**1. Introduction**

Many of the studies in recent years admit that humans reply and react to music and this music has a high impression on the activity of the human brain. In one examination of the explanations why people hear music, researchers discovered that music played a crucial role in relating arousal and mood. Two of the most important functions of music are it is ability is participants rated to help them achieve a good mood and become more self-aware. Musical preferences have been demonstrated to be highly related to personality traits and moods [1].

The meter, timbre, rhythm, and pitch of music are managed in areas of the brain that affects emotions and mood [2]. Interaction between individuals may be a major aspect of lifestyle. It reveals perfect details and much of data among humans, whether they are in the form of body language, speech, facial expression, or emotions [3]. Nowadays, emotion detection is considered the most important technique used in many applications such as smart card applications, surveillance, image database investigation, criminal, video indexing, civilian applications, security, and adaptive human-computer interface with multimedia environments.

With the increase in technology for digital signal processing and other effective feature extraction algorithms, automated emotion detection in multimedia attributes like music or movies is growing rapidly and this system can play an important role in many potential applications like human-computer interaction systems and music entertainment. We use facial expressions to propose a recommender system for emotion recognition that can detect user emotions and suggest a list of appropriate songs [13-24]. The proposed system detects the emotions of a person, if the person has a negative emotion, then a certain playlist will be shown that includes the most related types of music that will enhance his mood. And if the emotion is positive, a specific playlist will be presented which contains different types of music that will inflate the positive emotions [4]. The dataset we used for emotion detection is from Kaggle Facial Expression Recognition [5]. Dataset for the music player has been created from Bollywood Hindi songs. Implementation of facial emotion detection is performed using Convolutional Neural Network which gives approximately 95.14% of accuracy [2].

**2. literature Survey**

# Title: Smart music player integrating facial emotion recognition and music mood recommendation

**Author:**[Shlok Gilda](https://ieeexplore.ieee.org/author/37086195220), [Husain Zafar](https://ieeexplore.ieee.org/author/37086338167)

**Year: 2019**

**Description:** Songs, as a medium of expression, have always been a popular choice to depict and understand human emotions. Reliable emotion based classification systems can go a long way in helping us parse their meaning. However, research in the field of emotion-based music classification has not yielded optimal results. In this paper, we present an affective cross-platform music player, EMP, which recommends music based on the real-time mood of the user. EMP provides smart mood based music recommendation by incorporating the capabilities of emotion context reasoning within our adaptive music recommendation system. Our music player contains three modules: Emotion Module, Music Classification Module and Recommendation Module. The Emotion Module takes an image of the user's face as an input and makes use of deep learning algorithms to identify their mood with an accuracy of 90.23%. The Music Classification Module makes use of audio features to achieve a remarkable result of 97.69% while classifying songs into 4 different mood classes. The Recommendation Module suggests songs to the user by mapping their emotions to the mood type of the song, taking into consideration the preferences of the user.

# Title: An Intelligent Music Player Based on Emotion Recognition

**Author:**[RamyaRamanathan](https://ieeexplore.ieee.org/author/37086446596), [RadhaKumaran](https://ieeexplore.ieee.org/author/37086446097)

**Year: 2020**

**Description:** This paper proposes an intelligent agent that sorts a music collection based on the emotions conveyed by each song and then suggests an appropriate playlist to the user based on his/her current mood. The user's local music collection is initially clustered based on the emotion the song conveys, i.e. the mood of the song. This is calculated taking into consideration the lyrics of the song, as well as the melody. Every time the user wishes to generate a mood-based playlist, the user takes a picture of themselves at that instant. This image is subjected to facial detection and emotion recognition techniques, recognizing the emotion of the user. The music that best matches this emotion is then recommended to the user as a playlist.

# Title: Emotion Based Music Recommendation System Using Wearable Physiological Sensors

**Author:**[DegerAyata](https://ieeexplore.ieee.org/author/37085998566), [Yusuf Yaslan](https://ieeexplore.ieee.org/author/37269291100)

**Year: 2018**

**Description:** Most of the existing music recommendation systems use collaborative or content based recommendation engines. However, the music choice of a user is not only dependent to the historical preferences or music contents. But also dependent to the mood of that user. This paper proposes an emotion based music recommendation framework that learns the emotion of a user from the signals obtained via wearable physiological sensors. In particular, the emotion of a user is classified by a wearable computing device which is integrated with a galvanic skin response (GSR) and photo plethysmography (PPG) physiological sensors. This emotion information is feed to any collaborative or content based recommendation engine as a supplementary data. Thus, existing recommendation engine performances can be increased using these data. Therefore, in this paper emotion recognition problem is considered as arousal and valence prediction from multi-channel physiological signals. Experimental results are obtained on 32 subjects' GSR and PPG signal data with/out feature fusion using decision tree, random forest, support vector machine and k-nearest neighbors algorithms. The results of comprehensive experiments on real data confirm the accuracy of the proposed emotion classification system that can be integrated to any recommendation engine.

# Title: Music Recommender System for Users Based on Emotion Detection through Facial Features

**Author:**[AhlamAlrihaili](https://ieeexplore.ieee.org/author/37088383948), [AlaaAlsaedi](https://ieeexplore.ieee.org/author/37088340639)

**Year: 2019**

**Description:** In recent years, facial emotion detection received massive attention because of its applications in computer vision and human-computer interaction fields. Due to the active works in this field, various algorithms and applications were proposed and implemented. In this research, we propose a recommender system for emotion recognition that is capable of detecting the user emotions and suggest a list of appropriate songs that can improve his mood. A brief search was conducted on how music can affect the user mood in short-term to gain knowledge and enable us to provide the users with a list of music tracks that work well on improving the user moods. The proposed system detects the emotions, if the subject has a negative emotion then specific playlist will be presented that contains the most suitable types of music that will improve his mood. On the other hand, if the detected emotion is positive, a suitable playlist will be provided which includes different types of music that will enhance the positive emotions. Implementation of the proposed recommender system is performed using Viola-Jonze algorithm and Principal Component Analysis (PCA) techniques, we were able to implement the proposed system successfully in MATLAB(R2018a).

# Title: How convolutional neural network see the world - A survey of convolutional neural network visualization methods

**Author:**[Zhuwei Qin](https://arxiv.org/search/cs?searchtype=author&query=Qin%2C+Z), [Fuxun Yu](https://arxiv.org/search/cs?searchtype=author&query=Yu%2C+F)

**Year: 2020**

**Description:** Nowadays, the Convolutional Neural Networks (CNNs) have achieved impressive performance on many computer vision related tasks, such as object detection, image recognition, image retrieval, etc. These achievements benefit from the CNNs outstanding capability to learn the input features with deep layers of neuron structures and iterative training process. However, these learned features are hard to identify and interpret from a human vision perspective, causing a lack of understanding of the CNNs internal working mechanism. To improve the CNN interpretability, the CNN visualization is well utilized as a qualitative analysis method, which translates the internal features into visually perceptible patterns. And many CNN visualization works have been proposed in the literature to interpret the CNN in perspectives of network structure, operation, and semantic concept. In this paper, we expect to provide a comprehensive survey of several representative CNN visualization methods, including Activation Maximization, Network Inversion, Deconvolutional Neural Networks (DeconvNet), and Network Dissection based visualization. These methods are presented in terms of motivations, algorithms, and experiment results. Based on these visualization methods, we also discuss their practical applications to demonstrate the significance of the CNN interpretability in areas of network design, optimization, security enhancement, etc.

**3. System Analysis**

**3.1 Proposed System**

The proposed system benefits us to present interaction between the user and the music player. The purpose of the system is to capture the face properly with the camera. Captured images are fed into the Convolutional Neural Network which predicts the emotion. Then emotion derived from the captured image is used to get a playlist of songs. The main aim of our proposed system is to provide a music playlist automatically to change the user's moods, which can be happy, sad, natural, or surprised. The proposed system detects the emotions, if the topic features a negative emotion, then a selected playlist is going to be presented that contains the foremost suitable sorts of music that will enhance the mood of the person positively.

**Advantage**

* It will try to enhance the users’ mood.
* Testing of the system is done on the FER2013 dataset. Facial expressions are captured using an inbuilt camera.
* This system is better than static recommendation system as it will suggest music based on users’ mood by facial recognition and helps them to improve their mood.

**3.2 Existing system**

Many of the studies in recent years admit that humans reply and react to music and this music has a high impression on the activity of the human brain.The meter, timbre, rhythm, and pitch of music are managed in areas of the brain that affects emotions and mood.This emotion information is feed to any collaborative or content based recommendation engine as a supplementary data. Thus, existing recommendation engine performances can be decreased using these data.

**Disadvantage**

* These systems only allow you to live a static user-experience as the system will give recommendation based on the history without regard to other parameters that might impact the prediction such as feeling or emotion.
* These recommendation systems will sometimes fail to give the correct output because their suggestions are based on outdated input.
* So, the user cannot be satisfied with the output as it doesn’t satisfy his emotion

**4. SYSTEM REQUIREMENT**

**HARDWARE REQUIREMENTS:**

* **System :** Pentium IV 2.4 GHz.
* **Hard Disk :** 40 GB.
* **Floppy Drive :** 1.44 Mb.
* **Monitor** : 14’ Colour Monitor.
* **Mouse :** Optical Mouse.
* **Ram :** 512 Mb.

**SOFTWARE REQUIREMENTS:**

* **Operating system :** Windows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.

**5. Modules**

**5.1 Database Description**

We built the Convolutional Neural Network model using the Kaggle dataset. The database is FER2013 which is split into two parts training and testing dataset. The training dataset consists of 24176 and the testing dataset contains 6043 images. There are 48x48 pixel grayscale images of faces in the dataset. Each image in FER-2013 is labeled as one of five emotions: happy, sad, angry, surprise, and neutral. The faces are automatically registered so that they are more or less centered in each image and take up about the same amount of space. The images in FER-2013 contain both posed and unposed headshots, which are in grayscale and 48x48 pixels.

The FER-2013 dataset was created by gathering the results of a Google image search of every emotion and synonyms of the emotions. FER systems being trained on an imbalanced dataset may perform well on dominant emotions such as happy, sad, angry, neutral, and surprised but they perform poorly on the under-represented ones like disgust and fear. Usually, the weighted-SoftMax loss approach is used to handle this problem by weighting the loss term for each emotion class supported by its relative proportion within the training set. However, this weighted-loss approach is predicated on the SoftMax loss function, which is reported to easily force features of various classes to stay apart without listening to intra-class compactness. One effective strategy to deal with the matter of SoftMax loss is to use an auxiliary loss to coach the neural network. To treating missing and Outlier values we have used a loss function named categorical crossentropy. For each iteration, a selected loss function is employed to gauge the error value. So, to treating missing and Outlier values, we have used a loss function named categorical crossentropy.

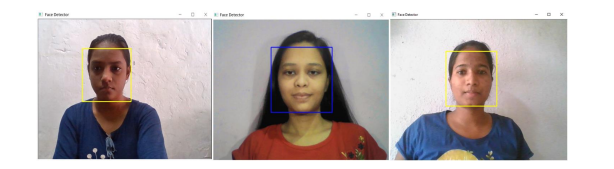


**Figure 1. Samples from FER2013 dataset.**

**5.2 Emotion Detection Module**

**5.2.1 Face Detection**

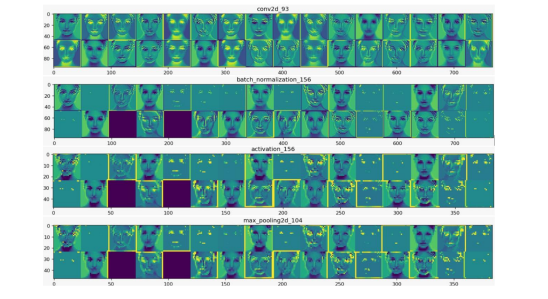
Face detection is one of the applications which is considered under computer vision technology. This is the process in which algorithms are developed and trained to properly locate faces or objects in object detection or related system in images. This detection can be real-time from a video frame or images. Face detection uses such classifiers, which are algorithms that detect what's either a face (1) or not a face (0) in an image. Classifiers are trained to detect faces using numbers of images to get more accuracy. OpenCV uses two sorts of classifiers, LBP (Local Binary Pattern) and Haar Cascades. A Haar classifier is used for face detection where the classifier is trained with pre-defined varying face data which enables it to detect different faces accurately. The main aim of face detection is to spot the face within the frame by reducing external noises and other factors. It is a machine learning-based approach where the cascade function is trained with a group of input files. It is supported the Haar Wavelet technique to research pixels inside the image into squares by function [9]. This uses machine learning techniques to urge a high degree of accuracy from what's called "training data".



**Figure 2. Face detection**

**5.2.2 Feature Extraction**

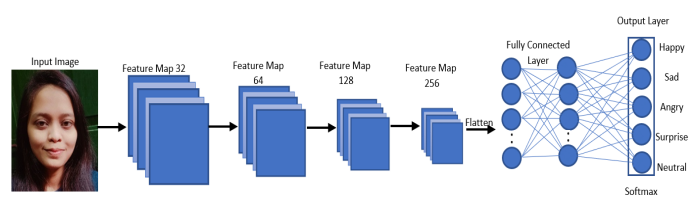
While performing feature extraction, we treat the pre-trained network that is a sequential model as an arbitrary feature extractor. Allowing the input image to pass on it forward, stopping at the pre-specified layer, and taking the outputs of that layer as our features. Starting layers of a convolutional network extract high-level features from the taken image, so use only a few filters. As we make further deeper layers, we increase the number of the filters to twice or thrice the dimension of the filter of the previous layer. Filters of the deeper layers gain more features but are computationally very intensive.



**Figure 3. Visualization of The Feature Map**

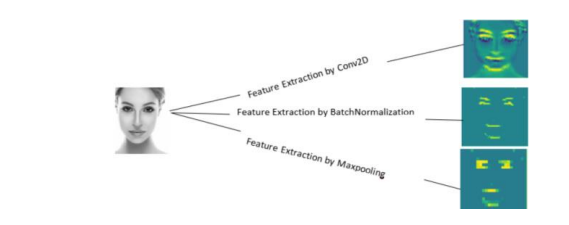
Doing this we utilized the robust, discriminative features learned by the Convolution neural network [10]. The outputs of the model are going to be feature maps, which are an intermediate representation for all layers after the very first layer. Load the input image for which we want to view the Feature map to know which features were prominent to classify the image. Feature maps are obtained by applying Filters or Feature detectors to the input image or the feature map output of the prior layers. Feature map visualization will provide insight into the interior representations for specific input for each of the Convolutional layers within the model.

**5.2.3 Emotion Detection**



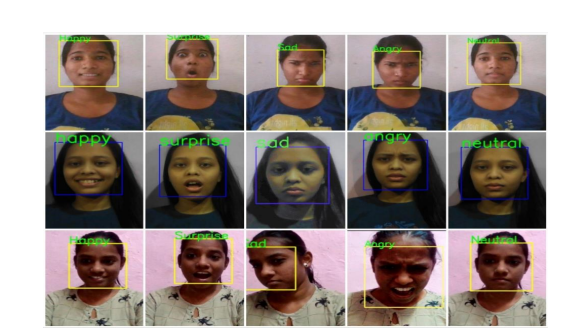
**Figure 4. Convolution neural network Architecture**

Convolution neural network architecture applies filters or feature detectors to the input image to get the feature maps or activation maps using the Relu activation function [11]. Feature detectors or filters help in identifying various features present in the image such as edges, vertical lines, horizontal lines, bends, etc. After that pooling is applied over the feature maps for invariance to translation. Pooling is predicted on the concept that once we change the input by a touch amount, the pooled outputs don’t change. We can use any of the pooling from min, average, or max. But max-pooling provides better performance than min or average pooling. Flatten all the input and giving these flattened inputs to a deep neural network which are outputs to the class of the object.



**Figure 5. Feature Extraction by each layer in Convolutional Neural Network**

The class of the image will be binary, or it will be a multi-class classification for identifying digits or separating various apparel items. Neural networks are as a black box, and learned features in a Neural Network are not interpretable. So basically, we give an input image then the CNN model returns the results [10]. Emotion detection is performed by loading the model which is trained by weights using CNN. When we take the real-time image by a user then that image was sent to the pre-trained CNN model, then predict the emotion and adds the label to the image.



**Figure 6. Results of Emotion Detection.**

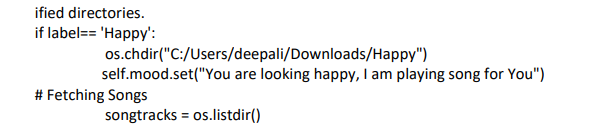
**5.3 Music Recommendation Module**

**5.3.1 Songs Database**

We created a database for Bollywood Hindi songs. It consists of 100 to 150 songs per emotion. As we all know music is undoubtedly involved in enhancing our mood. So, suppose a user is sad then the system will recommend such a music playlist which motivates him or her and by this automatic mood will be delighted.

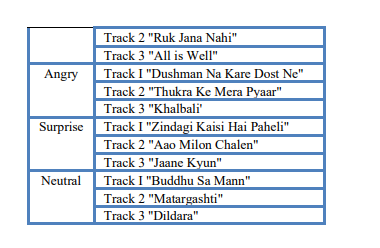
**5.3.2 Music Playlist Recommendation**

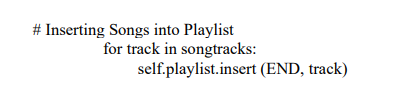
By using the emotion module real-time emotion of the user is detected. This will give the labels like Happy, Sad, Angry, Surprise, and Neutral. Using the os.listdir() method in python we connected these labels with the folders of the songs database which we have created. Table 1 shows the list of songs. This method of os.listdir() is used to get the list of any file in the specified directories.



**Table 1. Database of songs.**

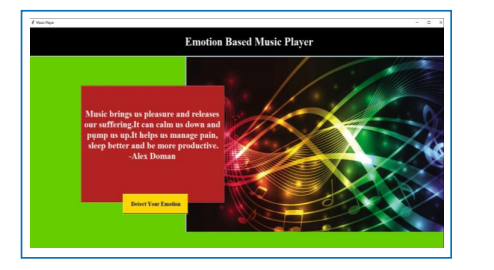




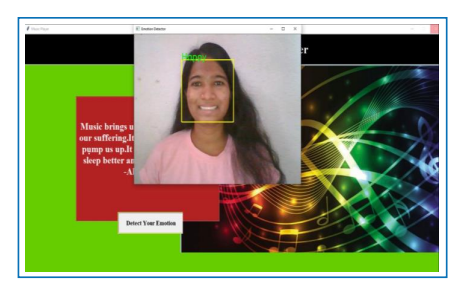


This will result in the recommended playlist for the user in the GUI of the music player by showing captions according to detected emotions. We have used a library called Pygame for playing the audio as this library supports playing various multimedia formats like audio, video, etc. Functions of this library such as playsong, pauseong, resumesong, and stopsong are used to working with the music player. Variables like playlist, songstatus, and root are used for storing the name of all songs, storing the status of currently active songs, and for the main GUI window respectively. For developing the GUI, we have used Tkinter.

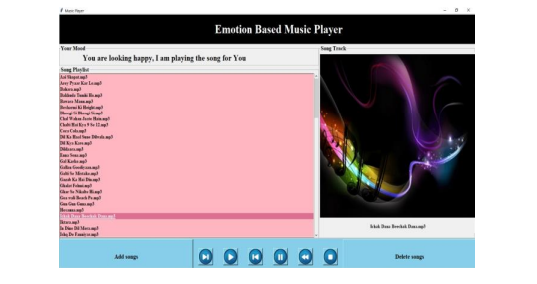
Here are the results:



**Figure 7. GUI of the front page.**



**Figure 8. Detection of emotion**



**Figure 9. Recommendation of music playlist**

**6. Algorithm**

**Opencv**

OpenCV is a cross-platform library using which we can develop real-time **computer vision applications**. It mainly focuses on image processing, video capture and analysis including features like face detection and object detection.

Let’s start the chapter by defining the term "Computer Vision".

**Computer Vision**

Computer Vision can be defined as a discipline that explains how to reconstruct, interrupt, and understand a 3D scene from its 2D images, in terms of the properties of the structure present in the scene. It deals with modeling and replicating human vision using computer software and hardware.

Computer Vision overlaps significantly with the following fields

* **Image Processing** − It focuses on image manipulation.
* **Pattern Recognition** − It explains various techniques to classify patterns.
* **Photogrammetry** − It is concerned with obtaining accurate measurements from images.

### Computer Vision Vs Image Processing

**Image processing** deals with image-to-image transformation. The input and output of image processing are both images.

**Computer vision** is the construction of explicit, meaningful descriptions of physical objects from their image. The output of computer vision is a description or an interpretation of structures in 3D scene.

## Applications of Computer Vision

Here we have listed down some of major domains where Computer Vision is heavily used.

### Robotics Application

* Localization − Determine robot location automatically
* Navigation
* Obstacles avoidance
* Assembly (peg-in-hole, welding, painting)
* Manipulation (e.g. PUMA robot manipulator)
* Human Robot Interaction (HRI) − Intelligent robotics to interact with and serve people

### Medicine Application

* Classification and detection (e.g. lesion or cells classification and tumor detection)
* 2D/3D segmentation
* 3D human organ reconstruction (MRI or ultrasound)
* Vision-guided robotics surgery

### Industrial Automation Application

* Industrial inspection (defect detection)
* Assembly
* Barcode and package label reading
* Object sorting
* Document understanding (e.g. OCR)

### Security Application

* Biometrics (iris, finger print, face recognition)
* Surveillance − Detecting certain suspicious activities or behaviors

### Transportation Application

* Autonomous vehicle
* Safety, e.g., driver vigilance monitoring

## Features of OpenCV Library

Using OpenCV library, you can −

* Read and write images
* Capture and save videos
* Process images (filter, transform)
* Perform feature detection
* Detect specific objects such as faces, eyes, cars, in the videos or images.
* Analyze the video, i.e., estimate the motion in it, subtract the background, and track objects in it.

OpenCV was originally developed in C++. In addition to it, Python and Java bindings were provided. OpenCV runs on various Operating Systems such as windows, Linux, OSx, FreeBSD, Net BSD, Open BSD, etc.

This tutorial explains the concepts of OpenCV with examples using Java bindings.

## OpenCV Library Modules

Following are the main library modules of the OpenCV library.

### Core Functionality

This module covers the basic data structures such as Scalar, Point, Range, etc., that are used to build OpenCV applications. In addition to these, it also includes the multidimensional array **Mat**, which is used to store the images. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.core**.

### Image Processing

This module covers various image processing operations such as image filtering, geometrical image transformations, color space conversion, histograms, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.imgproc**.

### Video

This module covers the video analysis concepts such as motion estimation, background subtraction, and object tracking. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.video**.

### Video I/O

This module explains the video capturing and video codecs using OpenCV library. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.videoio**.

### calib3d

This module includes algorithms regarding basic multiple-view geometry algorithms, single and stereo camera calibration, object pose estimation, stereo correspondence and elements of 3D reconstruction. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.calib3d**.

### features2d

This module includes the concepts of feature detection and description. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.features2d**.

### Objdetect

This module includes the detection of objects and instances of the predefined classes such as faces, eyes, mugs, people, cars, etc. In the Java library of OpenCV, this module is included as a package with the name **org.opencv.objdetect**.

### Highgui

This is an easy-to-use interface with simple UI capabilities. In the Java library of OpenCV, the features of this module is included in two different packages namely, **org.opencv.imgcodecs** and **org.opencv.videoio**.

## A Brief History of OpenCV

OpenCV was initially an Intel research initiative to advise CPU-intensive applications. It was officially launched in 1999.

* In the year 2006, its first major version, OpenCV 1.0 was released.
* In October 2009, the second major version, OpenCV 2 was released.
* In August 2012, OpenCV was taken by a nonprofit organization OpenCV.org.

### Image processing operations

When you write a Computer Vision algorithm, there are a lot of basic image processing operations that you will use over and over again. Most of these functions are present in the imgproc module. You can do things such as image filtering, morphological operations, geometric transformations, colorconversions, drawing on images, histograms, shape analysis, motion analysis, feature detection, and so on. Let's consider the following figure:

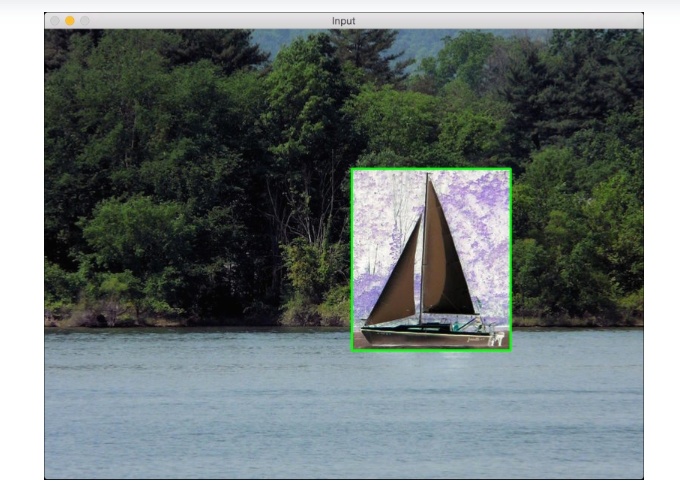


The right-hand side image is a rotated version of the left-hand side image. We can do this transformation with a single line in OpenCV. There is another module called ximgproc that contains advanced image processing algorithms such as structured forests for edge detection, domain transform filters, adaptive manifold filters, and so on.

### Building GUI

OpenCV provides a module called highgui that handles all the high-level user interface operations. Let's say that you are working on a problem and you want to check what the image looks like before you proceed to the next step. This module has functions that can be used to create windows to display images and/or video. There is also a waiting function that will wait until you hit a key on your keyboard before it goes to the next step. There is a function that can detect mouse events as well. This is very useful to develop interactive applications. Using this functionality, you can draw rectangles on these input windows and then proceed based on the selected region.

Consider the following image:



As you can see, we have drawn a green rectangle on the image and applied a negative film effect to that region. Once we have the coordinates of this rectangle, we can operate only on that region.

### Video analysis

Video analysis includes tasks such as analyzing the motion between successive frames in a video, tracking different objects in a video, creating models for video surveillance, and so on. OpenCV provides a module called video that can handle all of this. There is a module called videostab that deals with video stabilization. Video stabilization is an important part of video cameras. When you capture videos by holding the camera in your hands, it's hard to keep your hands perfectly steady. If you look at that video as it is, it will look bad and jittery. All modern devices use video stabilization techniques to process the videos before they are presented to the end user.

### 3D reconstruction

3D reconstruction is an important topic in Computer Vision. Given a set of 2D images, we can reconstruct the 3D scene using the relevant algorithms. OpenCVprovides algorithms that can find the relationship between various objects in these 2D images to compute their 3D positions. We have a module called calib3d that can handle all this. This module can also handle camera calibration, which is essential to estimate the parameters of the camera. These parameters are basically the internal parameters of any given camera that uses them to transform the captured scene into an image. We need to know these parameters to design algorithms, or else we might get unexpected results. Let's consider the following figure:



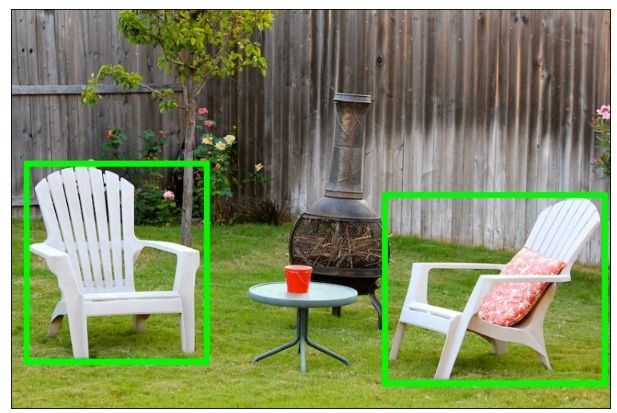
As shown in the preceding image, the same object is captured from multiple poses. Our job is to reconstruct the original object using these 2D images.

### Feature extraction

As discussed earlier, the human visual system tends to extract the salient features from a given scene so that it can be retrieved later. To mimic this, people started designing various feature extractors that can extract these salient points from a given image. Some of the popular algorithms include **SIFT** (**Scale Invariant Feature Transform**), **SURF** (**Speeded Up Robust Features**), **FAST** (**Features from Accelerated Segment Test**), and so on. There is a module called features2d that provides functions to detect and extract all these features. There is another module called xfeatures2d that provides a few more feature extractors, some of which are still in the experimental phase. You can play around with these if you get a chance. There is also a module called bioinspired that provides algorithms for biologically inspired Computer Vision models.

### Object detection

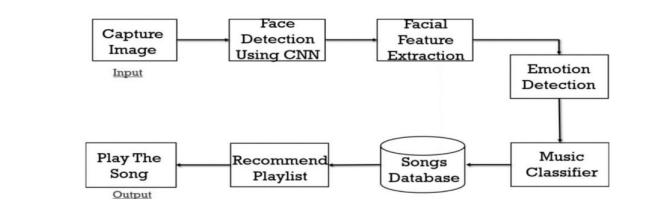
Object detection refers to detecting the location of an object in a given image. This process is not concerned with the type of object. If you design a chair detector, it will just tell you the location of the chair in a given image. It will not tell you whether it's a red chair with a high back or a blue chair with a low back. Detecting the location of objects is a very critical step in many Computer Vision systems. Consider the following image:



If you run a chair detector on this image, it will put a green box around all the chairs. It won't tell you what kind of chair it is! Object detection used to be a computationally intensive task because of the number of calculations required to perform the detection at various scales. To solve this, Paul Viola and Michael Jones came up with a great algorithm in their seminal paper in 2001. You can read it at <https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>. They provided a fast way to design an object detector for any object. OpenCV has modules called objdetect and xobjdetect that provide the framework to design an object detector. You can use it to develop detectors for random items such as sunglasses, boots, and so on.

**7. System Design**

**7.1 Architecture Diagram**



**Figure 10. Block diagram of the proposed system**

**8. System testing**

**PYTHON**

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [CPython](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive [standard library](https://en.wikipedia.org/wiki/Standard_library)

Python is a [multi-paradigm programming language](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language). [Object-oriented programming](https://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](https://en.wikipedia.org/wiki/Structured_programming) are fully supported, and many of its features support [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and [aspect-oriented programming](https://en.wikipedia.org/wiki/Aspect-oriented_programming)

Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the identical manner to that forming tuple literals and, as a whole, are put on the left hand side of the equal sign in an assignment statement. The statement expects an iterable object on the right hand side of the equal sign that produces the same number of values as the provided writable expressions when iterated through, and will iterate through it, assigning each of the produced values to the corresponding expression on the left.

The assignment statement (token '=', the equals sign). This operates differently than in traditional [imperative programming](https://en.wikipedia.org/wiki/Imperative_programming) languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in [C](https://en.wikipedia.org/wiki/C_(programming_language)), e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an [allocated storage location](https://en.wikipedia.org/wiki/Memory_allocation) for which the (left-hand) [variable name](https://en.wikipedia.org/wiki/Variable_(computer_science)) is the symbolic address.

The memory allocated to the variable is large enough (potentially quite large) for the declared [type](https://en.wikipedia.org/wiki/Type_system). In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a [reference](https://en.wikipedia.org/wiki/Pointer_(computer_programming)) to a separate, dynamically allocated [object](https://en.wikipedia.org/wiki/Object_(computer_science)) of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed [data type](https://en.wikipedia.org/wiki/Type_system) with it. However at a given time a name will be bound to some object,

Python has [array index](https://en.wikipedia.org/wiki/Array_index) and [array slicing](https://en.wikipedia.org/wiki/Array_slicing) expressions on lists, denoted as

 a[key],

a[start:stop]

 or

a[start:stop:step].

Indexes are [zero-based](https://en.wikipedia.org/wiki/Zero-based), and negative indexes are relative to the end. Slices take elements from the start index up to, but not including, the stop index. The third slice parameter, called step or stride, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a copy of the entire list. Each element of a slice is a [shallow copy](https://en.wikipedia.org/wiki/Shallow_copy).

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)), or [Ruby](https://en.wikipedia.org/wiki/Ruby_(programming_language)). This leads to duplicating some functionality. For example:

[**List comprehensions**](https://en.wikipedia.org/wiki/List_comprehensions)**vs. for-loops**

[**Conditional**](https://en.wikipedia.org/wiki/Conditional_(programming))**expressions vs. if blocks**

The eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression, so list and other comprehensions or [lambda expressions](https://en.wikipedia.org/wiki/Lambda_(programming)), all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as a = 1 cannot form part of the conditional expression of a conditional statement. This has the advantage of avoiding a classic C error of mistaking an assignment operator = for an equality operator == in conditions: if (c = 1) { ... } is syntactically valid (but probably unintended) C code but if c = 1: ...causes a syntax error in Python.

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An [interpreted language](https://en.wikipedia.org/wiki/Interpreted_language), Python has a design philosophy that emphasizes code [readability](https://en.wikipedia.org/wiki/Readability) (notably using [whitespace](https://en.wikipedia.org/wiki/Whitespace_character) indentation to delimit [code blocks](https://en.wikipedia.org/wiki/Code_block) rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer [lines of code](https://en.wikipedia.org/wiki/Source_lines_of_code) than might be used in languages such as [C++](https://en.wikipedia.org/wiki/C%2B%2B)or [Java](https://en.wikipedia.org/wiki/Java_(programming_language)). It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many [operating systems](https://en.wikipedia.org/wiki/Operating_system). [C Python](https://en.wikipedia.org/wiki/CPython), the [reference implementation](https://en.wikipedia.org/wiki/Reference_implementation) of Python, is [open source](https://en.wikipedia.org/wiki/Open_source) software and has a community-based development model, as do nearly all of its variant implementations. C Python is managed by the non-profit [Python Software Foundation](https://en.wikipedia.org/wiki/Python_Software_Foundation). Python features a [dynamic type](https://en.wikipedia.org/wiki/Dynamic_type) system and automatic [memory management](https://en.wikipedia.org/wiki/Memory_management). It supports multiple [programming paradigms](https://en.wikipedia.org/wiki/Programming_paradigm), including [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming), [imperative](https://en.wikipedia.org/wiki/Imperative_programming), [functional](https://en.wikipedia.org/wiki/Functional_programming) and [procedural](https://en.wikipedia.org/wiki/Procedural_programming), and has a large and comprehensive standard library

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write yourprograms.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code withinobjects.
* **Python is a Beginner's Language** − Python is a great language for the beginner- level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers togames.

**Python's features include −**

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the languagequickly.
* **Easy-to-read** − Python code is more clearly defined and visible to theeyes.
* **Easy-to-maintain** − Python's source code is fairlyeasy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross- platform compatible on UNIX, Windows, andMacintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets ofcode.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on allplatforms.
* **Extendable** − you can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be moreefficient.
* **Databases** − Python provides interfaces to all major commercialdatabases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system ofUnix.

**Scalable** − Python provides a better structure and support for large programs than shellscripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* It supports functional and structured programming methods as well asOOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic typechecking.
* It supports automatic garbagecollection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, andJava.

**DJANGO**

Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source.

Django's primary goal is to ease the creation of complex, database-driven websites. Django emphasizes [reusability](https://en.wikipedia.org/wiki/Reusability)and "pluggability" of components, rapid development, and the principle of [don't repeat yourself](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself). Python is used throughout, even for settings files and data models.



Django also provides an optional administrative [create, read, update and delete](https://en.wikipedia.org/wiki/Create,_read,_update_and_delete) interface that is generated dynamically through [introspection](https://en.wikipedia.org/wiki/Introspection_(computer_science)) and configured via admin models



**Python – The New Generation Language**

Python is a widely used general-purpose, high level programming language. It was initially designed by

Guido van Rossum in 1991 and developed by Python Software Foundation. It was mainly developed for an emphasis on code readability, and its syntax allows programmers to express concepts in fewer lines of code.

Python is dynamically typed and garbage-collected. It supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Python is often described as a "batteries included"language due to its comprehensive standard library.

**Features**

• Interpreted

In Python there is no separate compilation and execution steps like C/C++. It directly run the program

from the source code. Internally, Python converts the source code into an intermediate form called bytecodes which is then translated into native language of specific computer to run it.

• Platform Independent

Python programs can be developed and executed on the multiple operating system platform. Python can

be used on Linux, Windows, Macintosh, Solaris and many more.

• Multi- Paradigm

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and as pectorientedprogramming .

• Simple

Python is a very simple language. It is a very easy to learn as it is closer to English language. In python more emphasis is on the solution to the problem rather than the syntax.

• Rich Library Support

Python standard library is very vast. It can help to do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, email, XML, HTML,

WAV files, cryptography, GUI and many more.

• Free and Open Source

Firstly, Python is freely available. Secondly, it is open-source. This means that its source code is available to the public. We can download it, change it, use it, and distribute it. This is called FLOSS (Free/Libre and Open Source Software). As the Python community, we’re all headed toward one goal- an

ever-bettering Python

**Types of Machine Learning**

The types of machine learning algorithms differ in their approach, the type of data they input and output, andthe type of task or problem that they are intended to solve. Broadly Machine Learning can be categorized into four categories. I.

**Supervised Learning**

**Unsupervised Learning**

**Reinforcement LearningIV.**

Semi-supervised Learning Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly

**Supervised Learning**

Supervised Learning is a type of learning in which we are given a data set and we already know what are correct output should look like, having the idea that there is a relationship between the input and output. Basically, it is learning task of learning a function that maps an input to an output based on example input-output pairs. It infers a function from labeled training data consisting of a set of training examples. Supervised learning problems are categorized

**Unsupervised Learning**

Unsupervised Learning is a type of learning that allows us to approach problems with little or no idea what our problem should look like. We can derive the structure by clustering the data based on a relationship among the variables in data. With unsupervised learning there is no feedback based on prediction result. Basically, it is a type of self-organized learning that helps in finding previously unknown patterns in data set without pre-existing label.

**Reinforcement Learning**

Reinforcement learning is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine theidealbehavior within a specific context in order to maximize its performance. Simple reward feedback isrequired for the agent to learn which action is best.

**Semi-Supervised Learning**

Semi-supervised learning fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant

resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn’t require

**Objectives**

Main objectives of training were to learn:

How to determine and measure program complexity,

* Python Programming
* ML Library Scikit, Numpy , Matplotlib, Pandas , Theano , TensorFlow
* Statistical Math for the Algorithms.
* Learning to solve statistics and mathematical concepts.
* Supervised and Unsupervised Learning
* Classification and Regression

**ML Algorithms**

**Machine Learning Programming and Use Case**

**Advantages of Machine Learning**

Every coin has two faces, each face has its own property and features. It’s time to uncover the faces of ML.

A very powerful tool that holds the potential to revolutionize the way things work.

1. Easily identifies trends and patterns -

Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans. For instance, for an e-commerce website like Amazon, it serves to understand the browsing behaviors and purchase histories of its users to help cater to the right products, deals, and reminders relevant to them. It uses the results to reveal relevant advertisements to them.

2. No human intervention needed (automation) -

With ML, we don’t need to babysit our project every step of the way. Since it means giving machines the ability to learn, it lets them make predictions and also improve the algorithms on their own. A common example of this is anti-virus software. they learn to filter new threats as they are recognized. ML is also good at recognizing spam.

3. Continuous Improvement -

As ML algorithms gain experience, they keep improving in accuracy and efficiency. This lets them make better decisions. Say we need to make a weather forecast model. As the amount of data, we have keeps growing, our algorithms learn to make more accurate predictions faster.

4. Handling multi-dimensional and multi-variety data -Machine Learning algorithms are good at handling data that are multi-dimensional and multi-variety, and they can do this in dynamic or uncertain environments.

5. Wide Applications -

We could be an e-seller or a healthcare provider and make ML work for us. Where it does apply, it holds the capability to help deliver a much more personal experience to customers while also targeting the right customers.

**Applications of Machine Learning**

**Applications of Machine Learning include:**

**Web Search Engine**:

One of the reasons why search engines like google, bin getc work so well is because the system has learnt how to rank pages through a complex learning algorithm.

**Photo tagging Applications:**

Be it facebook or any other photo tagging application, the ability to tag friends makes it even more happening. It is all possible because of a face recognition algorithm that runs behind the application.

**Spam Detector:**

Our mail agent like Gmail or Hotmail does a lot of hard work for us in classifying the mails and moving the spam mails to spam folder. This is again achieved by a spam classifier running in the back end of mail application.

**Database Mining for growth of automation:**

Typical applications include Web-click data for better UX, Medical records for better automation in healthcare, biological data and many more.

**Applications that cannot be programmed:**

There are some tasks that cannot be programmed as the computers we use are not modelled that way. Examples include Autonomous Driving, Recognition tasks from unordered data (Face Recognition/ Handwriting Recognition), Natural language Processing, computer Vision etc.

**Understanding Human Learning:**

This is the closest we have understood and mimicked the human brain. It is the start of a new revolution, The real AI. Now, after a brief insight lets come to a moreformal definition of Machine Learning

**REQUIREMENT ANALYSIS**

The project involved analyzing the design of few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

**REQUIREMENT SPECIFICATION**

**Functional Requirements**

* Graphical User interface with the User.

**Software Requirements**

For developing the application the following are the Software Requirements:

1. Python
2. Django
3. MySql
4. MySqlclient
5. WampServer 2.4

**Operating Systems supported**

1. Windows 7
2. Windows XP
3. Windows 8

**Technologies and Languages used to Develop**

1. Python

**Debugger and Emulator**

* Any Browser (Particularly Chrome)

**Hardware Requirements**

For developing the application the following are the Hardware Requirements:

* Processor: Pentium IV or higher
* RAM: 256 MB
* Space on Hard Disk: minimum 512MB

**8. SYSTEM STUDY**

**FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

**Three key considerations involved in the feasibility analysis are,**

* **ECONOMICAL FEASIBILITY**
* **TECHNICAL FEASIBILITY**
* **SOCIAL FEASIBILITY**

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**SYSTEM TEST**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, **the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination** of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**10.SAMPLE CODING:-**

**LOAD AND PROCESS.PY:-**

from sklearn.model\_selection import train\_test\_split

import pandas as pd

import cv2

import numpy as np

dataset\_path = 'fer2013/fer2013/fer2013.csv'

image\_size=(48,48)

def load\_fer2013():

data = pd.read\_csv(dataset\_path)

pixels = data['pixels'].tolist()

width, height = 48, 48

faces = []

for pixel\_sequence in pixels:

face = [int(pixel) for pixel in pixel\_sequence.split(' ')]

face = np.asarray(face).reshape(width, height)

face = cv2.resize(face.astype('uint8'),image\_size)

faces.append(face.astype('float32'))

faces = np.asarray(faces)

faces = np.expand\_dims(faces, -1)

emotions = pd.get\_dummies(data['emotion']).as\_matrix()

return faces, emotions

def preprocess\_input(x, v2=True):

x = x.astype('float32')

x = x / 255.0

if v2:

x = x - 0.5

x = x \* 2.0

return x

faces, emotions = load\_fer2013()

faces = preprocess\_input(faces)

xtrain, xtest,ytrain,ytest = train\_test\_split(faces, emotions,test\_size=0.2,shuffle=True)

**TRAIN EMOTION CLASSIFIER.PY:-**

"""

Description: Train emotion classification model

"""

from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping

from keras.callbacks import ReduceLROnPlateau

from keras.preprocessing.image import ImageDataGenerator

from sklearn.model\_selection import train\_test\_split

from keras.layers import Activation, Convolution2D, Dropout, Conv2D

from keras.layers import AveragePooling2D, BatchNormalization

from keras.layers import GlobalAveragePooling2D

from keras.models import Sequential

from keras.layers import Flatten

from keras.models import Model

from keras.layers import Input

from keras.layers import MaxPooling2D

from keras.layers import SeparableConv2D

from keras import layers

from keras.regularizers import l2

from load\_and\_process import load\_fer2013

from load\_and\_process import preprocess\_input

from models.cnn import mini\_XCEPTION

import pandas as pd

import cv2

import numpy as np

# parameters

batch\_size = 32

num\_epochs = 5

input\_shape = (48, 48, 1)

validation\_split = .2

verbose = 1

num\_classes = 7

patience = 50

base\_path = 'models/'

l2\_regularization=0.01

# data generator

data\_generator = ImageDataGenerator(

featurewise\_center=False,

featurewise\_std\_normalization=False,

rotation\_range=10,

width\_shift\_range=0.1,

height\_shift\_range=0.1,

zoom\_range=.1,

horizontal\_flip=True)

# model parameters/compilation

model = mini\_XCEPTION(input\_shape, num\_classes)

model.compile(optimizer='adam', loss='categorical\_crossentropy',

metrics=['accuracy'])

regularization = l2(l2\_regularization)

model.summary()

# base

img\_input = Input(input\_shape)

x = Conv2D(8, (3, 3), strides=(1, 1), kernel\_regularizer=regularization, use\_bias=False)(img\_input)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = Conv2D(8, (3, 3), strides=(1, 1), kernel\_regularizer=regularization, use\_bias=False)(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

# module 1

residual = Conv2D(16, (1, 1), strides=(2, 2), padding='same', use\_bias=False)(x)

residual = BatchNormalization()(residual)

x = SeparableConv2D(16, (3, 3), padding='same', kernel\_regularizer=regularization, use\_bias=False)(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = SeparableConv2D(16, (3, 3), padding='same', kernel\_regularizer=regularization, use\_bias=False)(x)

x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

x = layers.add([x, residual])

# module 2

residual = Conv2D(32, (1, 1), strides=(2, 2), padding='same', use\_bias=False)(x)

residual = BatchNormalization()(residual)

x = SeparableConv2D(32, (3, 3), padding='same', kernel\_regularizer=regularization, use\_bias=False)(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = SeparableConv2D(32, (3, 3), padding='same', kernel\_regularizer=regularization, use\_bias=False)(x)

x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

x = layers.add([x, residual])

# module 3

residual = Conv2D(64, (1, 1), strides=(2, 2),padding='same', use\_bias=False)(x)

residual = BatchNormalization()(residual)

x = SeparableConv2D(64, (3, 3), padding='same',kernel\_regularizer=regularization,use\_bias=False)(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = SeparableConv2D(64, (3, 3), padding='same',kernel\_regularizer=regularization,use\_bias=False)(x)

x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

x = layers.add([x, residual])

# module 4

residual = Conv2D(128, (1, 1), strides=(2, 2),padding='same', use\_bias=False)(x)

residual = BatchNormalization()(residual)

x = SeparableConv2D(128, (3, 3), padding='same',kernel\_regularizer=regularization,use\_bias=False)(x)

x = BatchNormalization()(x)

x = Activation('relu')(x)

x = SeparableConv2D(128, (3, 3), padding='same',kernel\_regularizer=regularization,use\_bias=False)(x)

x = BatchNormalization()(x)

x = MaxPooling2D((3, 3), strides=(2, 2), padding='same')(x)

x = layers.add([x, residual])

x = Conv2D(num\_classes, (3, 3), padding='same')(x)

x = GlobalAveragePooling2D()(x)

output = Activation('softmax',name='predictions')(x)

model = Model(img\_input, output)

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

model.summary()

# callbacks

log\_file\_path = base\_path + '\_emotion\_training.log'

csv\_logger = CSVLogger(log\_file\_path, append=False)

early\_stop = EarlyStopping('val\_loss', patience=patience)

reduce\_lr = ReduceLROnPlateau('val\_loss', factor=0.1,

patience=int(patience/4), verbose=1)

trained\_models\_path = base\_path + '\_mini\_XCEPTION'

model\_names = trained\_models\_path + '.{epoch:02d}-{val\_acc:.2f}.hdf5'

model\_checkpoint = ModelCheckpoint(model\_names, 'val\_loss', verbose=1,

save\_best\_only=True)

callbacks = [model\_checkpoint, csv\_logger, early\_stop, reduce\_lr]

# loading dataset

faces, emotions = load\_fer2013()

faces = preprocess\_input(faces)

num\_samples, num\_classes = emotions.shape

xtrain, xtest,ytrain,ytest = train\_test\_split(faces, emotions,test\_size=0.2,shuffle=True)

model.fit\_generator(data\_generator.flow(xtrain, ytrain,

batch\_size),

steps\_per\_epoch=len(xtrain) / batch\_size,

epochs=num\_epochs, verbose=1, callbacks=callbacks,

validation\_data=(xtest,ytest))

**REAL TIME VIDEO.PY:-**

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

import imutils

import cv2

from keras.models import load\_model

import numpy as np

from playsound import playsound

import time

# 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","sad","happy","scared","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:

for i in range(0,200):

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 via the CNN

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]

print(preds)

emotion\_probability = np.max(preds)

label = EMOTIONS[preds.argmax()]

# try:

for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):

# construct the label text

if prob > 0.3:

print(emotion)

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

print(text)

# 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)

cv2.imshow('your\_face', frameClone)

cv2.imshow("Probabilities", canvas)

if cv2.waitKey(1) & 0xFF == ord('q'):

break

if emotion == 'happy':

playsound('SONGS/happy/muthumazhyay.mp3')

playsound('SONGS/happy/koodemele.mp3')

playsound('SONGS/happy/Ale-Ale.mp3')

playsound('SONGS/happy/Ranjithame-MassTamilan.dev.mp3')

playsound('SONGS/happy/otha-sollaala.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'sad':

playsound('SONGS/sad/enn-kathalle.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'neutral':

playsound('SONGS/neutral/Engaum-Epothum.mp3')

playsound('SONGS/neutral/kanna-nee-thoogada.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'scared':

playsound('SONGS/scared/Chandramukhi.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'surprised':

playsound('SONGS/surprised/poona-usiru.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'angry':

playsound('SONGS/angry/Kalippu-Premam.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

if emotion == 'fear':

playsound('SONGS/fear/agayam.mp3')

camera.release()

cv2.destroyAllWindows()

time.sleep(20)

break

# for c in range(0, 3):

# frame[200:320, 10:130, c] = emoji\_face[:, :, c] \* \

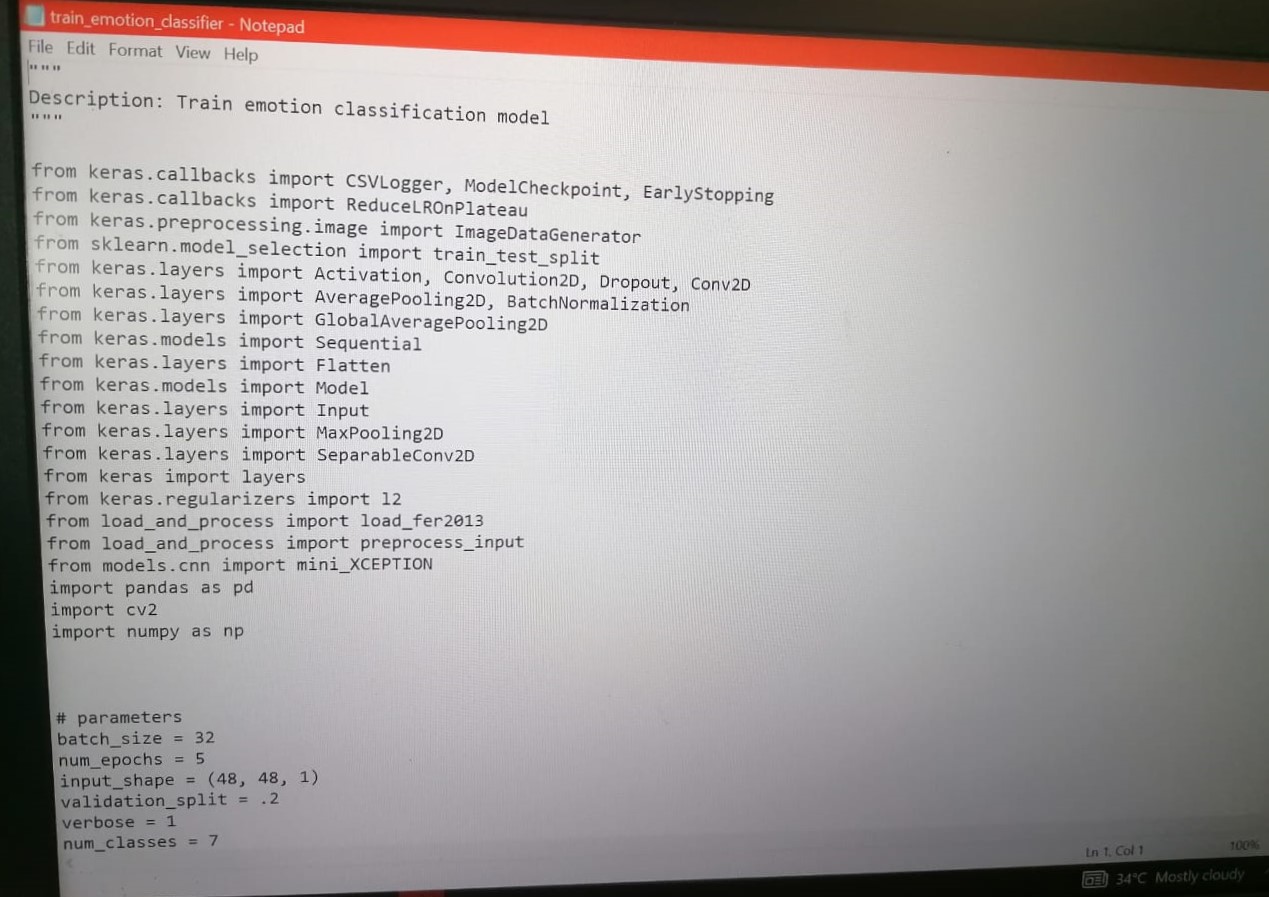
# (emoji\_face[:, :, 3] / 255.0) + frame[200:320,

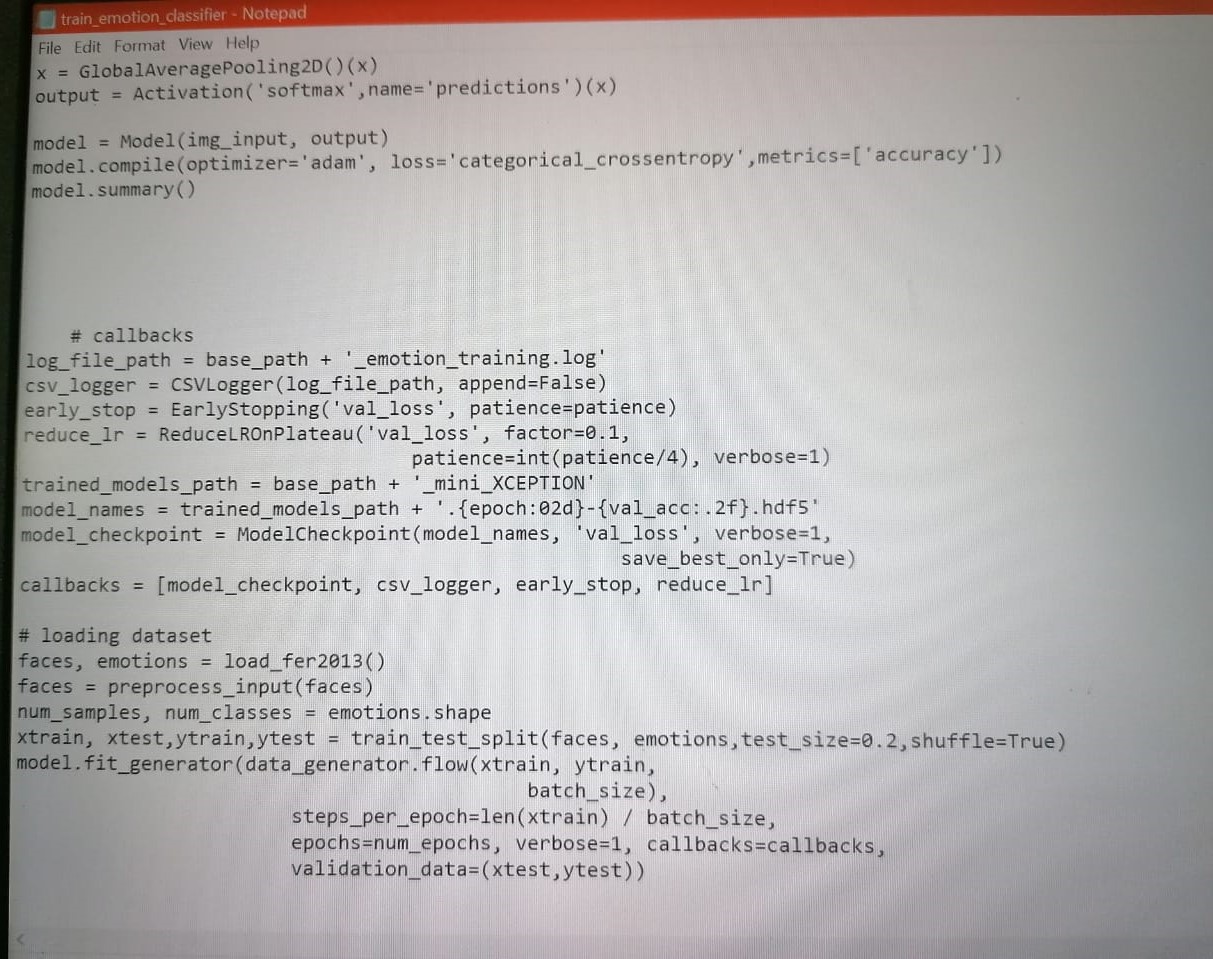
# 10:130, c] \* (1.0 - emoji\_face[:, :, 3] / 255.0)

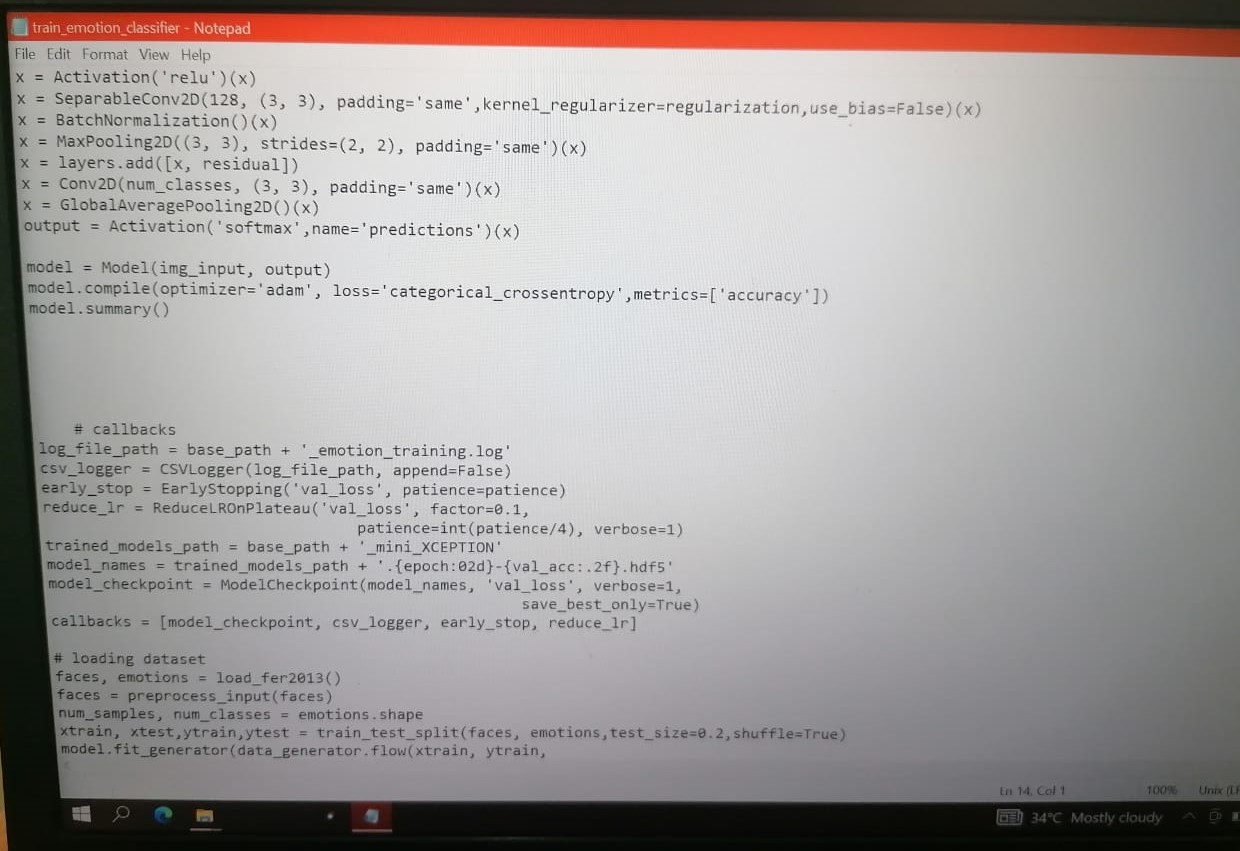
# except :

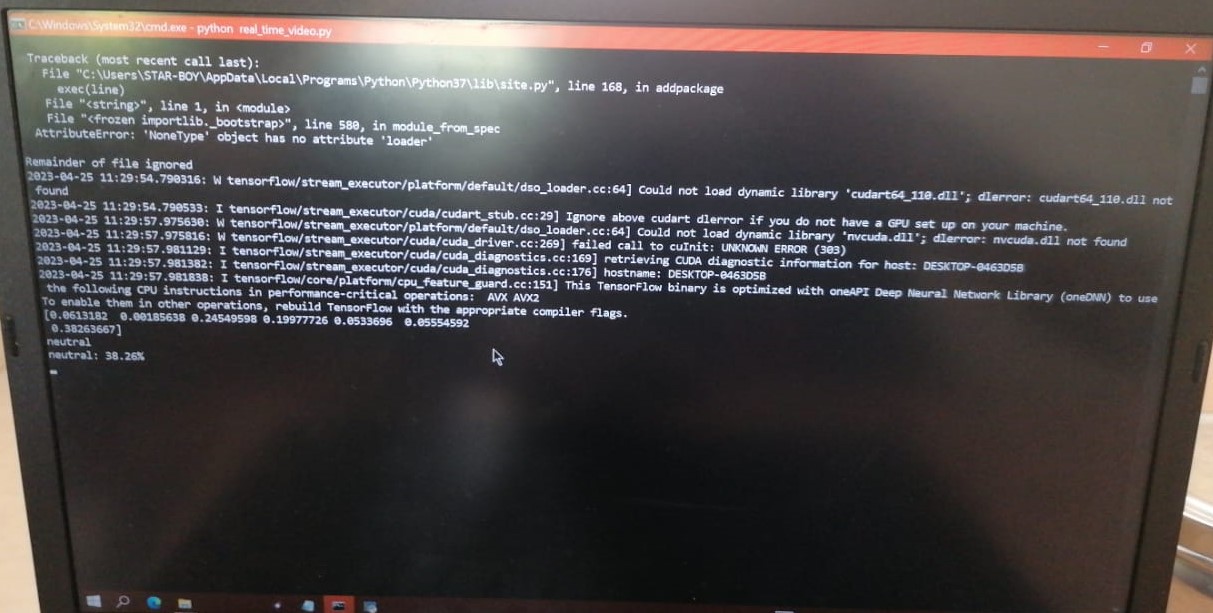
# er = 'error'

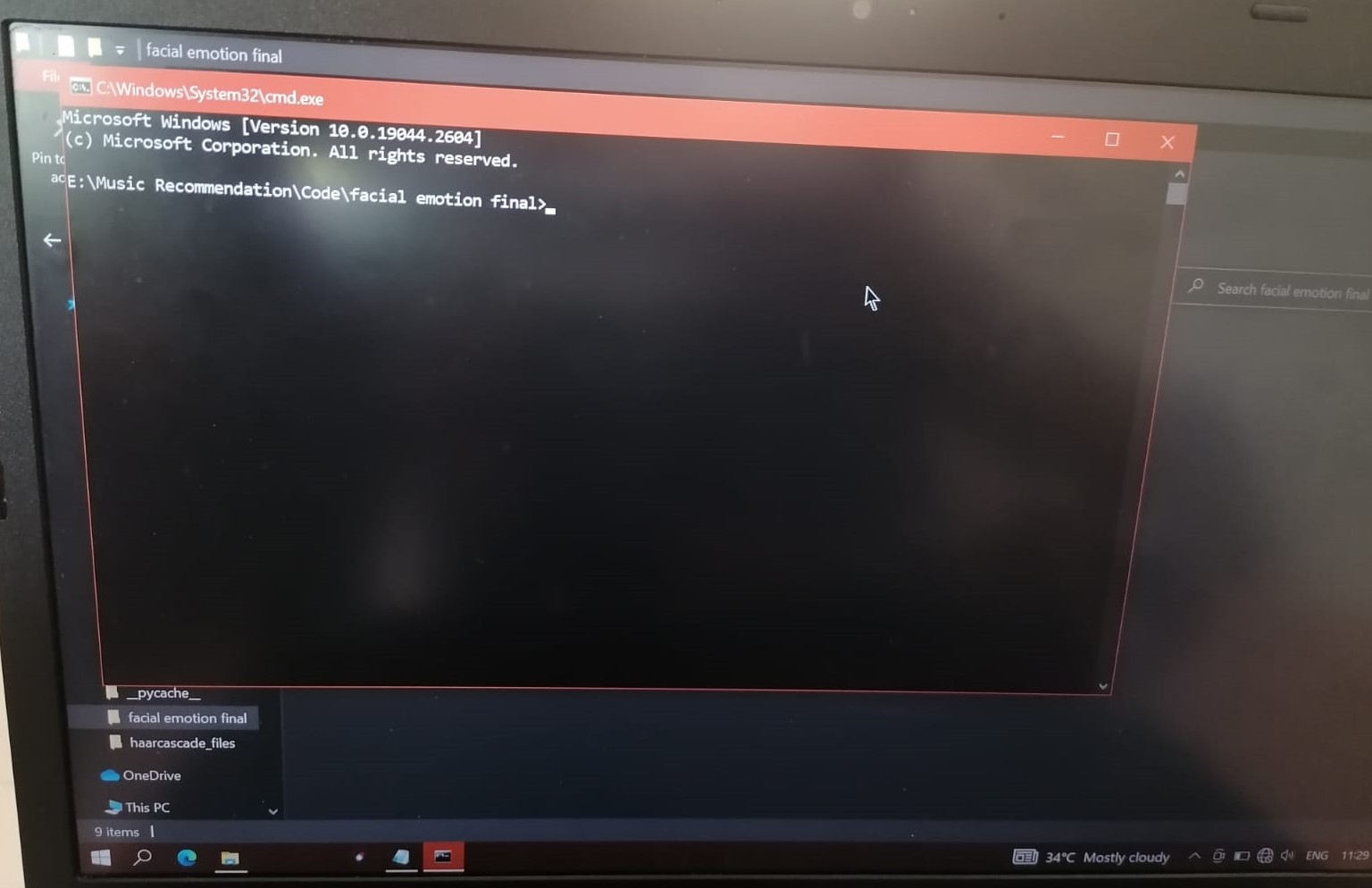
**11.SCREENSHOTS:-**

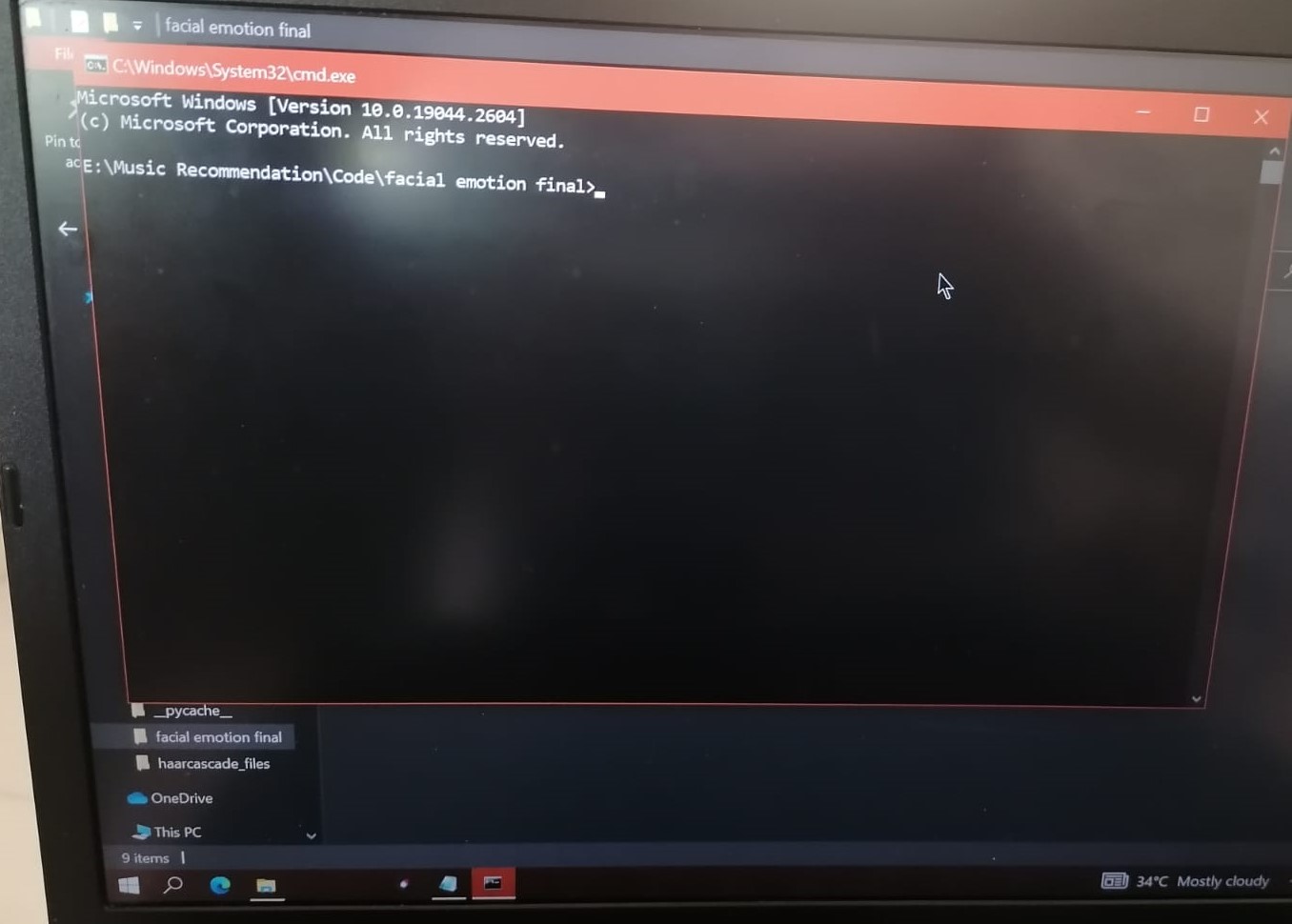


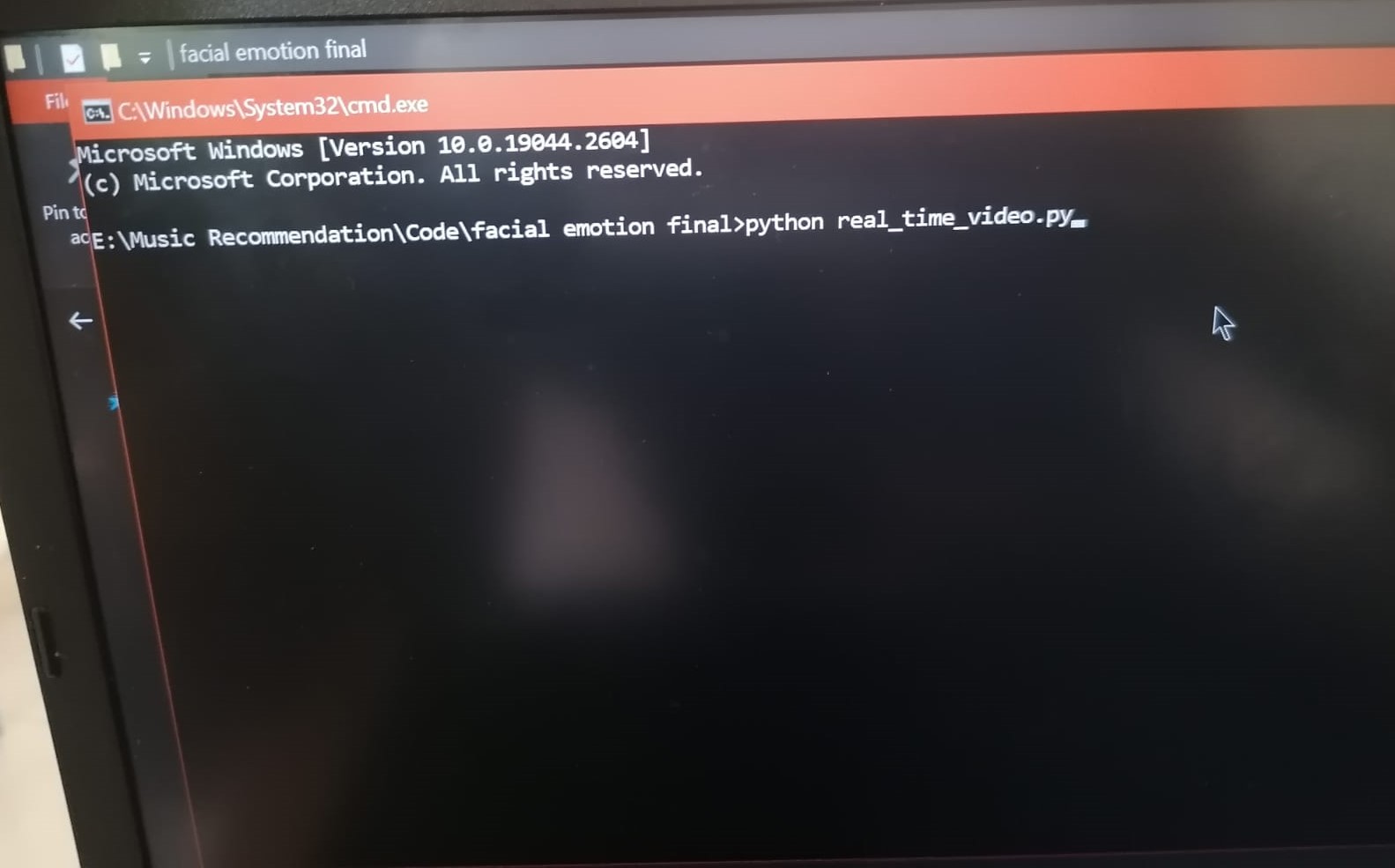


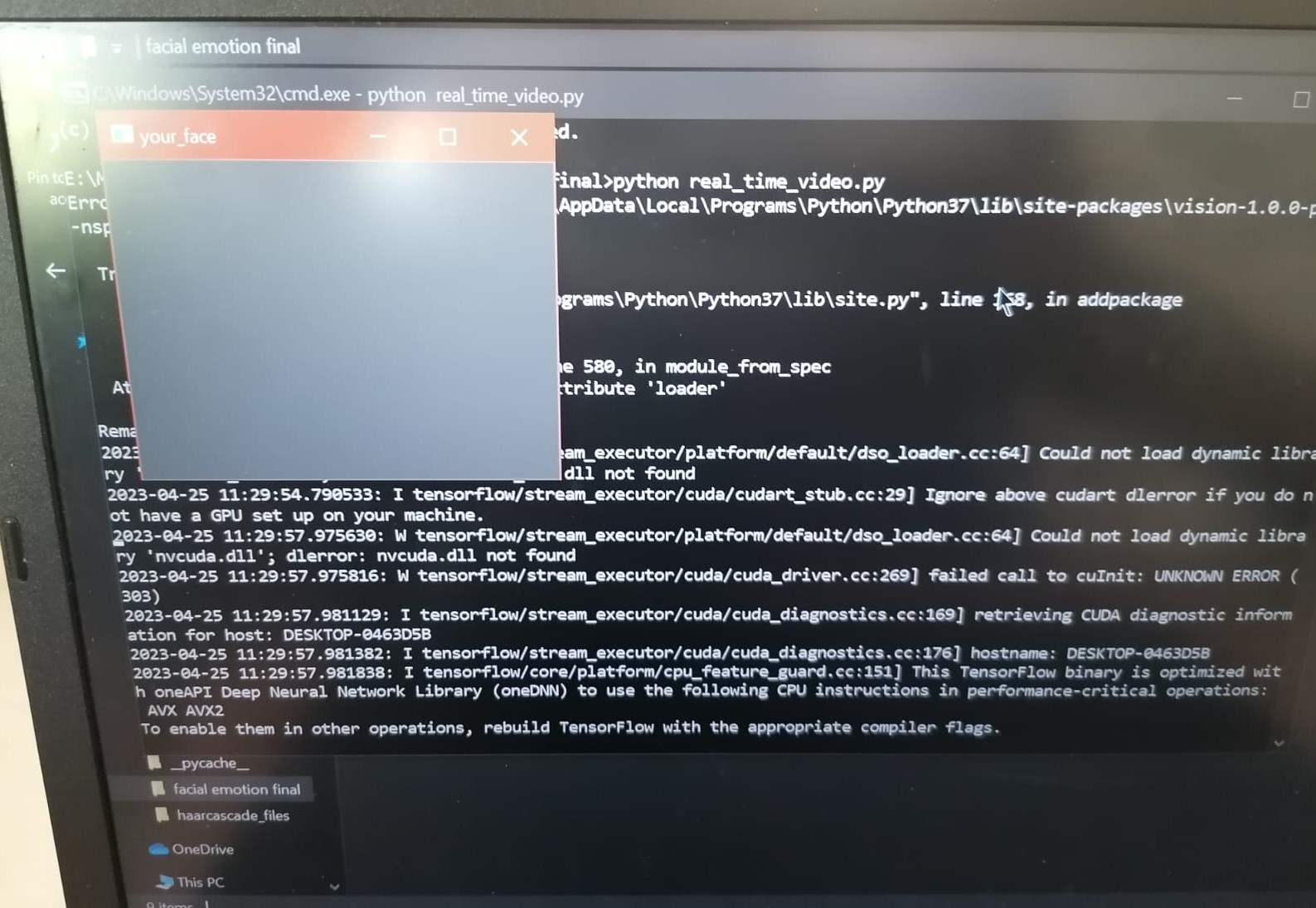


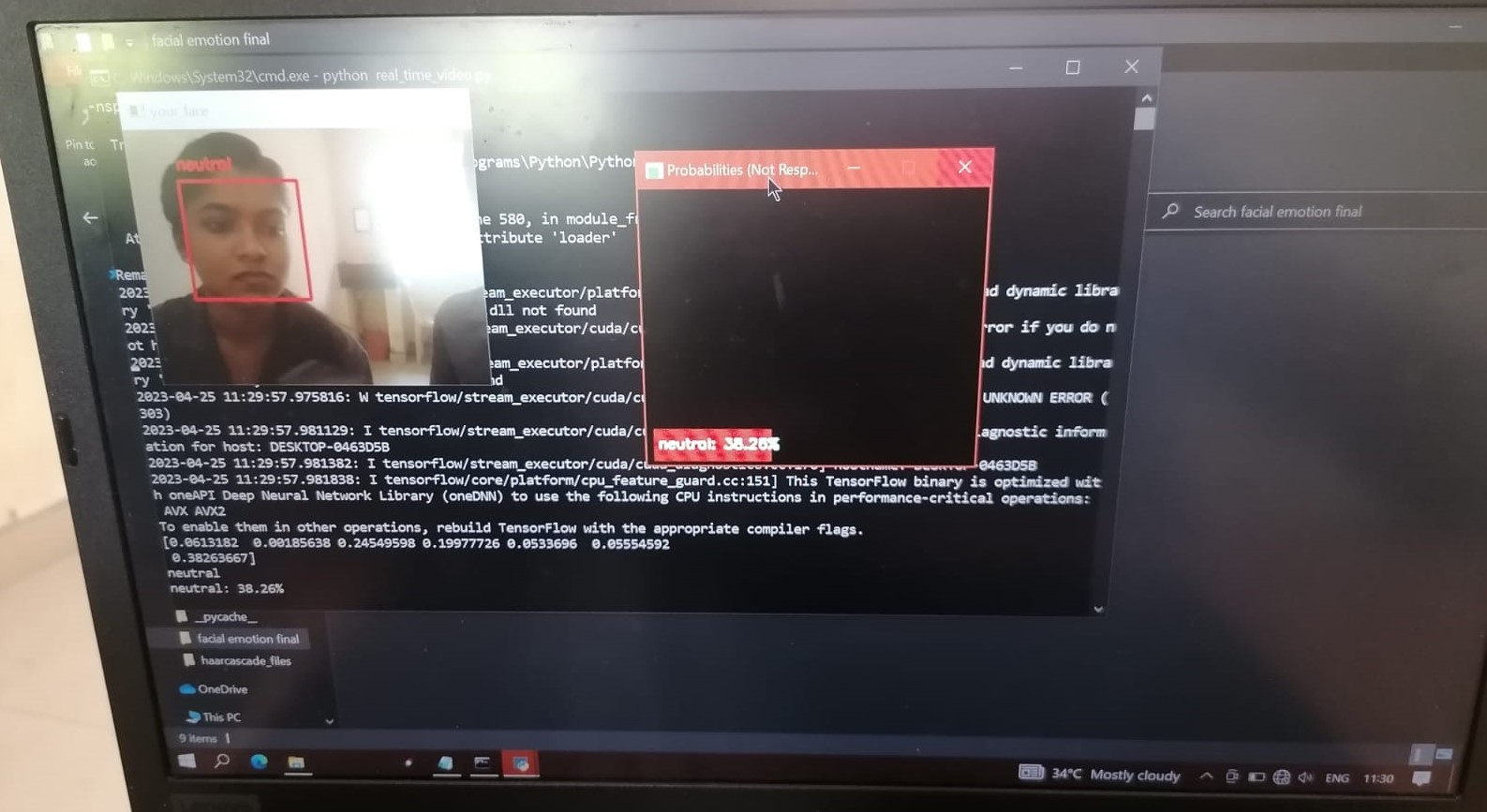


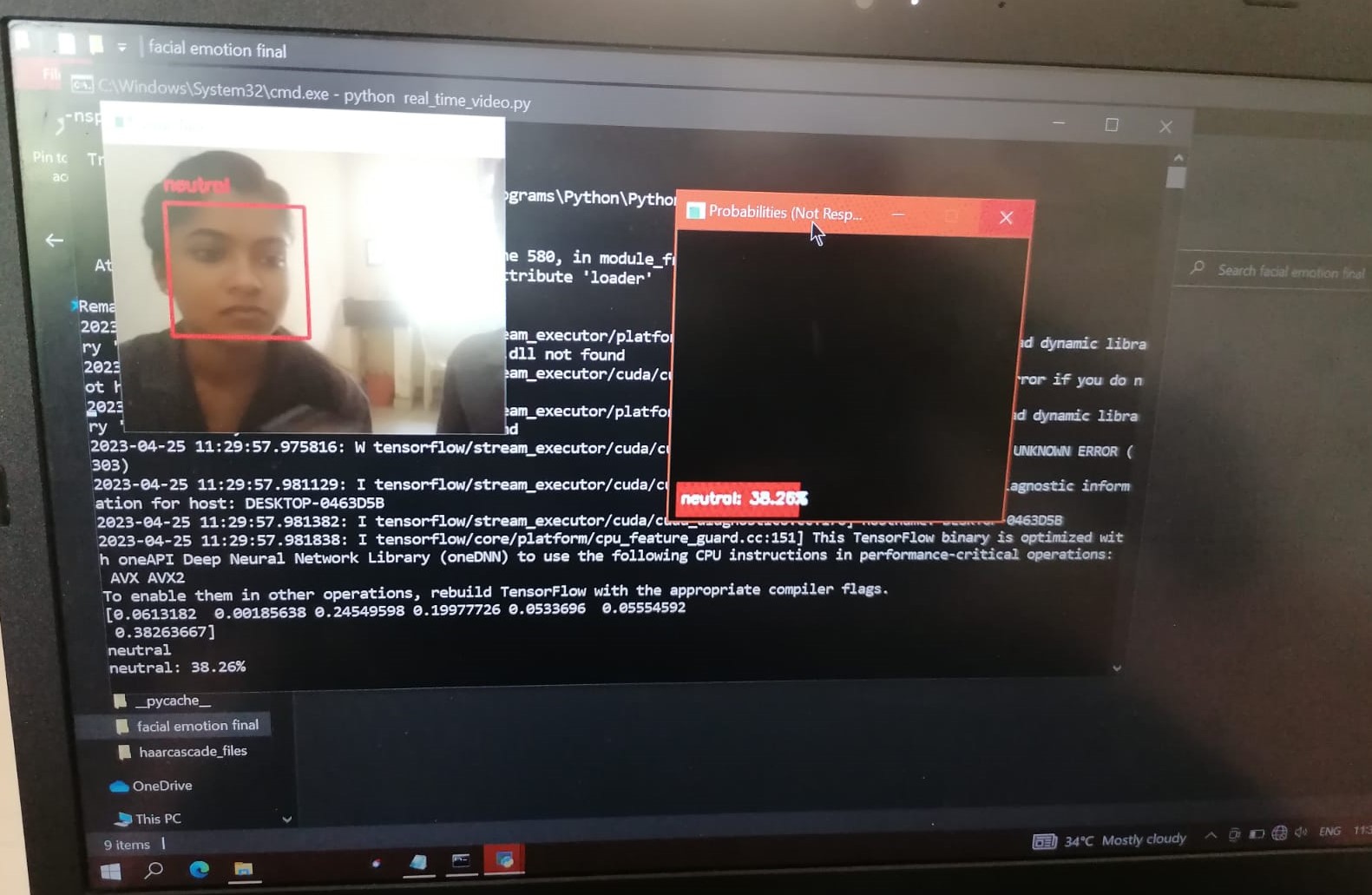












**12. CONCLUSION:-**

A thorough review of the literature tells that there are many approaches to implement Music Recommender System. A study of methods proposed by previous scientists and developers was done. Based on the findings, the objectives of our system were fixed. As the power and advantages of AI-powered applications are trending, our project will be a state-of-the-art trending technology utilization. In this system, we provide an overview of how music can affect the user's mood and how to choose the right music tracks to improve the user's moods. The implemented system can detect the user's emotions. The emotions that the system can detect were happy, sad, angry, neutral, or surprised. After determining the user’s emotion, the proposed system provided the user with a playlist that contains music matches that detected the mood. Processing a huge dataset is memory as well as CPU intensive. This will make development more challenging and attractive. The motive is to create this application in the cheapest possible way and also to create it under a standardized device. Our music recommendation system based on facial emotion recognition will reduce the efforts of users in creating and managing playlists.

**13.Future Scope:-**

This system, although completely functioning, does have scope for improvement in the future. There are various aspects of the application that can be modified to produce better results and a smoother overall experience for the user. Some of these that an alternative method, based on additional emotions which are excluded in our system as disgust and fear. This emotion included supporting the playing of music automatically. The future scope within the system would style a mechanism that might be helpful in music therapy treatment and help the music therapist to treat the patients suffering from mental stress, anxiety, acute depression, and trauma. The current system does not perform well in extremely bad light conditions and poor camera resolution thereby provides an opportunity to add some functionality as a solution in the future.

**14. Reference:-**

[1] RamyaRamanathan, RadhaKumaran, Ram Rohan R, Rajat Gupta, and VishalakshiPrabhu, an intelligent music player based on emotion recognition, 2nd IEEE International Conference on Computational Systems and Information Technology for Sustainable Solutions 2017. https://doi.org/10.1109/CSITSS.2017.8447743

[2] Shlok Gilda, Husain Zafar, ChintanSoni, KshitijaWaghurdekar, Smart music player integrating facial emotion recognition and music mood recommendation, Department of Computer Engineering, Pune Institute of Computer Technology, Pune, India, (IEEE),2017. <https://doi.org/10.1109/WiSPNET.2017.8299738>

[3] DegerAyata, Yusuf Yaslan, and Mustafa E. Kamasak, Emotion-based music recommendation system using wearable physiological sensors, IEEE transactions on consumer electronics, vol. 14, no. 8, May 2018.https://doi.org/10.1109/TCE.2018.2844736

[4] AhlamAlrihail, AlaaAlsaedi, KholoodAlbalawi, Liyakathunisa Syed, Music recommender system for users based on emotion detection through facial features, Department of Computer Science Taibah University, (DeSE), 2019. <https://doi.org/10.1109/DeSE.2019.00188>

[5] Research Prediction Competition, Challenges in representation learning: facial expression recognition challenges, Learn facial expression from an image, (KAGGLE).

[6] Preema J.S, Rajashree, Sahana M, Savitri H, Review on facial expression-based music player, International Journal of Engineering Research & Technology (IJERT), ISSN-2278-0181, Volume 6, Issue 15, 2018.

[7] AYUSH Guidel, BiratSapkota, Krishna Sapkota, Music recommendation by facial analysis, February 17, 2020.

[8] CH. sadhvika, Gutta.Abigna, P. Srinivasreddy, Emotion-based music recommendation system, Sreenidhi Institute of Science and Technology, Yamnampet, Hyderabad; International Journal of Emerging Technologies and Innovative Research (JETIR) Volume 7, Is-sue 4, April 2020.

[9] Vincent Tabora, Face detection using OpenCV with Haar Cascade Classifiers, Becominghuman.ai,2019.

[10] Zhuwei Qin, Fuxun Yu, Chenchen Liu, Xiang Chen. How convolutional neural networks see the world - A survey of convolutional neural network visualization methods. Mathematical Foundations of Computing, May 2018.

[11] Ahmed HamdyAlDeeb, Emotion- Based Music Player Emotion Detection from Live Camera, ResearchGate, June 2019.

[12] FransNorden and Filip von Reis Marlevi, A Comparative Analysis of Machine Learning Algorithms in Binary Facial Expression Recognition, TRITA-EECS-EX-2019:143.

[13] P. Singhal, P. Singh and A. Vidyarthi (2020) Interpretation and localization of Thorax diseases using DCNN in Chest X-Ray. Journal of Informatics Electrical and Elecrtonics Engineering,1(1), 1, 1-7

[14] M. Vinny, P. Singh (2020) Review on the Artificial Brain Technology: BlueBrain. Journal of Informatics Electrical and Electronics Engineering,1(1), 3, 1-11.

[15] A. Singh and P. Singh (2021) Object Detection. Journal of Management and Service Science, 1(2), 3, pp. 1-20.