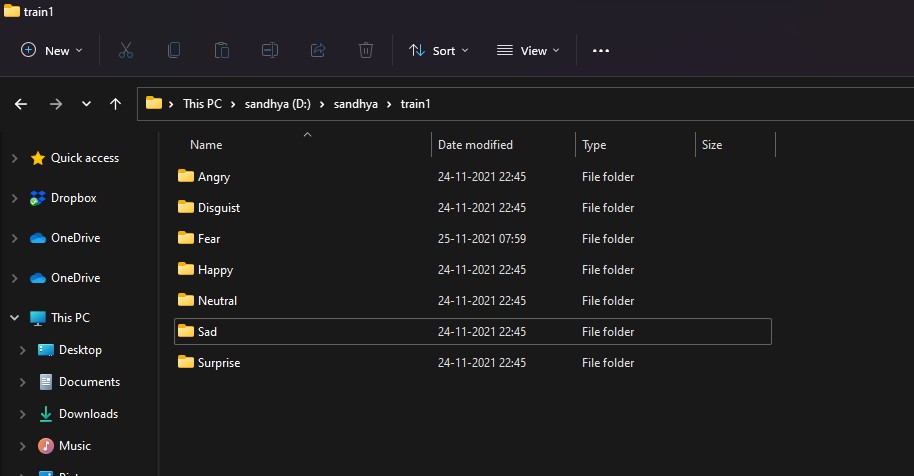
The above figure 5.2 shows Accuracy and loss for each epoch from 0 to 10. Here we can see that there is a gradual decrease in the Error rate in both training and validation. It states that there may be new classes introduced to the network so by getting the new thing as an input it is unable to find the result and the error has increased compared to the previous epochs. But as the training goes on the training loss decreases to 0.086 and Accuracy Increased to 96.67% .

## Output screens:



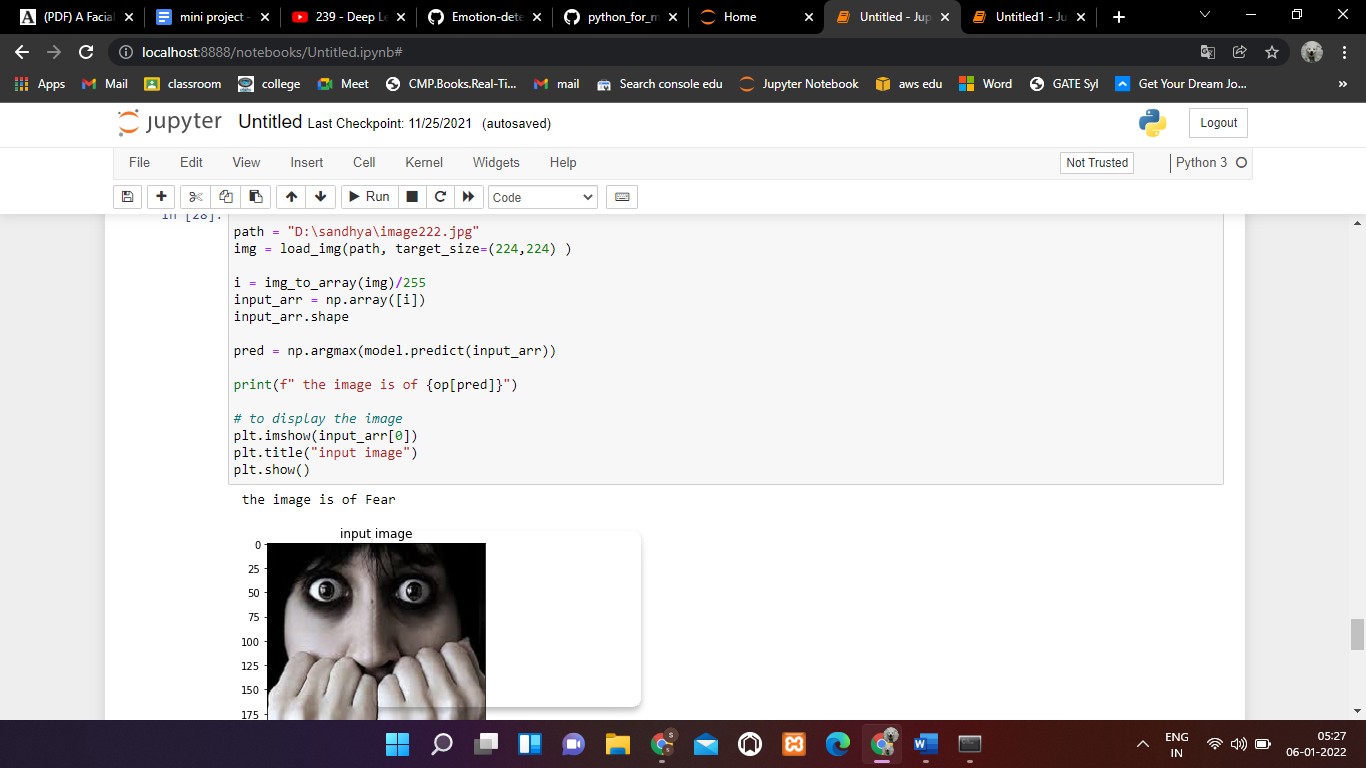
##### Figure 5.3 Homepage of Anaconda Navigator.

##### Figure 5.3 shows the Homepage of Anaconda Navigator from which the Jupyter Notebook can be used for projects.



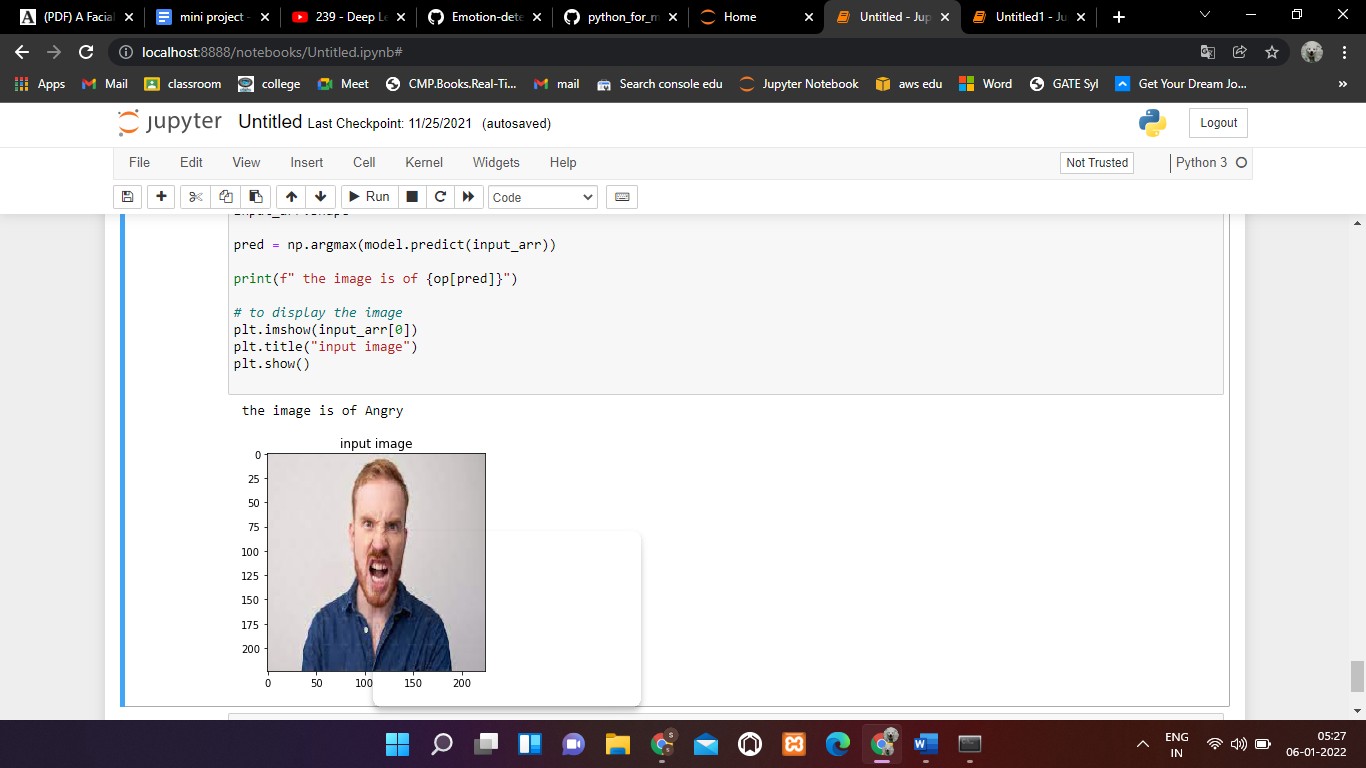
##### Figure 5.4 Dataset location

The figure 5.4 shows the dataset that is used for validating and training the model.

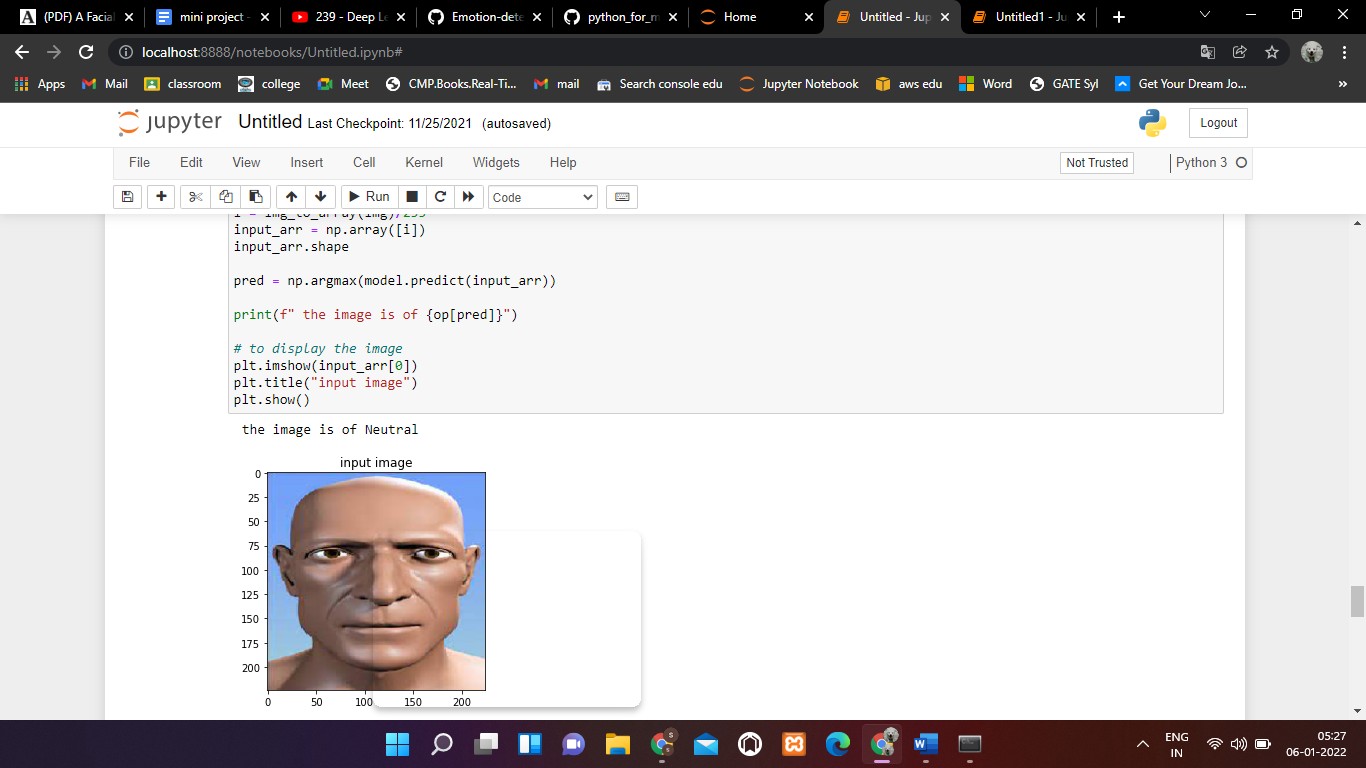


##### Figure 5.5 Fear Emotion

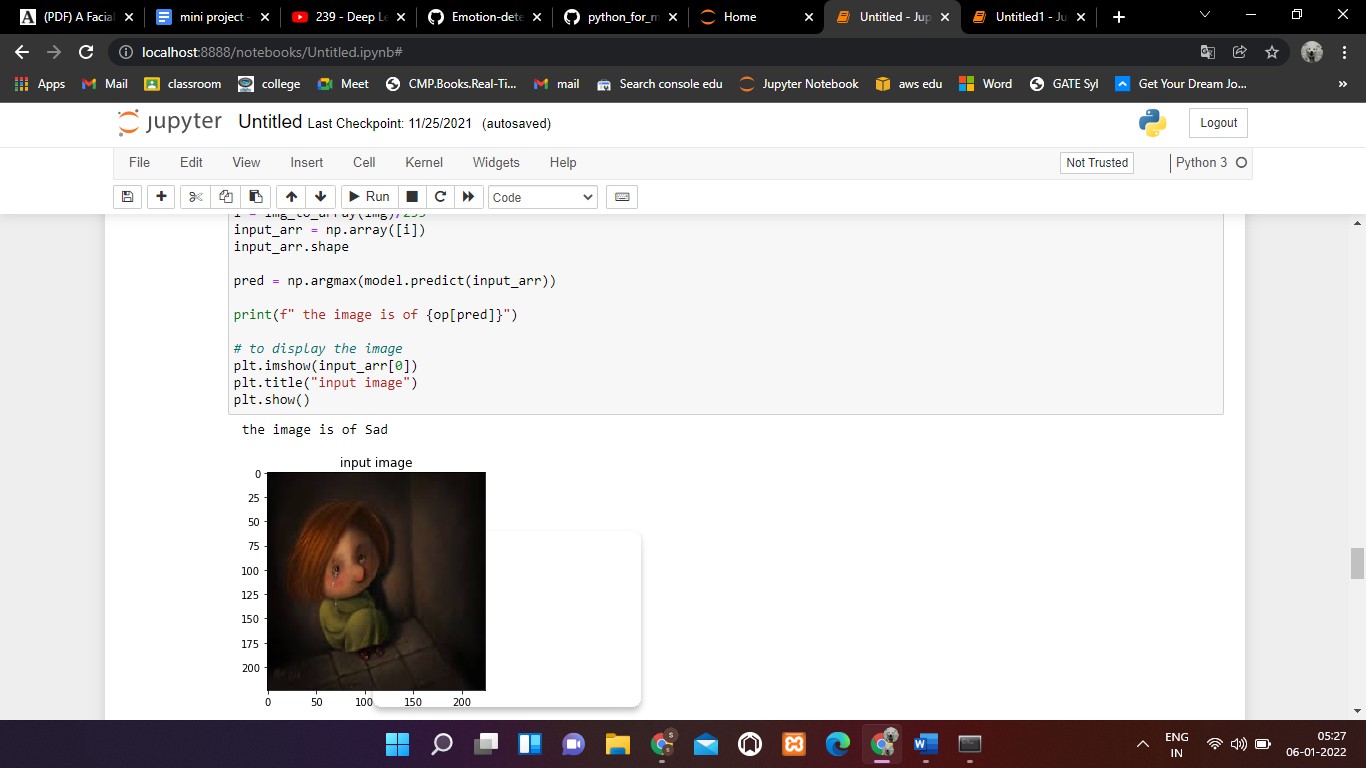
In the figure 5.5, the emotion has been detected as Fear.



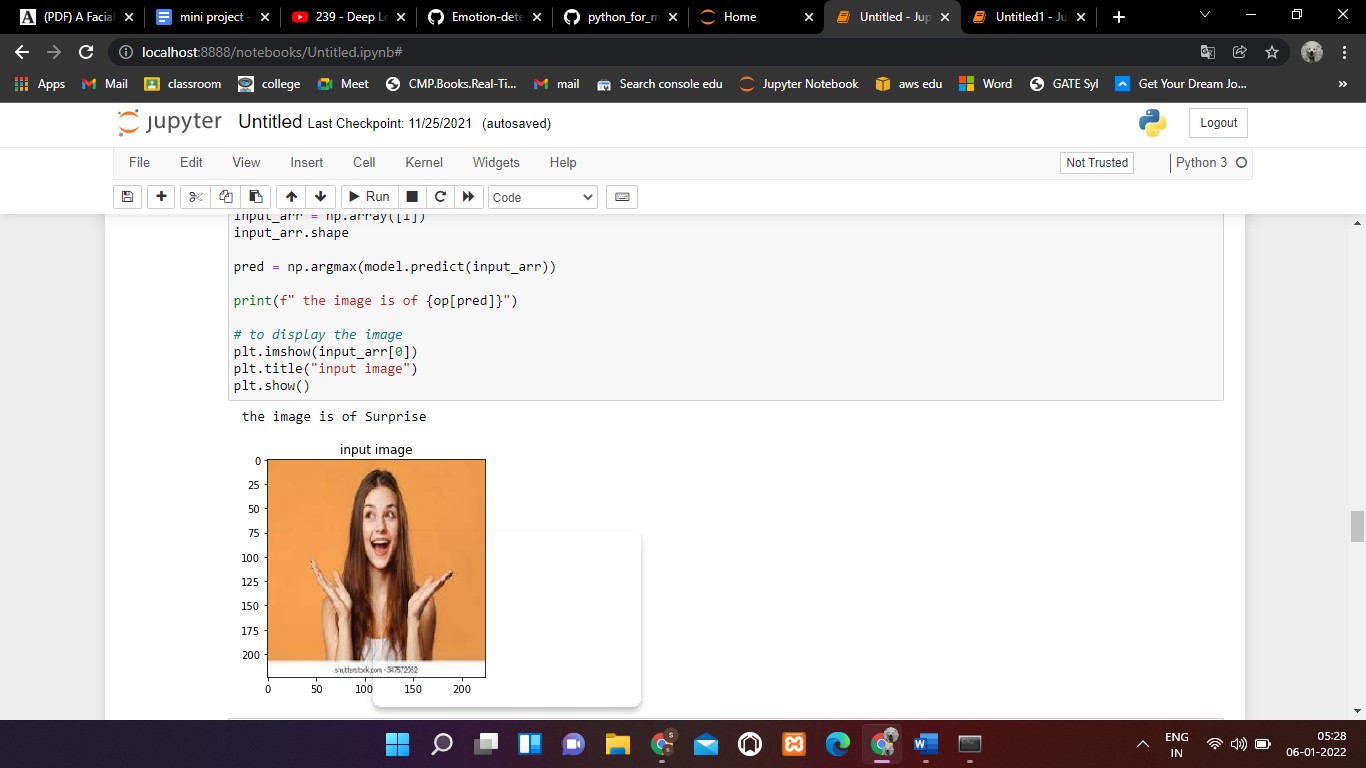
**Figure 5.6 Angry Emotion .** In this figure the emotion has been detected as Angry.



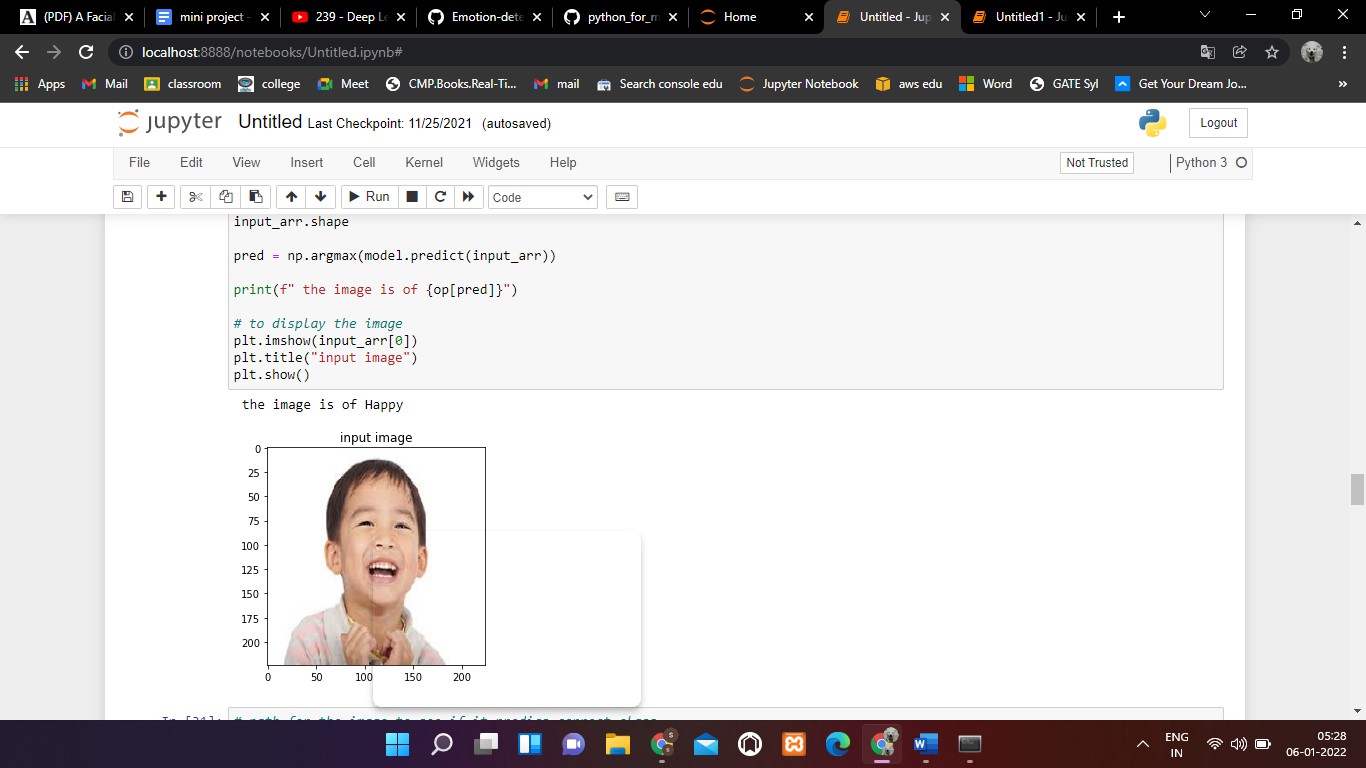
**Figure 5.7 Neutral Emotion .** In this figure the emotion has been detected as neutral.



**Figure 5.8 Sad Emotion .** In this figure the emotion has been detected as sad



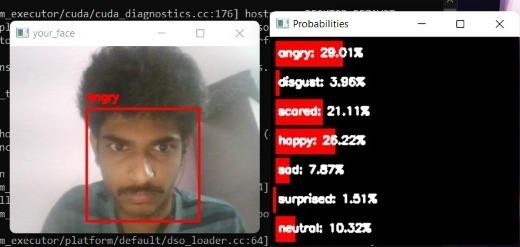
**Figure 5.9 Surprise Emotion** The emotion has been detected as Surprise



**Figure 5.10 Happy Emotion** .The emotion has been detected as Happy

## Real time Video output Screens

By running the python file **realtime\_video.py** the following outputs are generated.In this live video detection we consider the probabilities of the emotions. The one with maximum Probability is taken as output.



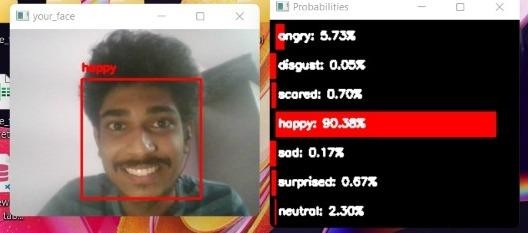
##### Figure 5.11 : Angry

In the figure 5.11, the one with maximum Probability is taken as output that is “angry”.



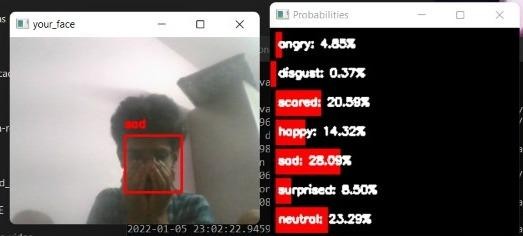
##### Figure 5.12 Neutral

In the figure 5.12, the one with maximum Probability is taken as output that is “ neutral”.



##### Figure 5.13 Happy

In the figure 5.13, the one with maximum Probability is taken as output that is “happy”.



##### Figure 5.14 :Sad

In the figure 5.14, the one with maximum Probability is taken as output that is “sad”.



##### Figure 5.15: Surprised

In the figure 5.15, the one with maximum Probability is taken as output that is “surprised”.

**CONCLUSION AND FUTURE SCOPE**

**Conclusion**

This project proposes an approach for recognizing the category of facial expressions. Face Detection and Extraction of expressions from facial images is useful in many applications, such as robotics vision, video surveillance, digital cameras, security and human-computer interaction. This project’s objective was to develop a facial expression recognition system implementing the computer visions and enhancing the advanced feature extraction and classification in face expression recognition.

In this project, seven different facial expressions of different persons’ images from different datasets have been analyzed. This project involves facial expression preprocessing of captured facial images followed by feature extraction and classification of facial expressions based on training of datasets of facial images based on openCV python and MobileNetV2.

**Future Scope**

Face expression recognition systems have improved a lot over the past decade. The focus has definitely shifted from posed expression recognition to spontaneous expression recognition. Promising results can be obtained under face registration errors, fast processing time, and high correct recognition rate (CRR) and significant performance improvements can be obtained in our system. System is fully automatic and has the capability to work with images feed. It is able to recognize spontaneous expressions. Our System can be used in Digital Cameras wherein the image can be captured only when the person smiles. In security systems which can identify a person, in any form of expression he presents himself. Rooms in homes can set the lights, television to a person’s taste when they enter the room. Doctors can use the system to understand the intensity of pain or illness of a deaf patient. Our system can be used to detect and track a user’s state of mind, and in mini-marts, shopping centers to view the feedback of the customers to enhance the business etc.

# BIBLIOGRAPHY

1. Bettadapura, V. (2012). Face expression recognition and analysis: the state of theart. arXiv preprint arXiv:1203.6722 .
2. Shan, C., Gong, S., & McOwan, P. W. (2005, September). Robust facial expression recognition using local binary patterns. In Image Processing, 2005. ICIP 2005. IEEE International Conference on (Vol. 2, pp. II-370). IEEE.
3. Bhatt, M., Drashti, H., Rathod, M., Kirit, R., Agravat, M., & Shardul, J. (2014). AStudyof Local Binary Pattern Method for Facial Expression Detection. arXiv preprint arXiv:1405.6130 .
4. Chen, J., Chen, Z., Chi, Z., & Fu, H. (2014, August). Facial expression recognition based on facial components detection and hog features. In International Workshops on Electrical and Computer Engineering Subfields (pp.884-888).
5. Ahmed, F., Bari, H., & Hossain, E. (2014). Person-independent facial expression recognition based on compound local binary pattern (CLBP). Int. Arab J. Inf.Technol. , 11 (2), 195-203.
6. Happy, S. L., George, A., & Routray, A. (2012, December). A real time facial expression classification system using Local Binary Patterns. In Intelligent HumanComputer Interaction (IHCI), 2012 4th International Conference on (pp. 1-5).IEEE.
7. Zhang, S., Zhao, X., & Lei, B. (2012). Facial expression recognition based on local binary patterns and local fisher discriminant analysis. WSEAS Trans. Signal Process , 8 (1), 21-31. 30
8. Chibelushi, C. C., & Bourel, F. (2003). Facial expression recognition: A brief tutorial overview. CVonline: On-Line Compendium of Computer Vision, 9.
9. Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006, December). Beyond Accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In Australasian Joint Conference on Artificial Intelligence (pp. 1015-1021). Springer Berlin Heidelberg.
10. Michel, P., & El Kaliouby, R. (2005). Facial expression recognition using support vector machines. In The 10th International Conference on Human-ComputerInteraction, Crete, Greece.
11. Michel, P., & El Kaliouby, R. (2003, November). Real time facial expression recognition in video using support vector machines. In Proceedings of the 5thinternational conference on Multimodal interfaces (pp. 258-264). ACM.

# APPENDIX

##### emotion\_detection.ipynb:

##### #CODE FOR BUILDING EMOTION DETECTION\_MODEL #IMPORT NECESSARY PACKAGES

import numpy as np import pandas as pd

import matplotlib.pyplot as plt **.**

from keras.layers import Flatten, Dense from keras.models import Model

from keras.preprocessing.image import ImageDataGenerator , img\_to\_array, load\_img from keras.applications.mobilenet import MobileNet, preprocess\_input

from keras.losses import categorical\_crossentropy

##### # Working with pre trained model

base\_model **=** MobileNet( input\_shape**=**(224,224,3), include\_top**= False** )

**for** layer **in** base\_model**.**layers:

layer**.**trainable **= False**

x **=** Flatten()(base\_model**.**output)

x **=** Dense(units**=**7 , activation**=**'softmax' )(x)

##### # creating our model.

model **=** Model(base\_model**.**input, x)

##### #compile the model

model.compile(optimizer='adam', loss= categorical\_crossentropy , metrics=['accuracy'] )

##### #preparing our data using DataGenerator

train\_datagen = ImageDataGenerator( zoom\_range = 0.2,

shear\_range = 0.2, horizontal\_flip=True,

rescale = 1./255)

train\_data = train\_datagen.flow\_from\_directory(directory= "/sandhya/train1",

target\_size=(224,224), batch\_size=32,)

train\_data.class\_indices

##### #scaling the images

val\_datagen = ImageDataGenerator(rescale = 1./255 )

val\_data = val\_datagen.flow\_from\_directory(directory= "/sandhya/test1",

target\_size=(224,224), batch\_size=32,)

##### #visualizing the data that is fed to train data gen

##### # to visualize the images in the training data generator

t\_img , label = train\_data.next()

##### #-----------------------------------------------------------------------------

**# function when called will prot the images def** plotImages(img\_arr, label):

"""

input :- images array output :- plots the images """

count = 0

**for** im, l **in** zip(img\_arr,label) :

plt.imshow(im) plt.title(im.shape) plt.axis = **False** plt.show()

count += 1

**if** count == 10:

##### break

##### #-----------------------------------------------------------------------------

##### # function call to plot the images

plotImages(t\_img, label)

##### ## having early stopping and model check point

from keras.callbacks import ModelCheckpoint, EarlyStopping

##### # early stopping

es = EarlyStopping(monitor='val\_accuracy', min\_delta= 0.01 , patience= 5, verbose= 1, mode='auto')

##### # model check point

mc = ModelCheckpoint(filepath="best\_model.h5", monitor= 'val\_accuracy', verbose= 1, save\_best\_only= True, mode = 'auto')

##### # puting call back in a list

call\_back = [es, mc]

##### # fitting the train\_data in to model

hist **=** model**.**fit\_generator(train\_data,

steps\_per\_epoch**=** 10,

epochs**=** 30, validation\_data**=** val\_data, validation\_steps**=** 8, callbacks**=**[es,mc])

##### # Loading the best fit model

**from** keras.models **import** load\_model

model **=** load\_model("/content/best\_model.h5") h **=** hist**.**history

h**.**keys()

**#plotting the training accuracy** plt**.**plot(h['accuracy']) plt**.**plot(h['val\_accuracy'] , c **=** "red") plt**.**title("acc vs v-acc")

plt**.**show()

##### #plotting the training loss

plt**.**plot(h['loss']) plt**.**plot(h['val\_loss'] , c **=** "red") plt**.**title("loss vs v-loss") plt**.**show()

##### # just to map o/p values

op **=** dict(zip( train\_data**.**class\_indices**.**values(), train\_data**.**class\_indices**.**keys()))

##### # path for the image to see if it predicts correct class

path = "D:\project\happyboy.jpg"

img = load\_img(path, target\_size=(224,224) ) i = img\_to\_array(img)/255

input\_arr = np.array([i]) input\_arr.shape

pred = np.argmax(model.predict(input\_arr)) print(f" the image is of {op[pred]}")

**# to display the image** plt.imshow(input\_arr[0]) plt.title("input image") plt.show()

### REAL TIME LIVE VIDEO

##### #import packages

from keras.preprocessing.image import img\_to\_array import imutils import cv2

from keras.models import load\_model import numpy as np

##### # parameters for loading data and images

detection\_model\_path = 'haarcascade\_files/haarcascade\_frontalface\_default.xml' emotion\_model\_path = 'models/\_mini\_XCEPTION.102-0.66.hdf5'

##### # hyper-parameters for bounding boxes shape # loading models

face\_detection = cv2.CascadeClassifier(detection\_model\_path)

emotion\_classifier = load\_model(emotion\_model\_path, compile=False) EMOTIONS = ["angry"

,"disgust","scared", "happy", "sad", "surprised", "neutral"]

##### #feelings\_faces = []

##### #for index, emotion in enumerate(EMOTIONS):

##### # feelings\_faces.append(cv2.imread('emojis/' + emotion + '.png', -1)) # starting video streaming

cv2.namedWindow('your\_face') camera = cv2.VideoCapture(0) while **True**:

frame = camera.read()[1]

##### #reading the frame

frame = imutils.resize(frame,width=300)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

faces=face\_detection.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,30), flags=cv2.CASCADE\_SCALE\_IMAGE)

canvas = np.zeros((250, 300, 3), dtype="uint8") frameClone = frame.copy()

**if** len(faces) > 0:

faces = sorted(faces, reverse=True, key=lambda x: (x[2] - x[0]) \* (x[3] - x[1]))[0] (fX, fY, fW, fH) = faces

##### # Extract the ROI of the face from the grayscale image, resize it to a fixed 28x28 pixels, and then prepare

**# the ROI for classification** roi = gray[fY:fY + fH, fX:fX + fW] roi = cv2.resize(roi, (64, 64))

roi = roi.astype("float") / 255.0 roi = img\_to\_array(roi)

roi = np.expand\_dims(roi, axis=0)

preds = emotion\_classifier.predict(roi)[0] emotion\_probability = np.max(preds) label = EMOTIONS[preds.argmax()]

##### else: continue

**for** (i, (emotion, prob)) **in** enumerate(zip(EMOTIONS, preds))**: # construct the label text**

text = "{}: {:.2f}%".format(emotion, prob \* 100)

##### # draw the label + probability bar on the canvas # emoji\_face = feelings\_faces[np.argmax(preds)]

w = int(prob \* 300) cv2.rectangle(canvas, (7, (i \* 35) + 5),

(w, (i \* 35) + 35), (0, 0, 255), -1)

cv2.putText(canvas, text, (10, (i \* 35) + 23),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (255, 255, 255), 2)

cv2.putText(frameClone, label, (fX, fY - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (0, 0, 255), 2)

cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH), (0, 0, 255), 2)

##### # for c in range(0, 3):

##### # frame[200:320, 10:130, c] = emoji\_face[:, :, c] \* \ # (emoji\_face[:, :, 3] / 255.0) + frame[200:320,

##### # 10:130, c] \* (1.0 - emoji\_face[:, :, 3] / 255.0)

cv2.imshow('your\_face', frameClone) cv2.imshow("Probabilities", canvas) **if** cv2.waitKey(1) & 0xFF == ord('q'): **break**

camera.release()

cv2.destroyAllWindows()