AN INDUSTRIAL ORIENTED MINI PROJECT REPORT ON

EMOTION BASED MUSIC PLAYER

Is Submitted to Jawaharlal Nehru Technology University, Hyderabad,
In partial fulfillment of requirement for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

INFORMATION TECHNOLOGY

SUBMITTED BY

SAI NITHYA AKUTHOTA (19J21A1201)

Under the guidance of

Mr. P. SRINIVAS

Associate Professor

Department of Information Technology

JOGINPALLY B.R. ENGINEERING COLLEGE

Accredited by NAAC with A+ Grade, Recognized under Sec. 2(f) of UGC Act. 1956
Approved by AICTE and Affiliated to Jawaharlal Nehru Technological University, Hyderabad
Bhaskar Nagar, Yenkapally, Moinabad,
Ranga Reddy, Hyderabad, Telangana - 500075.

2019-2023

JOGINPALLY B.R ENGINEERING COLLEGE

Accredited by NAAC with A+ Grade, Recognized under Sec. 2(f) of UGC Act. 1956

Approved by AICTE and Affiliated to Jawaharlal Nehru Technological University, Hyderabad

Bhaskar Nagar, Yenkapally, Moinabad,

Ranga Reddy, Hyderabad, Telangana - 500075.

CERTIFICATE

This is to certify that an Industrial Oriented Mini Project entitled "**EMOTION BASED MUSIC PLAYER**" is the bonafide work carried out by **SAI NITHYA AKUTHOTA (19J21A1201)** of **IV B.Tech (INFORMATION TECHNOLOGY)** under our guidance and supervision. The Mini Project Report is submitted to JNTU Hyderabad in partial fulfillment of requirements for the award of degree of Bachelor of Technology in Information Technology, during the academic year 2022-2023.

INTERNAL GUIDE

Mr. P. SRINIVAS

Associate Professor

HEAD OF THE DEPARTMENT

Mr. P. SRINIVAS

Associate Professor

EXTERNAL EXAMINER

PRINCIPAL

ACKNOWLEDGMENT

I would like to take this opportunity to place it on record that an Industrial Oriented Mini Project would never have taken shape but for the cooperation extended to me by certain individuals. Though it is not possible to name all of them, it would be unpardonable on my part if I do not mention some of the very important persons.

I express our gratitude to **Dr. B. VENKATA RAMANA REDDY**, Principal of **JOGINPALLY B.R. ENGINEERING COLLEGE** for valuable suggestions and advice. I also extend our thanks to other faculty members for their cooperation during our Mini Project.

I express our gratitude to **Mr P. SRINIVAS,** HOD of **INFORMATION TECHNOLOGY**

for valuable suggestions and advice.

Sincerely, I acknowledge my deep sense of gratitude to my Mini Project guide, **Mr. P. SRINIVAS**, Associate Professor for his constant encouragement, help and valuable suggestions.

SAI NITHYA AKUTHOTA (19J21A1201)

DECLARATION

I hereby declare that my Industrial Oriented Mini Project entitled "**EMOTION BASED MUSIC PLAYER** "is the work done during the academic year **2022-2023** and my Mini Project is submitted in partial fulfilment of the requirements for the award of degree of **Bachelor of Technology** in Information Technology to the **Jawaharlal Nehru Technology University,** Hyderabad.

SAI NITHYA AKUTHOTA (19J21A1201)

ABSTRACT

Human expression plays a vital role in determining the current state and mood of an individual, it helps in extracting and understanding the emotion that an Individual has based on various features of the face such as eyes, cheeks, forehead or even through the curve of the smile. Music is basically an art form that soothes and calms human brain and body. Taking these two aspects and blending them together our project deals with detecting emotion of an individual through facial expression and playing music according to the mood detected that will alleviate the mood or simply calm the individual and can also get quicker song according to the mood, saving time from looking up different songs and parallel developing a software that can be used anywhere with the help of providing the functionality of playing music according to the emotion detected. By developing a recommendation system, it could assist a user to make a decision regarding which music one should listen to helping the user to reduce his/her stress levels. The image of the user is captured with the help of a webcam. The user's picture is taken and then as per the mood/ emotion of the user an appropriate song from the playlist of the user is shown matching the user's requirement.

CONTENTS

LIST OF FIGURES

1. INTRODUCTION

Photo processing is a way where in the pixels are pre-processed and the capabilities are extracted out by using appearing diverse operations. There are literally styles of picture processing specifically, digital and analog photo processing. The analog pix are pictures which are in hard copies.

Example: scientific reports and pictures. Virtual Photo processing uses diverse algorithms to perform photograph mannering on various digital pics. There are some strategies worried in virtual image processing. They are photograph enhancing, photo restoration, independent thing analysis linear filtration. Photograph processing also consists of exclusive steps like photograph enhancement restoration, and many others. Music is a melody that connects the soul and mind of the character together. It performs a critical function in human lifestyles. Feelings affect us physically as well as mentally. Our body reacts to one of a kind emotional state. The sturdy feelings are brought out by using listening to the track according to their scenario.

2. LITERATURE SURVEY

1. Emotion Detection of Autistic Children Using Image Processing

Abstract:

Facial Emotion Detection is an approach towards detecting human emotions through facial expressions. Autism Spectrum Disorder is an advance neurobehavioral disorder. Autistic people have repetitive, rude behavior. They are not ready to do social communication. People with this syndrome have problems with emotion recognition. This paper works on detecting the emotions of autistic children from the expression of their faces. This paper works on four emotions. These emotions are sad, happy, neutral, and angry. To detect the emotion of autistic children is performed with image processing and machine learning algorithms. The features are extracted from the faces of autistic children with local binary pattern. Machine learning algorithms are used for classification of emotions. Machine learning classifiers used in classification process are support vector machine and neural network.

Author: Pooja Rani

YEAR: 2020

- Machine Learning,
- Support Vector Machine,
- Neural Network,
- Emotion Detection,
- Image Processing,
- Local Binary Pattern

2. A Review on Different Facial Feature Extraction Methods for Face Emotions Recognition System

Abstract:

Emotion recognition using facial images is the state-of-the-art research area in the human-computer interaction paradigm. In order to recognize emotions from images, the system needs to extract facial features like mouth, eyes, etc. Emotions can also be extracted from frontal and

non-frontal images. Traditional methods used for extracting facial features are geometry based method, template based method, and appearance based method. The main focus of this paper is to review different types of facial feature extraction method and this research article illustrates a comparison between different methods and in the end, it describes some future research works that would be helpful to make FER more reliable and efficient.

Author: Viha Upadhyay; Devangi Kotak

Year: 2020

3. Multi Label Classification for Emotion Analysis of Autism Spectrum Disorder Children using Deep Neural Networks

Abstract:

Emotion recognition and analysis is the process of identifying emotions and feelings of a person. Emotion analysis process is accurate in identifying expression in normal people in a single attempt. Emotion analysis is difficult in case of Autism Spectrum Disorder (ASD) children which are suffering with communication problems and speech problems. This paper proposed an optimized deep learning model with multi label classification for predicting ASD and NoASD with emotion analysis in children of age group 1 to 10 years. The kaggle dataset [1] of 1857 ASD children and 1850 Typically Developed (TD) children are used in this paper. Proposed model performance is tested on Yale Expression Dataset [2], CAFÉ children dataset [3] and also tested on social media dataset of autism parents group. The model is implemented by extracting face landmarks and is used to predict ASD and NoASD as first classification label and emotion is detected based on landmarks by computing internal and external distances by feature wise. Convolutional Neural Networks (CNN) is used to work with extracted face landmarks by using optimization methods, dropout, batch normalization and parameter updating. The proposed model is applied to predict 6 emotions irrespective of 4 general emotions with better accuracy.

AUTHOR: T. Lakshmi Praveena; N.V.Muthu Lakshmi

Year: 2021

4. Image Processing Methods for Face Recognition using Machine Learning Techniques

Abstract:

The face is one of the simplest ways to distinguish one another's personal image. Face recognition is a personal identification system which uses a person's personal features to recognize the identity of the individual. Human facial identification is basically a two-phase procedure, including face detection, where the process is carried out very rapidly in people, whereas the second is the implementation of environments that classify the face as persons, when the eye is positioned within a short distance. Stage is then repeated and established to be one of the most researched biometric strategies and established by experts for facial expression recognition. In this study, we implemented the area of face detection and face recognition image processing MTCNN techniques while utilizing the VGG face model dataset. In this initiative, python framework is the program necessity.

AUTHOR: T. R. Ganesh Babu; K. Shenbagadevi; V. Sri Shoba; S. Shrinidhi; J. Sabitha; U. Saravanakumar

YEAR: 2021

5. Virtual Markers based Facial Emotion Recognition using ELM and PNN Classifiers

Abstract:

Detecting different types of emotional expressions from the subject's face is important for developing intelligent systems for a variety of applications. This present work proposed virtual markers based on Facial emotion expression recognition using the Extreme Learning Machine (ELM) and Probabilistic Neural Network (PNN). A facial emotional expression database is developed with 55 undergraduate university students (male: 35, female: 20) of age range between 20 - 25 years with a mean age of 23.9 years. A HD webcam is used to capture the facial image and Haar Like features and Ada Boost classifier is used to detect the face and eyes through Open CV. A mathematical model based is used to place ten virtual markers called Action Units (AUs) on subjects face at a defined location. Later, Lucas-kanade optical flow

algorithm is used to track the marker movement while the subject expressing different emotions and the distance between the center of the face to each marker is used as a feature for classifying emotions. One way Analysis of Variance (ANOVA) is used to test the statistical significance of the features and five fold cross-validation method is used to input the feature for classifiers. In this work, two non-linear classifiers namely, ELM and PNN are used for emotional expression classification. The experimental results give a maximum mean emotion classification rate of 88% and 92% in ELM and PNN classifiers, respectively. Maximum individual class accuracy of happiness - 96%, surprise - 94%, anger- 92%, sadness - 88%, disgust - 90% and fear 89% is achieved using PNN. The experimental results confirm that the proposed system is able to distinguish six different emotional expressions and could be used as a potential tool for a variety of applications which include, e-learning, pain assessment, psychological counseling, human-machine interaction-based applications, etc.

Author: M Murugappan; Vasanthan Maruthapillai; Wan Khariunizam; A M Mutawa; Sai Sruthi; Chong Wen Yean

YEAR:2021

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Mood Player

This app makes use of face detection and temper reputation to work ,and supported this; it offers a personalized play listing. The face detection rule is predicted on Open CV library and additionally the mood detection half of is going to be based on pattern matching if we all understand he know-how that is required, we generally tend to use the remaining. Information that joins every song with tags that describe it. These implementations rectangular degree designed so one can get a list in keeping with the person temper's and provide these functionalities:

1. Set your mood manually i.e., Happy or Sad

2. Analyzing your mood periodically by capturing camera

3. Set tempo of the music from Calm to Energetic.

Stereo Mood

Stereo temper can be a cellular or pill utility. With the press of a button, we'll have a readymade play list for every time in our life. We'll choose our temper from our tags, concentrate, find out new music, share and tag our emotions in tune. This application gives the subsequent functionalities:

- Sharing your mood through social network like WhatsApp, Snapchat.
- Labelling the particular song with different user defined names.
- Upgrading the User's profile with its unique feature algorithm.

DISADVANTAGES OF EXISTING SYSTEM:

- It requires the user to manually select the songs.
- Randomly played songs may not match to the mood of the user.
- User has to classify the songs into various emotions and then for playing the songs user has to manually select a particular emotion proposed system

The proposed system tries to provide an interactive way for the user to carry out the task of creating a playlist. the working is based on different mechanisms carrying out their function in a predefined order to get the desired output. The working can be stated as follows:

- Proposed system upgrades with a unique Ul which enables the user to get files of audio.
- Files are being detected, they are scanned for audio features which it's extracted via Internet or Author's gallery.
- The extracted feature values are subjected to label the classified feature value with its parameters provided.
- These parameters include a limited set of genre types based on which the audio feature values will be processed.
- After this, the songs are segregated into different playlists based on the feature extraction process. Hence lists of similar sounding songs or songs belonging to similar genres are generated.

3.2 PROPOSED SYSTEM

The proposed system can detect the facial expressions of the user and based on his/her facial expressions extract the facial landmarks, which would then be classified to get a particular emotion of the user. Once the emotion has been classified the songs matching the user's emotions would be shown to the user.

Thus, our proposed system focuses on detecting human emotions for developing emotion-based music player. A brief idea about our systems working, playlist generation and classification.

In this system live face emotion is detected using webcam where in the facial expressions are detected, captured and classified into their respective type accordingly the system plays music from the collected facial expressions. This system makes use of Convolutional Neural Network (CNN) model for image classification. The model trains itself according to the results thus improving the efficiency and effectiveness of the system.

Emotion base Music Player is a useful application for music listeners with a smart phone and an internet connection. The Application is accessible by anyone who creates a Profile on the system. The Application is designed to meet the following needs of the users as described below;

Detected emotions:

Happy Sad Angry Neutral

Figure 1: Emotions

According, to which we can classify emotion directory for playing song we have chosen this 4 emotions

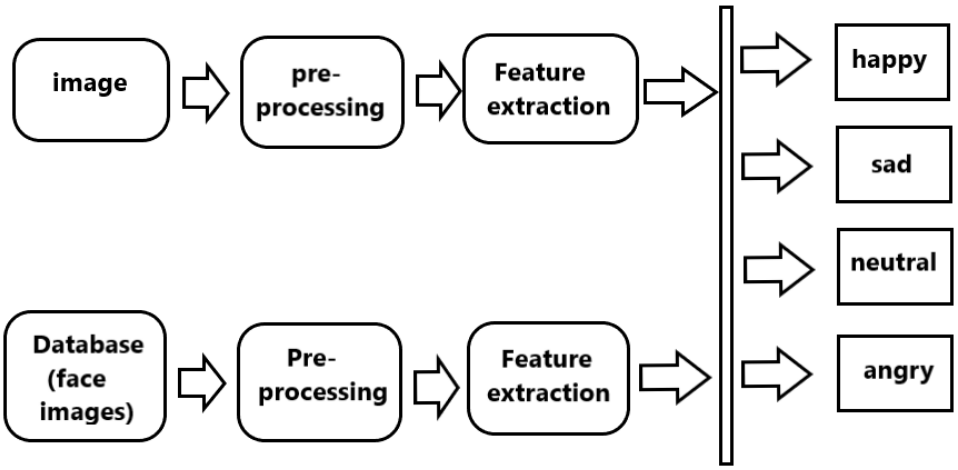

Figure 2: Emotion Classification

ADVANTAGES OF PROPOSED SYSTEM

The proposed system can detect the facial expressions of the user and based on his/her facial expressions extract the facial landmarks, which would then be classified to get a particular emotion of the user. Once the emotion has been classified the songs matching the user's emotions would be shown to the user. Throughout the years, the results from the studies proved that different music style can actually influence individuals in different ways. Our proposed system simply calm the individual and can also get quicker song according to the mood, saving time from looking up different songs.

3.3 MODULES

Module 1: - Emotion Extraction Feature

The photograph of the person is captured with the assist of a digital camera / web-cam. Once image captured, the frame of the captured picture from webcam feed is converted to a grayscale image to enhance performance of the classifier, that's used to perceive the face gift within the image as soon as the conversion is complete, the photograph is dispatched to the classifier set of rules which, with the assist of characteristic extraction strategies can extract the face from the body of the net digicam feed

From the extracted face, individual functions are obtained and are sent to the trained network to come across the emotion expressed by the way of the person. Those photos may be use to train the classifier so that when a completely new and unknown set of snap shots is offered to the classifier, it can extract the location of facial landmarks from the one's photos based totally on the know-how that it had already acquired from the training set and return the coordinates of the new facial landmarks that it detected. The network is educated with the assist of CK significant records set. This is used to perceive the emotion being voiced through the person. It's achieved by using OpenCV.

Module 2: - Audio Extraction Feature

In this module, a listing of songs paperwork the input. As songs are audio documents, they require a positive amount of pre-processing Stereo indicators obtained from the internet are transformed to 16-bit PCM mono sign around a variable sampling rate of 48.6 kHz. The conversion system is finished the usage of Audacity method. The pre- processed signal obtained undergoes an audio characteristics extraction, where in features like rhythm firming is extracted using MIR 15 Toolbox, pitch is extracted using Chroma Toolbox and different capabilities like centroid, spectral flux. spectral roll off, kurtosis, 15 MECC coefficients are extracted the use of Auditory Toolbox. Audio alerts are categorized into 7 kinds viz unhappy, marvel, excitement, pleasure, anger, satisfied, lonely. The extractions of converted files are carried out by using three device container units. MIR 15 Toolbox, Chroma Toolbox, and Auditory Toolbox which might be inbuilt in Haar Casade Classifier.

Module 3 :- Emotion Audio Integration

Emotions extracted for the songs are saved as a meta-information within the database. Mapping is 18 executed by way or querying the meta- statistics database. The emotion extraction module and audio feature extraction module is eventually mapped and mixed the use of an Emotion-Audio integration module.

4. FEASIBILITY STUDY

Feasibility observes pursuits to objectively and rationally uncover the strengths and weaknesses of an existing or proposed system, opportunities and threats gift inside the environment the sources required to carry thru, and ultimately the prospects for achievement. In its only phrases, the two criteria to judge feasibility are cost required and fee to be attained. A properly designed feasibility has a look at should offer a historical background of the task: typically, feasibility research precedes technical development and assignment implementation. A feasibility looks at evaluates the challenge capability for success.

3 key issues involved within the feasibility evaluation are: -

- Economic Feasibility
- Technical Feasibility
- Social Feasibility

4.1 Economic Feasibility

This examine is achieved to check the economic effect that the system could have on the corporation. The amount of fund t hat the organization can pour into the work and improvement of the gadget is confined. The costs must be justified. hence the advanced system as well in the f the gadget i t be justified. hence the advanced system as well in the budget and this turned into completed due to the fact most of the technologies used are freely to be had.

4.2 Technical Feasibility

This exam is finished to test the specialized achievable, the specialized requirements of the framework. The created bodywork has to have an unassuming necessity; because it was negligible or in-valid adjustments are required for actualizing this framework.

4.3 Social Feasibility

The aspect of observe is to che ck the extent of popularity of the system by using the person. This includes the process of training the consumer to use the gadget effectively. The person should no longer feel threatened through the system, as an alternative should be given it as need. the extent of attractiveness by means of the customers completely relies upon at the strategies which might be employed to teach the consumer approximately the gadget and to

make him acquainted with it. His level of confidence should be raised so that he’s also able to make a few constructive complaints, that is welcomed, as he is the final user of the gadget.

5. SOFTWARE REQUIREMENTS SPECIFICATIONS

5.1 SOFTWARE REQUIREMENTS

Operating System : Windows 8 or later, macOS, Linux

Technologies : **Python-**

- Python 3.95
- PIP and NumPy: Installed with PIP, Ubuntu*, Python 3.95, NumPy 1.20.3
- IOS/Mac: Python 3.95

Open CV-

- Open CV 4.5.2
- Open CV: Open CV 4.5.2 with installed HaarCasade Library of OpenCL1
- IOS/Mac: Open CV 4.5.2

Browser : Chrome 91 or above

Edge 91 or above

5.2 HARDWARE REQUIREMENTS:

Processor : Intel Atom® processor or Intel® Core™ i5 processor 8th gen

Hard disk : 160 GB

RAM : 10 GB

Graphics : Nvidia GeForce 949M or later, Intel Graphics 15.0 or later

6. SYSTEM DESIGN

6.1 UML DIAGRAMS

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagrams, which is as follows.

- **User Model View**

1. This view represents the system from the user's perspective.
2. The analysis representation describes a usage scenario from the end-user's perspective.

- **Structural Model view**

1. In this model the data and functionality are arrived from inside the system.
2. This model view models the static structures.

- **Behavioural Model View**

It represents the dynamic of behavioural as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

- **Implementation Model View**

In this the structural and behavioural as parts of the system are represented as they are to be built.

- **Environmental Model View**

In these the structural and behavioural aspects of the environment in which the system is to be implemented are represented.

CLASS DIAGRAM

The class diagram is the main building block of object-oriented modelling. It is used both for general conceptual modelling of the systematic of the application, and for detailed modelling translating the models into programming code. Class diagrams can also be used for data modelling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. In the diagram, classes are represented with boxes which contain three parts:

- The upper part holds the name of the class.
- The middle part contains the attributes of the class.
- The bottom part gives the methods or operations the class can take or undertake.

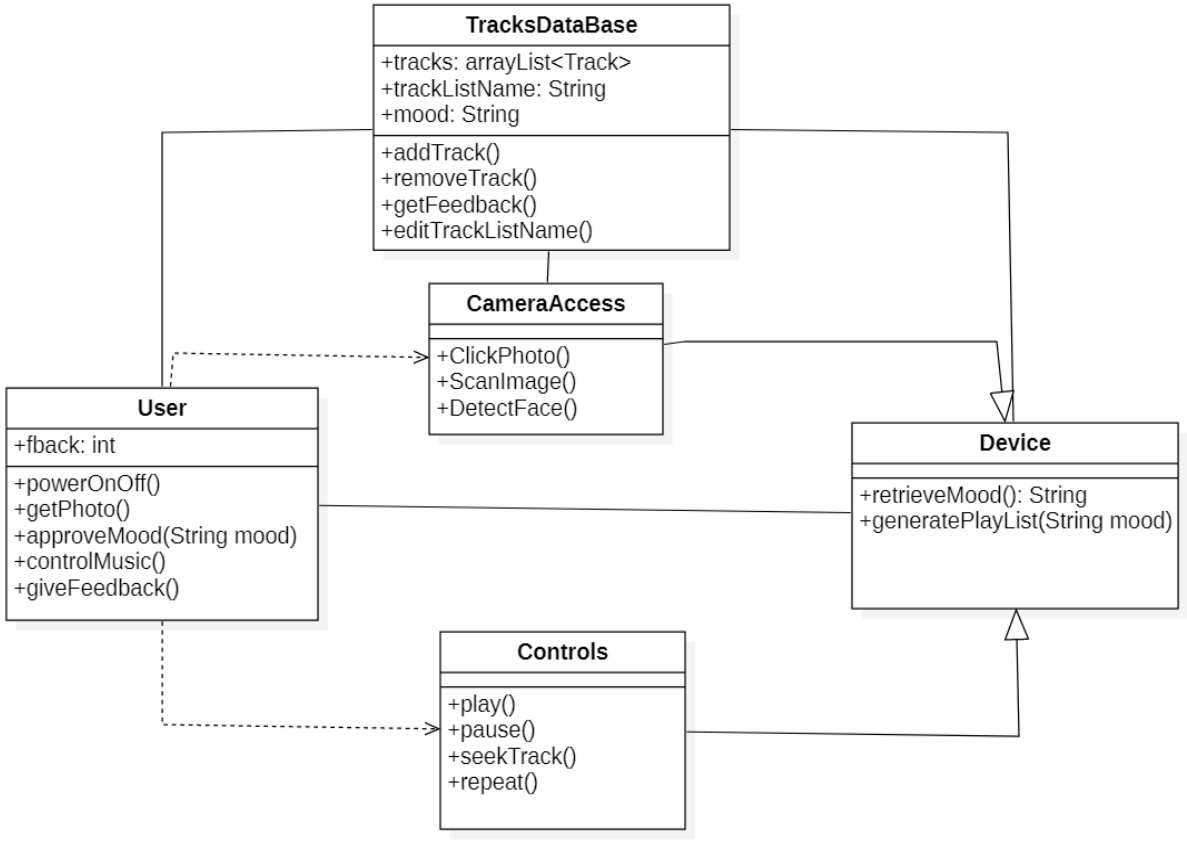

Figure 3: CLASS DIAGRAM

USE CASE DIAGRAM

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.

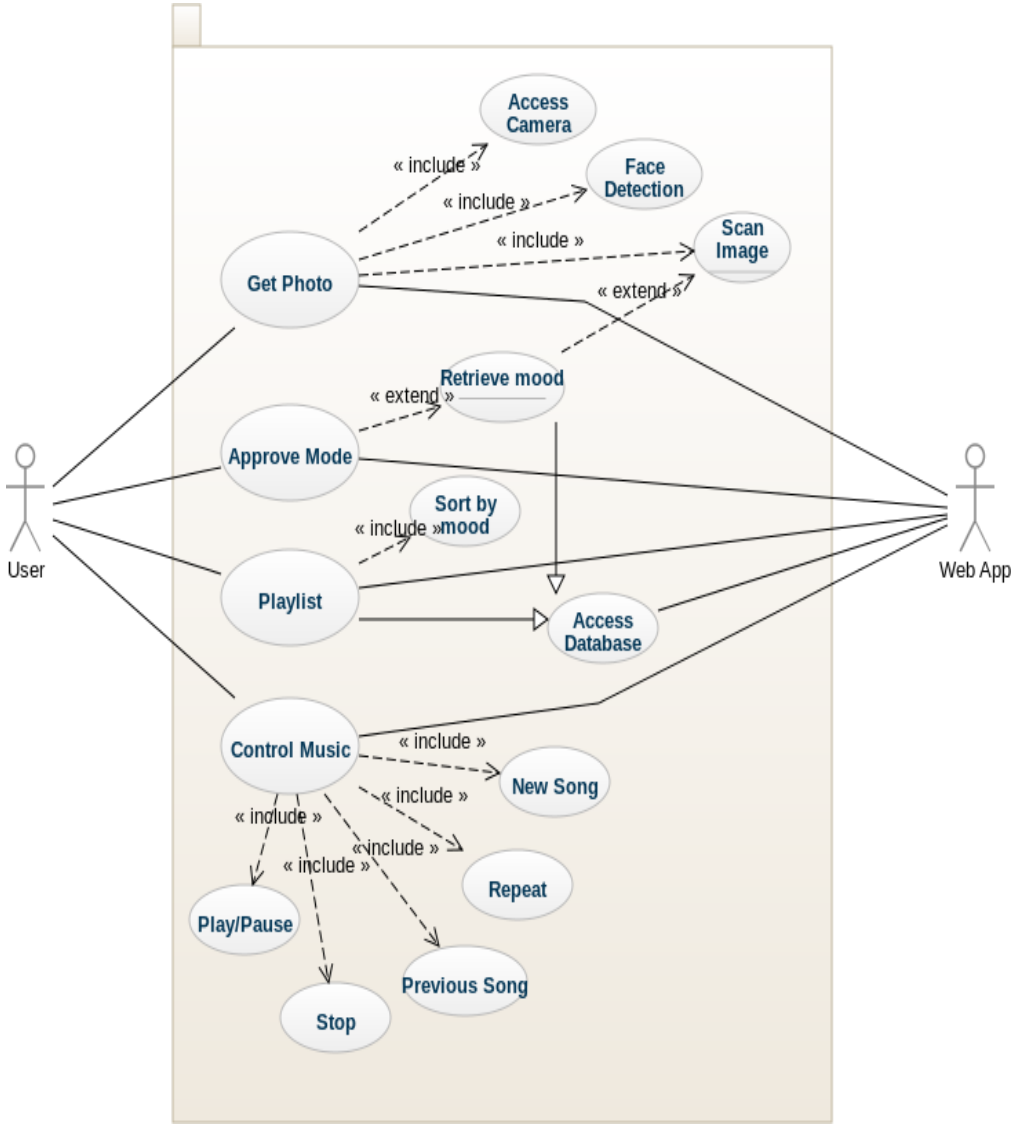

Figure 4 : USE CASE DIAGRAM

SEQUENCE DIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

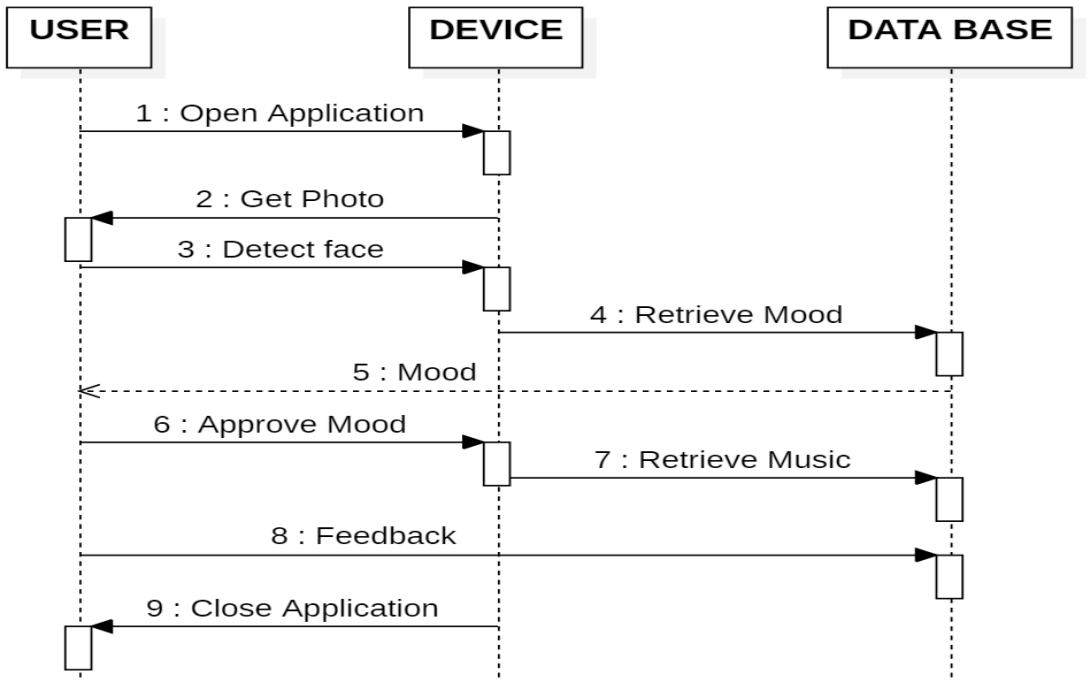

Figure 5: SEQUENCE DIAGRAM

DEPLOYMENT DIAGRAM

In the Unified Modeling Language, a component diagram depicts how components are wired together to form larger components and or software systems. They are used to illustrate the structure of arbitrarily complex systems.

Components are wired together by using an assembly connector to connect the required interface of one component with the provided interface of another component. This illustrates the service consumer - service provider relationship between the two components.

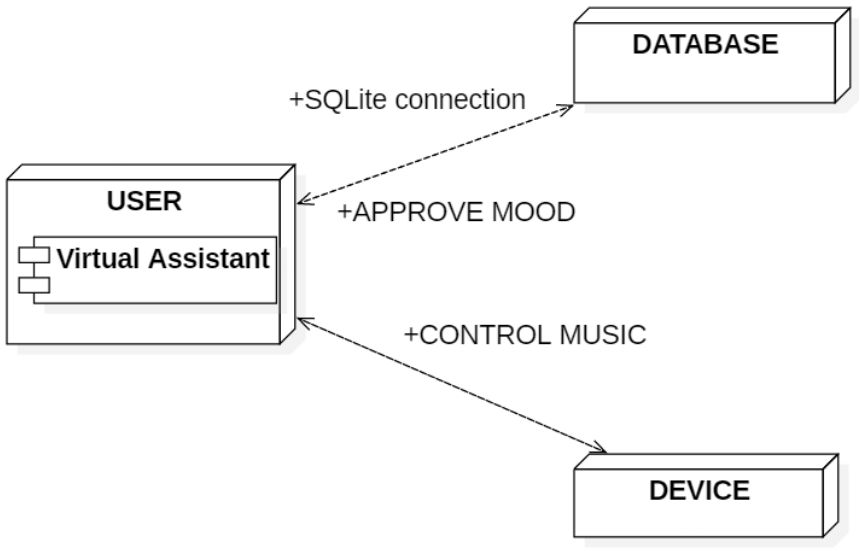

Figure 6: DEPLOYMENT DIAGRAM

ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe dynamic aspects of the system. It is basically a flow chart to represent the flow form one activity to another activity. The activity can be described as an operation of the system. So, the control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent.

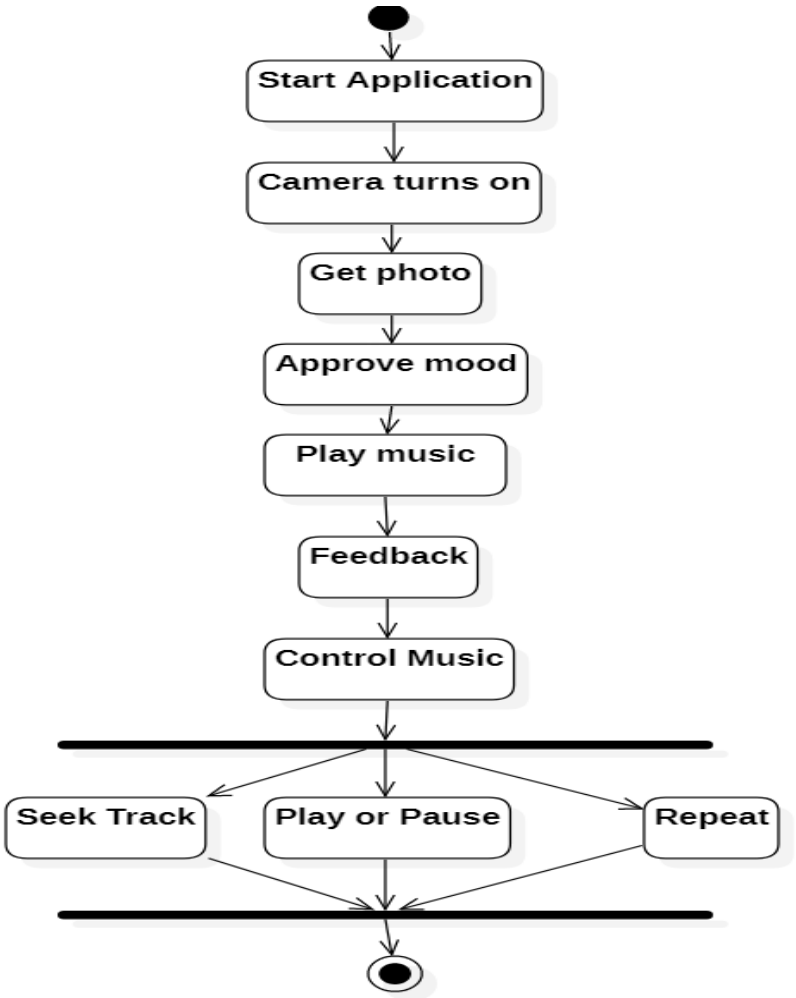

Figure 7: ACTIVITY DIAGRAM

STATECHART DIAGRAM

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents the behavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams. We prefer to model the states with three or more states.

- We use it to state the events responsible for change in state (we do not show what processes cause those events).
- We use it to model the dynamic behavior of the system .
- To understand the reaction of objects/classes to internal or external stimuli.

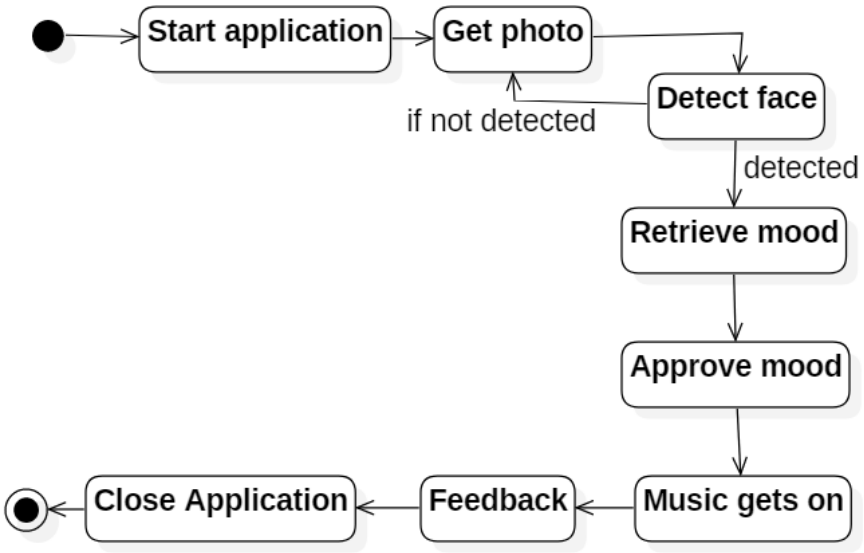

Figure 8: STATECHART DIAGRAM

6.2 DATA FLOW DIAGRAM

Data flow diagrams illustrate how data is processed by a system in terms of inputs and outputs. Data flow diagrams can be used to provide a clear representation of any business function. The technique starts with an overall picture of the business and continues by analysing each of the functional areas of interest. This analysis can be carried out in precisely the level of detail required. The technique exploits a method called top-down expansion to conduct the analysis in a targeted way.

As the name suggests, Data Flow Diagram (DFD) is an illustration that explicates the passage of information in a process. A DFD can be easily drawn using simple symbols. Additionally, complicated processes can be easily automated by creating DFDs using easy-to-use, free downloadable diagramming tools. A DFD is a model for constructing and analysing information processes. DFD illustrates the flow of information in a process depending upon the inputs and outputs. A DFD can also be referred to as a Process Model. A DFD monstrates business or technical process with the support of the outside data saved, plus the data flowing from the process to another and the end results.

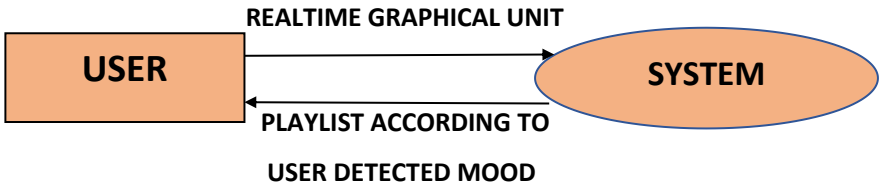

Figure 9: DFD LEVEL 0

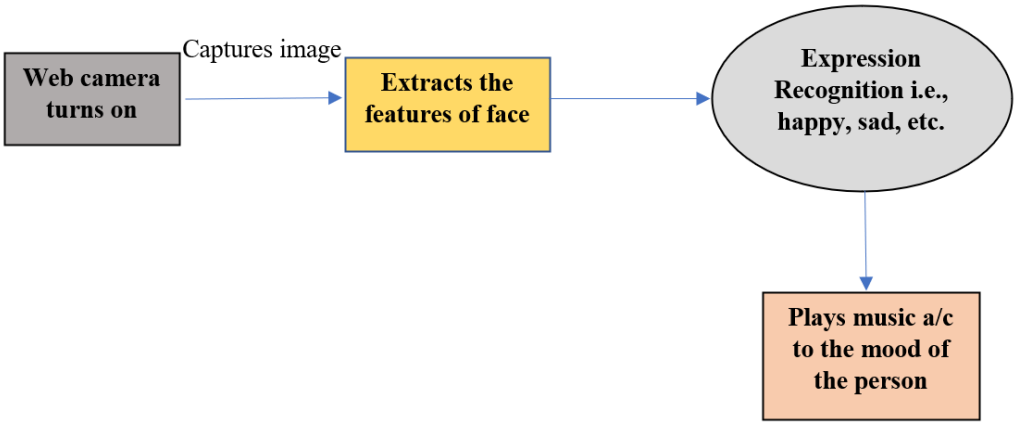

Figure 10: DFD LEVEL 1

7. SYSTEM IMPLEMENTATION

7.1 SYSTEM ARCHITECHTURE

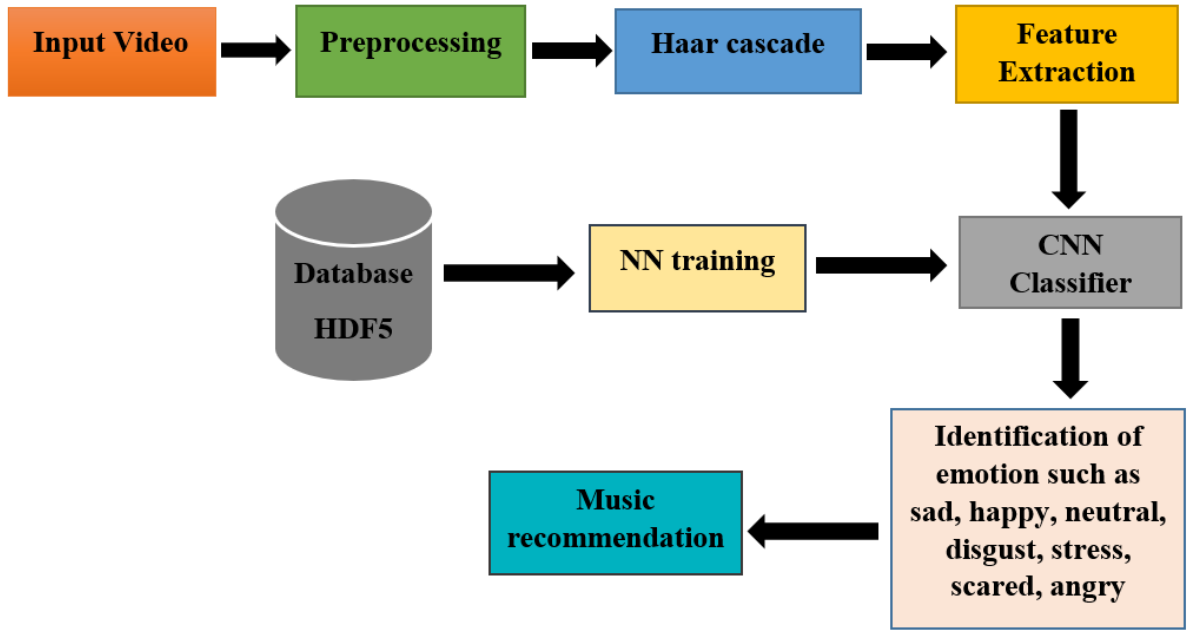

Figure 11: Architecture Diagram

Fig 11 shows the Environment of the system design and how the connectivity is done to each and everything in this system

7.2 TECHNOLOGIES / TOOLS REQUIRED

EXTERNAL INTERFACE REQUIREMENTS

USER INTERFACE

The user interface of this system is a user-friendly python Graphical User Interface.

HARDWARE INTERFACES

The interaction between the user and the console is achieved through python capabilities.

SOFTWARE INTERFACES

The required software is python.

OPERATING ENVIRONMENT

Windows 10.

OVERALL DESCRIPTION

A Software Requirements Specification (SRS) – a requirements specification for a software system is a complete description of the behavior of a system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. In addition to use cases, the SRS also contains non-functional requirements. Nonfunctional requirements are requirements which impose constraints on the design or implementation (such as performance engineering requirements, quality standards, or design constraints).

System requirements specification: A structured collection of information that embodies the requirements of a system. A business analyst, sometimes titled system analyst, is responsible for analyzing the business needs of their clients and stakeholders to help identify business problems and propose solutions. Within the systems development lifecycle domain, the BA typically performs a liaison function between the business side of an enterprise and the information technology department or external service providers.

PYTHON

Python is a general-purpose language. It has wide range of applications from Web development (like: Django and Bottle), scientific and mathematical computing (Orange, SymPy, NumPy) to desktop graphical user Interfaces (Pygame, Panda3D). The syntax of the language is clean and length of the code is relatively short. It's fun to work in Python because it allows you to think about the problem rather than focusing on the syntax.

HISTORY OF PYTHON:

Python is a fairly old language created by Guido Van Rossum. The design began in the late 1980s and was first released in February 1991.

WHY PYTHON WAS CREATED?

In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to design of a new language which was later named Python.

WHY THE NAME PYTHON?

No. It wasn't named after a dangerous snake. Rossum was fan of a comedy series from late seventies. The name "Python" was adopted from the same series "Monty Python's Flying Circus".

FEATURES OF PYTHON:

1. A simple language which is easier to learn

Python has a very simple and elegant syntax. It's much easier to read and write Python programs compared to other languages like: C++, Java, C#. Python makes programming fun and allows you to focus on the solution rather than syntax. If you are a newbie, it's a great choice to start your journey with Python.

2. Free and open-source

You can freely use and distribute Python, even for commercial use. Not only can you use and distribute software's written in it, you can even make changes to the Python's source code. Python has a large community constantly improving it in each iteration.

3. Portability

You can move Python programs from one platform to another, and run it without any changes. It runs seamlessly on almost all platforms including Windows, Mac OS X and Linux.

4. Extensible and Embeddable

Suppose an application requires high performance. You can easily combine pieces of C/C++ or other languages with Python code. This will give your application high performance as well as scripting capabilities which other languages may not provide out of the box.

5. A high-level, interpreted language

Unlike C/C++, you don't have to worry about daunting tasks like memory management, garbage collection and so on. Likewise, when you run Python code, it automatically converts your code to the language your computer understands. You don't need to worry about any lower-level operations.

6. Large standard libraries to solve common tasks

Python has a number of standard libraries which makes life of a programmer much easier since you don't have to write all the code yourself. For example: Need to connect MySQL database on a Web server? You can use MySQL dB library using import MySQL Db. Standard libraries in Python are well tested and used by hundreds of people. So you can be sure that it won't break your application.

7. Object-oriented

Everything in Python is an object. Object oriented programming (OOP) helps you solve a complex problem intuitively. With OOP, you are able to divide these complex problems into smaller sets by creating objects.

CNN

HAAR – CASCADES

Haar- like features are rectangular patterns in data. A cascade is a series of "Haar-like features" that are combined to form a classifier [14]. A Haar wavelet is a mathematical function that produces square wave output.

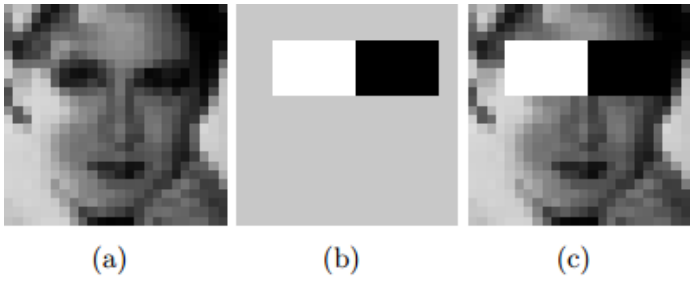

Figure 12 : Haar like Features

Figure 2.2 shows Haar like features, the background of a template like (b) is painted gray to highlight the pattern's support. Only those pixels marked in black or white are used when the corresponding feature is calculated [15].

Since no objective distribution can describe the actual prior probability for a given image to have a face, the algorithm must minimize both the false negative and false positive rates in order to achieve an acceptable performance [16]. This then requires an accurate numerical description of what sets human faces apart from other objects. Characteristics that define a face can be extracted from the images with a remarkable committee learning algorithm called Adaboost [17]. Adaboost (Adaptive boost) relies on a committee of weak classifiers that combine to form a strong one through a voting mechanism [18]. A classifier is weak if, in general, it cannot meet a predefined classification target in error terms [7]. The operational algorithm to be used must also work with a reasonable computational budget. Such techniques as the integral image and attention cascades have made the Viola-Jones algorithm [15] highly efficient: fed with a real time image sequence generated from a standard webcam or camera, it performs well on a standard PC.

The size and position of a pattern's support can vary provided its black and white rectangles have the same dimension, border each other and keep their relative positions. Thanks to this constraint, the number of features one can draw from an image is somewhat manageable: a 24 × 24 image, for instance, has 43200, 27600, 43200, 27600 and 20736 features of category (a),

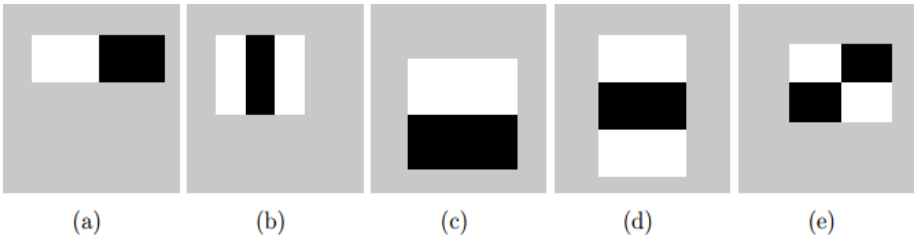

Figure 13 : Haar-like features with different sizes and orientation [13]

(b), (c), (d) and (e) respectively as shown in figure 2.3, hence 162336 features in all[13]. In practice, five patterns are considered. The derived features are assumed to hold all the information needed to characterize a face. Since faces are large and regular by nature, the use of Haar-like patterns.

1. How The HAAR – Like Features Work

A scale is chosen for the features say 24 × 24 pixels. This is then slid across the image. The average pixel values under the white area and the black area are then computed. If the difference between the areas is above some threshold, then the feature matches [7].

In face detection, since the eyes are of different color tone from the nose, the Haar feature (b) from Figure 2.3 can be scaled to fit that area as shown below,

Figure 14 : How the Haar like feature of figure 2.3 can be used to scale the eyes

One Haar feature is however not enough as there are several features that could match it (like the zip drive and white areas at the background of the image of figure 2.4 it is called a “weak classifier.” Haar cascades, the basis of Viola Jones detection framework

[16] therefore consist of a series of weak classifiers whose accuracy is at least 50% correct. If an area passes a single classifier, it moves to the next weak classifier and so on, otherwise, the area does not match.

Cascaded Classifier

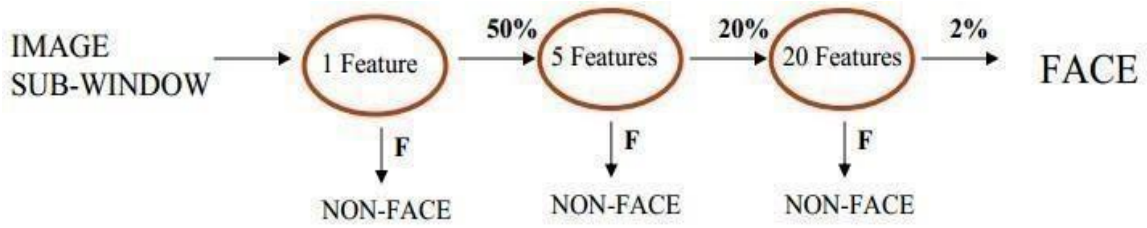

Figure 15 : Several classifiers combined to enhance face detection

From figure 2.5, a 1 feature classifier achieves 100% face detection rate and about 50% false positive rate. A 5 feature classifier achieves 100% detection rate and 40% false positive rate (20% cumulative). A 20 feature classifier achieves 100% detection rate with 10% false positive rate (2% cumulative)[17].Combining several weak classifiers improves the accuracy of detection.

A training algorithm called Adaboost, short for adaptive boosting [14], which had no application before Haar cascades [14], was utilized to combine a series of weak classifiers in to a strong classifier. Adaboost tries out multiple weak classifiers over several rounds, selecting the best weak classifier in each round and combining the best weak classifier to create a strong classifier [7]. Adaboost can use classifiers that are consistently wrong by reversing their decision [7]. In the design and development, it can take weeks of processing time to determine the final cascade sequence [18].

After the final cascade had been constructed, there was a need for a way to quickly compute the Haar features i.e. compute the differences in the two areas. The integral image was instrumental in this.

Integral Image

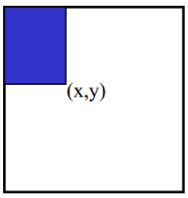

Figure 16 : Pixel Coordinates of an integral image

The Integral image also known as the “summed area table” developed in 1984 came in to widespread use in 2001 with the Haar cascades [4]. A summed area table is created in a single pass. This makes the Haar cascades fast, since the sum of any region in the image can be computed using a single formula. The integral image computes a value at each pixel (x, y) as is shown in figure 2.6, that is the sum of the pixel values above and to the left of (x, y), inclusive. This can quickly be computed in one pass through the image.

Let A, B, C D be the values of the integral image at the corners of a rectangle as shown in figure 2.7.

The sum of original image values within the rectangle can be computed.

$Su = A - B - C + D$ - (2.1)

Only three additions are required for any size of rectangle[17]. This face detection approach minimizes computation time while achieving high detection accuracy[15]. It is now used in many areas of computer vision [4] [7].

Minimum Neighbors Threshold

The minimum neighbor's threshold sets the cutoff level for discarding or keeping rectangle groups as either faces or not. This is based on the number of raw detections in the group and its values ranges from zero to four.

When the face detector is called behind the scenes, each positive face region generates many hits from the Haar detector as in Figure 2.8. The face region itself generates a large cluster of rectangles that to a large extend overlap. The isolated detections are usually false detections and are discarded. The multiple face region detections are then merged in to a single detection. The face detection function does all this before returning the list of the detected faces. The merge step groups rectangles that contain a large number of overlaps and then finds the average.

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Why ConvNets over Feed-Forward Neural Nets?

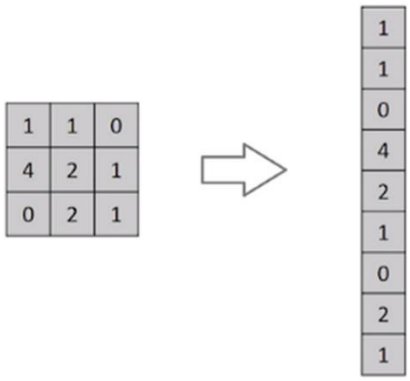

Figure 17 : Flattening of a 3x3 image matrix into a 9x1 vector

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

A ConvNet is able to successfully capture the Spatial and Temporal dependencies in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

Convolution Layer — The Kernel

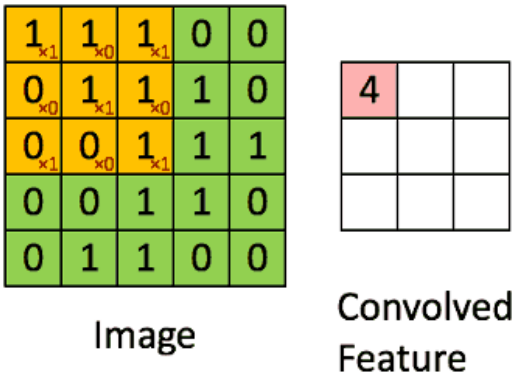

Figure 18 : The Kernel

Convoluting a 5x5x1 image with a 3x3x1 kernel to get a 3x3x1 convolved feature

Image Dimensions = 5 (Height) x 5 (Breadth) x 1 (Number of channels, eg. RGB)

In the above demonstration, the green section resembles our 5x5x1 input image, I. The element involved in carrying out the convolution operation in the first part of a Convolutional Layer is called the Kernel/Filter, K, represented in the color yellow. We have selected K as a 3x3x1 matrix.

Kernel/Filter,

K = 1 0 1

0 1 0

1 0 1

The Kernel shifts 9 times because of Stride Length = 1 (Non-Strided), every time performing a matrix multiplication operation between K and the portion P of the image over which the kernel is hovering.

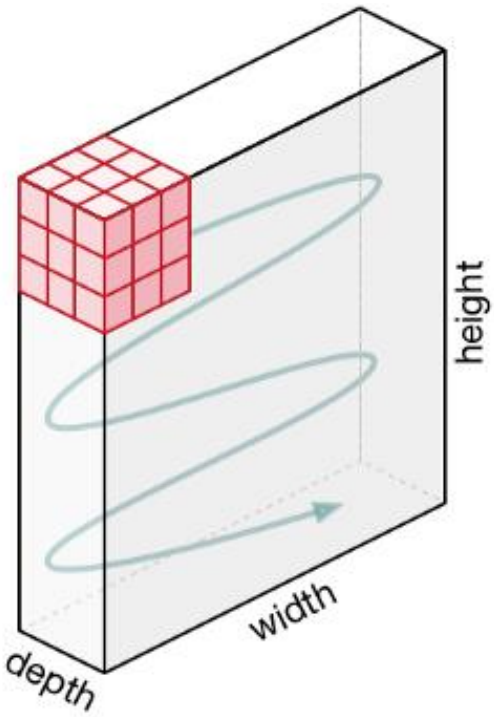

Figure 19: Movement of the Kernel

Movement of the Kernel

The filter moves to the right with a certain Stride Value till it parses the complete width. Moving on, it hops down to the beginning (left) of the image with the same Stride Value and repeats the process until the entire image is traversed.

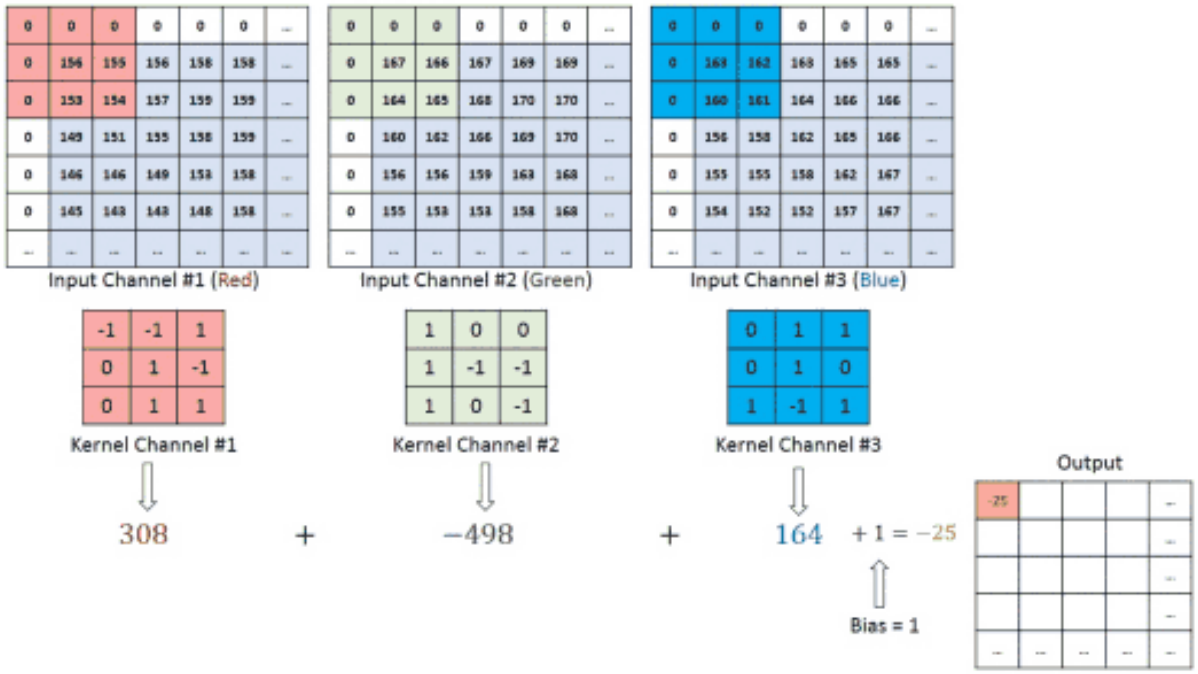

Figure 20 : Convolution operation on a MxNx3 image matrix with a 3x3x3 Kernel

In the case of images with multiple channels (e.g. RGB), the Kernel has the same depth as that of the input image. Matrix Multiplication is performed between Kn and In stack ([K1, I1]; [K2, I2]; [K3, I3]) and all the results are summed with the bias to give us a squashed one-depth channel Convoluted Feature Output.

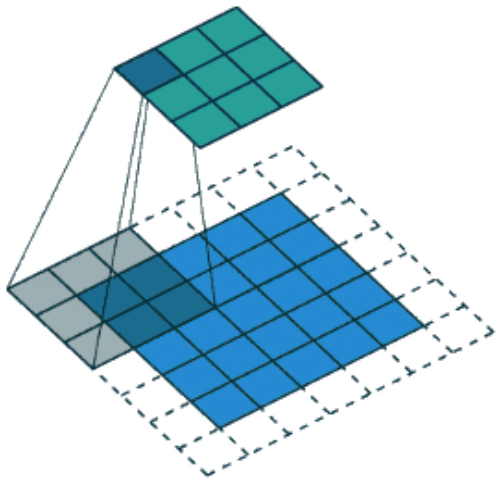

Figure 21 : Convolution Operation with Stride Length = 2

The objective of the Convolution Operation is to **extract the high-level features** such as edges, from the input image. ConvNets need not be limited to only one Convolutional Layer. Conventionally, the first ConvLayer is responsible for capturing the Low-Level features such as edges, color, gradient orientation, etc. With added layers, the architecture adapts to the High-Level features as well, giving us a network which has the wholesome understanding of images in the dataset, similar to how we would.

There are two types of results to the operation — one in which the convolved feature is reduced in dimensionality as compared to the input, and the other in which the dimensionality is either increased or remains the same. This is done by applying **Valid Padding** in case of the former, or **Same Padding** in the case of the latter.

Pooling Layer

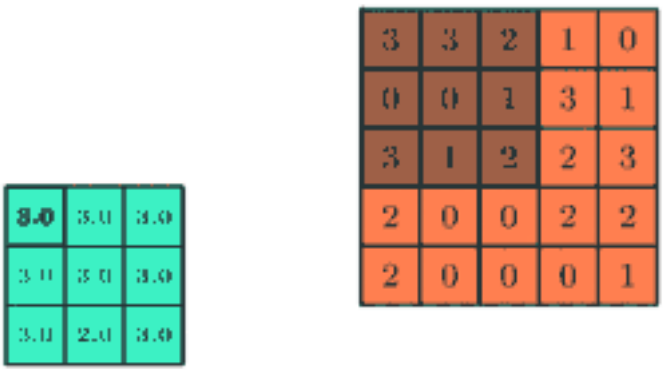

Figure 22 : 3x3 pooling over 5x5 convolved feature

Similar to the Convolutional Layer, the Pooling layer is responsible for reducing the spatial size of the Convolved Feature. This is to decrease the computational power required to process the data through dimensionality reduction. Furthermore, it is useful for extracting dominant features which are rotational and positional invariant, thus maintaining the process of effectively training of the model.

There are two types of Pooling: Max Pooling and Average Pooling. Max Pooling returns the maximum value from the portion of the image covered by the Kernel. On the other hand, Average Pooling returns the average of all the values from the portion of the image covered by the Kernel.

Max Pooling also performs as a Noise Suppressant. It discards the noisy activations altogether and also performs de-noising along with dimensionality reduction. On the other hand, Average Pooling simply performs dimensionality reduction as a noise suppressing mechanism. Hence, we can say that Max Pooling performs a lot better than Average Pooling.

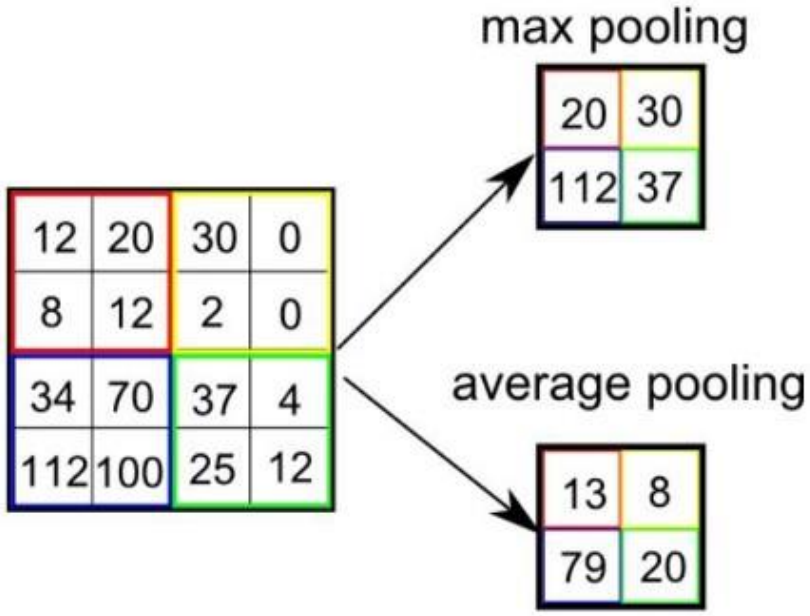

Figure 23 : Pooling Layer

Types of Pooling

The Convolutional Layer and the Pooling Layer, together form the i-th layer of a Convolutional Neural Network. Depending on the complexities in the images, the number of such layers may be increased for capturing low-levels details even further, but at the cost of more computational power.

After going through the above process, we have successfully enabled the model to understand the features. Moving on, we are going to flatten the final output and feed it to a regular Neural Network for classification purposes.

Adding a Fully-Connected layer is a (usually) cheap way of learning non-linear combinations of the high-level features as represented by the output of the convolutional layer. The Fully-Connected layer is learning a possibly non-linear function in that space.

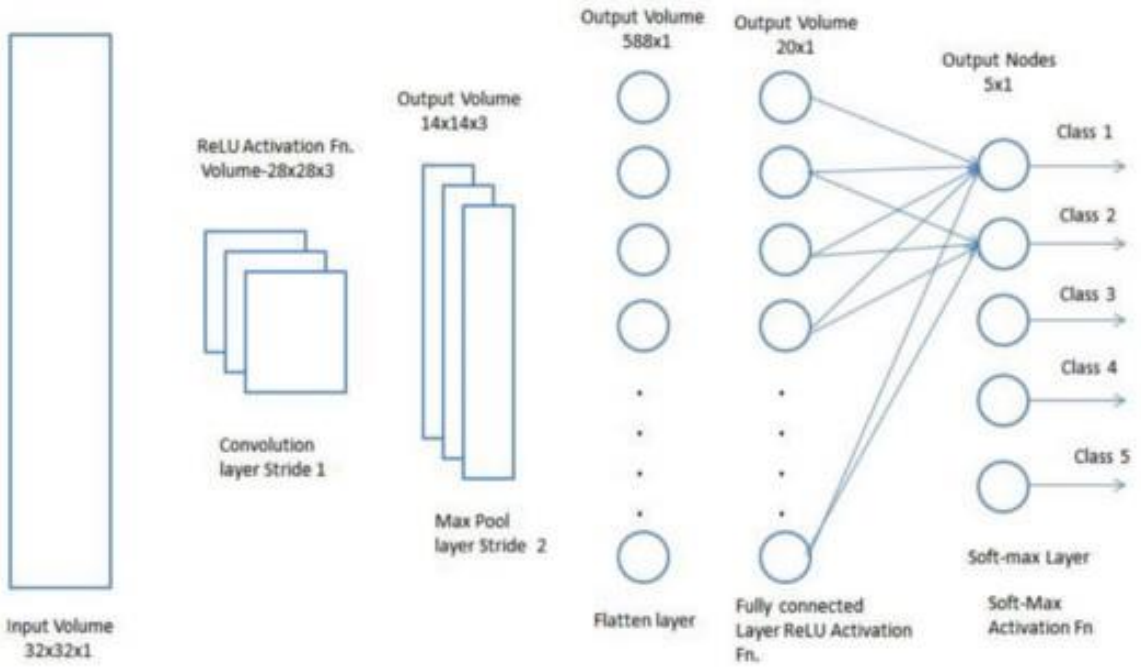

Figure 24 : Classification — Fully Connected Layer (FC Layer)

Now that we have converted our input image into a suitable form for our Multi-Level Perceptron, we shall flatten the image into a column vector. The flattened output is fed to a feed-forward neural network and backpropagation applied to every iteration of training. Over a series of epochs, the model is able to distinguish between dominating and certain low-level features in images and classify them using the **Softmax Classification** technique.

Neural Network:

Neural networks are predictive models loosely based on the action of biological neurons.

The selection of the name "neural network" was one of the great PR successes of the Twentieth Century. It certainly sounds more exciting than a technical description such as "A network of weighted, additive values with nonlinear transfer functions". However, despite the name, neural networks are far from "thinking machines" or "artificial brains". A typical artificial neural network might have a hundred neurons. In comparison, the human nervous system is believed to have about 3x1010 neurons. We are still light years from "Data".

The original "Perceptron" model was developed by Frank Rosenblatt in 1958. Rosenblatt's model consisted of three layers, (1) a "retina" that distributed inputs to the second layer, (2) "association units" that combine the inputs with weights and trigger a threshold step function which feeds to the output layer, (3) the output layer which combines the values.

Unfortunately, the use of a step function in the neurons made the perceptions difficult or impossible to train. A critical analysis of perceptrons published in 1969 by Marvin Minsky and Seymore Paper pointed out a number of critical weaknesses of perceptrons, and, for a period of time, interest in perceptrons waned.

Interest in neural networks was revived in 1986 when David Rumelhart, Geoffrey Hinton and Ronald Williams published "Learning Internal Representations by Error Propagation". They proposed a multilayer neural network with nonlinear but differentiable transfer functions that avoided the pitfalls of the original perceptron's step functions. They also provided a reasonably effective training algorithm for neural networks.

Types of Neural Networks:

1. Artificial Neural Network
2. Probabilistic Neural Networks
3. General Regression Neural Networks

DTREG implements the most widely used types of neural networks:

a. Multilayer Perceptron Networks (also known as multilayer feed-forward network),
b. Cascade Correlation Neural Networks,
c. Probabilistic Neural Networks (NN)
d. General Regression Neural Networks (GRNN).

The Multilayer Perceptron Neural Network Model

The following diagram illustrates a perceptron network with three layers:

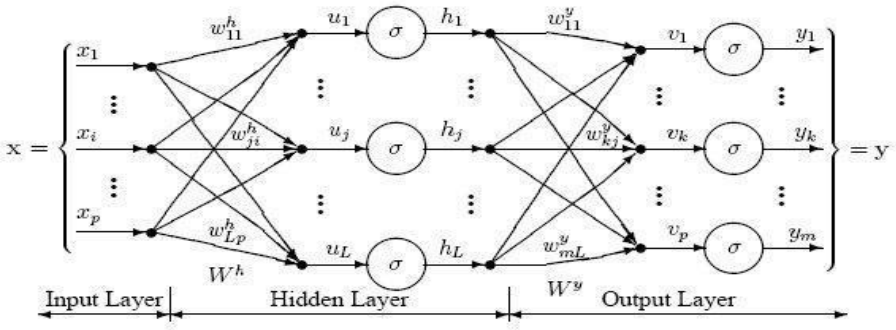

Figure 25 : The Multilayer Perceptron Neural Network Model

This network has an input layer (on the left) with three neurons, one hidden layer (in the middle) with three neurons and an output layer (on the right) with three neurons.

There is one neuron in the input layer for each predictor variable. In the case of categorical variables, N-1 neurons are used to represent the N categories of the variable.

Input Layer — A vector of predictor variable values (x1...xp) is presented to the input layer. The input layer (or processing before the input layer) standardizes these values so that the range of each variable is -1 to 1. The input layer distributes the values to each of the neurons in the hidden layer. In addition to the predictor variables, there is a constant input of 1.0, called the bias that is fed to each of the hidden layers; the bias is multiplied by a weight and added to the sum going into the neuron.

Hidden Layer — Arriving at a neuron in the hidden layer, the value from each input neuron is multiplied by a weight (wji), and the resulting weighted values are added together producing a combined value uj. The weighted sum (uj) is fed into a transfer function, σ, which outputs a value hj. The outputs from the hidden layer are distributed to the output layer.

Output Layer — Arriving at a neuron in the output layer, the value from each hidden layer neuron is multiplied by a weight (wkj), and the resulting weighted values are added together producing a combined value vj. The weighted sum (vj) is fed into a transfer function, σ, which outputs a value yk. The y values are the outputs of the network.

If a regression analysis is being performed with a continuous target variable, then there is a single neuron in the output layer, and it generates a single y value. For classification problems with categorical target variables, there are N neurons in the output layer producing N values, one for each of the N categories of the target variable.

Neural Networks (NN):

Neural Network (NN) and General Regression Neural Networks (GRNN) have similar architectures, but there is a fundamental difference: networks perform classification where the target variable is categorical, whereas general regression neural networks perform regression where the target variable is continuous. If you select a NN/GRNN network, DTREG will automatically select the correct type of network based on the type of target variable.

Architecture of a NN:

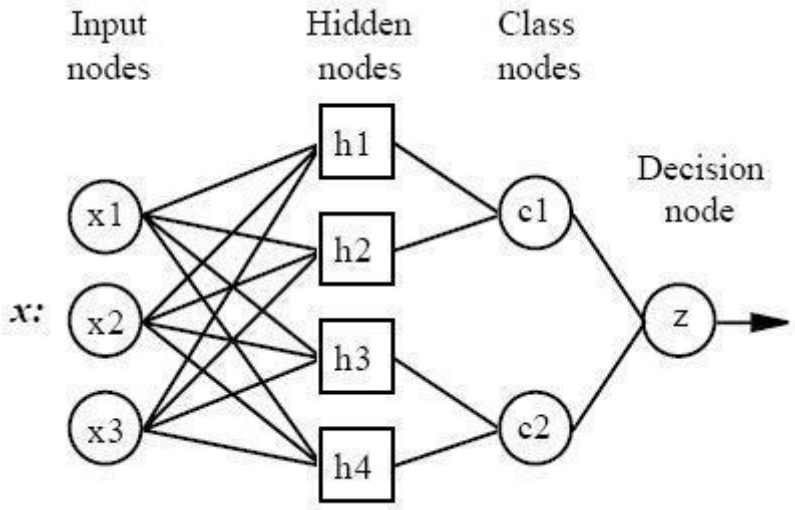

Figure 26 : Architecture of a NN

All NN networks have four layers:

1. **Input layer** — There is one neuron in the input layer for each predictor variable. In the case of categorical variables, N-1 neurons are used where N is the number of categories. The input neurons (or processing before the input layer) standardizes the range of the values by subtracting the median and dividing by the interquartile range. The input neurons then feed the values to each of the neurons in the hidden layer.
2. **Hidden layer** — This layer has one neuron for each case in the training data set. The neuron stores the values of the predictor variables for the case along with the target value. When presented with the x vector of input values from the input layer, a hidden neuron computes the Euclidean distance of the test case from the neuron's center point and then applies the RBF kernel function using the sigma value(s). The resulting value is passed to the neurons in the pattern layer.
3. **Pattern layer / Summation layer** — The next layer in the network is different for NN networks and for GRNN networks. For NN networks there is one pattern neuron for each category of the target variable. The actual target category of each training case is stored with each hidden neuron; the weighted value coming out of a hidden neuron is fed only to the pattern neuron that corresponds to the hidden neuron's category. The pattern neurons add the values for the class they represent (hence, it is a weighted vote for that category).

For GRNN networks, there are only two neurons in the pattern layer. One neuron is the denominator summation unit the other is the numerator summation unit. The denominator summation unit adds up the weight values coming from each of the hidden neurons. The numerator summation unit adds up the weight values multiplied by the actual target value for each hidden neuron.

4. **Decision layer** — The decision layer is different for NN and GRNN networks. For NN networks, the decision layer compares the weighted votes for each target category accumulated in the pattern layer and uses the largest vote to predict the target category.

For GRNN networks, the decision layer divides the value accumulated in the numerator summation unit by the value in the denominator summation unit and uses the result as the predicted target value.

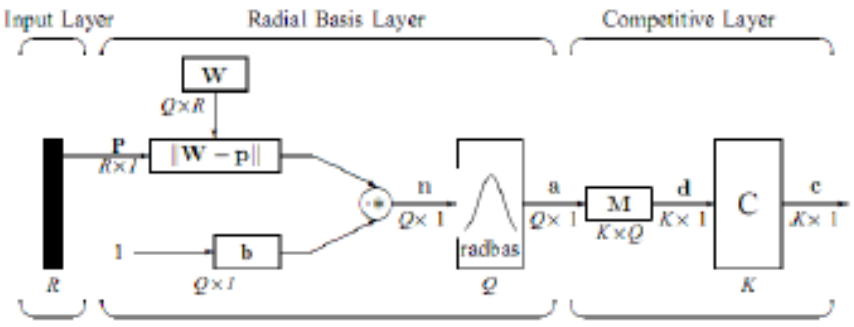

Figure 27 : Layers of NN

The following diagram is actual diagram or propose network used in our project

1) Input Layer:

The input vector, denoted as p, is presented as the black vertical bar.Its dimension is R × 1. In this paper, R = 3.

2) Radial Basis Layer:

In Radial Basis Layer, the vector distances between input vector p and the weight vector made of each row of weight matrix W are calculated. Here, the vector distance is defined as the dot product between two vectors [8]. Assume the dimension of W is Q×R. The dot product between p and the i-th row of W produces the i-th element of the distance vector ||W-p||, whose dimension is Q×1. The minus symbol, "-", indicates that it is the distance between vectors.

Then, the bias vector b is combined with ||W- p|| by an element-by-element multiplication, .The result is denoted as n = ||W- p|| ..p. The transfer function in NN has built into a distance criterion with respect to a center. In this paper, it is defined as radbas(n) = 2 n e- (1) Each element of n is substituted into Eq. 1 and produces corresponding element of a, the output vector of Radial Basis Layer. The i-th element of a can be represented as ai = radbas(||Wi - p|| ..bi) (2) where Wi is the vector made of the i-th row of W and bi is the i-th element of bias vector b.

Some characteristics of Radial Basis Layer:

The i-th element of a equals to 1 if the input p is identical to the ith row of input weight matrix W. A radial basis neuron with a weight vector close to the input vector p produces a value near 1 and then its output weights in the competitive layer will pass their values to the competitive function. It is also possible that several elements of a are close to 1 since the input pattern is close to several training patterns. .

3) Competitive Layer:

There is no bias in Competitive Layer. In Competitive Layer, the vector a is firstly multiplied with layer weight matrix M, producing an output vector d. The competitive function, denoted as C in Fig. 2, produces a 1 corresponding to the largest element of d, and 0's elsewhere. The output vector of competitive function is denoted as c. The index of 1 in c is the number of tumor that the system can classify. The dimension of output vector, K, is 5 in this paper.

How NN network work:

Although the implementation is very different, neural networks are conceptually similar to K-Nearest Neighbor (k-NN) models. The basic idea is that a predicted target value of an item is likely to be about the same as other items that have close values of the predictor variables. Consider this figure:

Assume that each case in the training set has two predictor variables, x and y. The cases are plotted using their x,y coordinates as shown in the figure. Also assume that the target variable has two categories, positive which is denoted by a square and negative which is denoted by a dash. Now, suppose we are trying to predict the value of a new case represented by the triangle with predictor values x=6, y=5.1. Should we predict the target as positive or negative?

Notice that the triangle is position almost exactly on top of a dash representing a negative value. But that dash is in a fairly unusual position compared to the other dashes which are clustered below the squares and left of center. So it could be that the underlying negative value is an odd case.

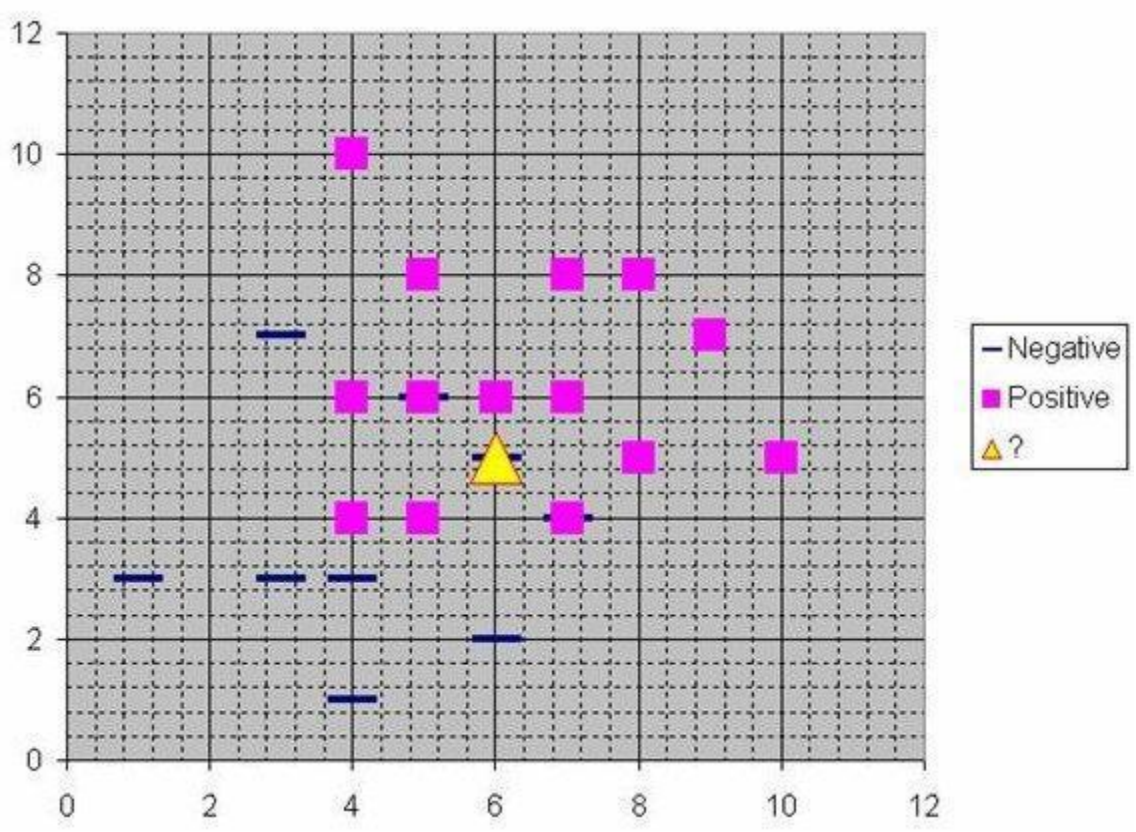

Figure 28 : NN NETWORK

The nearest neighbor classification performed for this example depends on how many neighboring points are considered. If 1-NN is used and only the closest point is considered, then clearly the new point should be classified as negative since it is on top of a known negative point. On the other hand, if 9-NN classification is used and the closest 9 points are considered, then the effect of the surrounding 8 positive points may overbalance the close negative point.

A neural network builds on this foundation and generalizes it to consider all of the other points. The distance is computed from the point being evaluated to each of the other points, and a radial basis function (RBF) (also called a kernel function) is applied to the distance to compute the weight (influence) for each point. The radial basis function is so named because the radius distance is the argument to the function.

Weight = RBF (distance)

The further some other point is from the new point, the less influence it has.

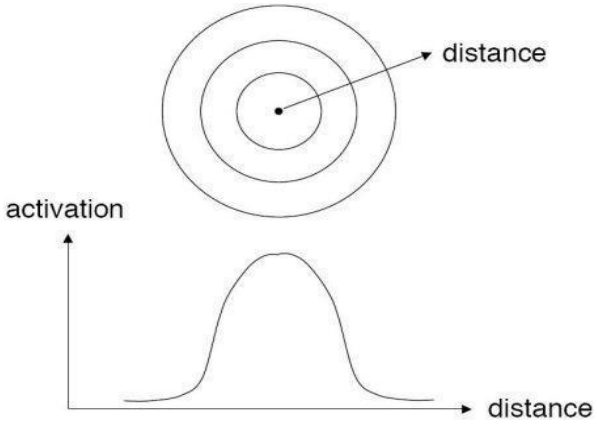

Figure 29 : Kernel function

Radial Basis Function

Different types of radial basis functions could be used, but the most common is the Gaussian function:

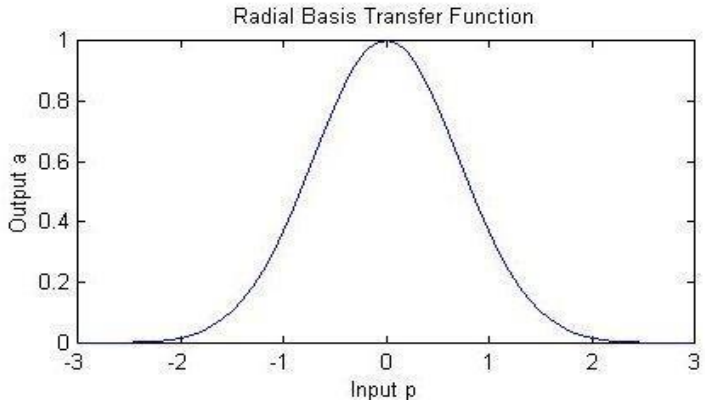

Figure 30 : Radial Basis Function

Advantages and disadvantages of NN networks:

1) It is usually much faster to train a NN/GRNN network than a multilayer perceptron network.

2) NN/GRNN networks often are more accurate than multilayer perceptron networks.

3) NN/GRNN networks are relatively insensitive to outliers (wild points).

4) NN networks generate accurate predicted target probability scores.

5)NN networks approach Bayes optimal classification.

6) NN/GRNN networks are slower than multilayer perceptron networks at classifying new cases.

7)NN/GRNN networks require more memory space to store the model.

Removing unnecessary neurons

One of the disadvantages of NN models compared to multilayer perceptron networks is that NN models are large due to the fact that there is one neuron for each training row. This causes the model to run slower than multilayer perceptron networks when using scoring to predict values for new rows.

DTREG provides an option to cause it remove unnecessary neurons from the model after the model has been constructed.

Removing unnecessary neurons has three benefits:

1. The size of the stored model is reduced.

2. The time required to apply the model during scoring is reduced.

3. Removing neurons often improves the accuracy of the model.

The process of removing unnecessary neurons is an iterative process. Leave-one-out validation is used to measure the error of the model with each neuron removed. The neuron that causes the least increase in error (or possibly the largest reduction in error) is then removed from the model. The process is repeated with the remaining neurons until the stopping criterion is reached.

When unnecessary neurons are removed, the “Model Size” section of the analysis report shows how the error changes with different numbers of neurons. You can see a graphical chart of this by clicking Chart/Model size.

7.3 SAMPLE CODE

```
from tensorflow.keras.preprocessing.image import img_to_array
import imutils
import cv2
from keras.models import load_model
import numpy as np
import time
import argparse
from playsound import playsound
parser = argparse.ArgumentParser()
parser.add_argument('-f', '--file',
    help='Path to video file (if not using camera)')
parser.add_argument('-c', '--color', type=str, default='gray',
    help='Color space: "gray" (default), "rgb", or "lab"')
parser.add_argument('-b', '--bins', type=int, default=16,
    help='Number of bins per channel (default 16)')
parser.add_argument('-w', '--width', type=int, default=0,
    help='Resize video to specified width in pixels (maintains aspect)')
args = vars(parser.parse_args())


# 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 = ["stress" ,"disgust","scared", "happy", "sad", "surprised",

 "neutral"]

# starting video streaming

cv2.namedWindow('my_face')

camera = cv2.VideoCapture(0)

time.sleep(2)

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=(30,3
0),flags=cv2.CASCADE_SCALE_IMAGE)

    canvas = np.zeros((600, 700, 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)

    else:continue
    preds = emotion_classifier.predict(roi)[0]
    emotion_probability = np.max(preds)
    label = EMOTIONS[preds.argmax()]

    for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
            # construct the label text
            text = "{}: {:.2f}%".format(emotion, prob * 100)

            w = int(prob * 300)
            cv2.rectangle(canvas, (7, (i * 35) + 5),
            (w, (i * 35) + 35), (0, 255, 0), -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, 255, 0), 2)
            cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH),
                    (0, 255, 0), 2)
            if label=="happy":
               print("you are in happy")
               cv2.putText(frameClone, "please listen music", (fX, fY - 30),
                        cv2.FONT_HERSHEY_SIMPLEX, 0.45, (0,0,255), 2)
               playsound('./nicee.mp3')

    cv2.imshow('emotion detection output', frameClone)
    cv2.imshow("Probabilities", canvas)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

camera.release()
cv2.destroyAllWindows()
```

8. SYSTEM TESTING

8.1 Testing:

The various levels of testing are

1. White Box Testing
2. Black Box Testing
3. Unit Testing
4. Functional Testing
5. Performance Testing
6. Integration Testing
7. Objective
8. Integration Testing
9. Validation Testing
10. System Testing
11. Structure Testing
12. Output Testing
13. User Acceptance Testing

White Box Testing

White-box testing (also known as clear box testing, glass box testing, transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application, as opposed to its functionality (i.e. black-box testing). In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT).

While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

White-box test design techniques include:

- Control flow testing
- Data flow testing
- Branch testing
- Path testing
- Statement coverage
- Decision coverage

White-box testing is a method of testing the application at the level of the source code. The test cases are derived through the use of the design techniques mentioned above: control flow testing, data flow testing, branch testing, path testing, statement coverage and decision coverage as well as modified condition/decision coverage. White-box testing is the use of these techniques as guidelines to create an error free environment by examining any fragile code.

These White-box testing techniques are the building blocks of white-box testing, whose essence is the careful testing of the application at the source code level to prevent any hidden errors later on. These different techniques exercise every visible path of the source code to minimize errors and create an error-free environment. The whole point of white-box testing is the ability to know which line of the code is being executed and being able to identify what the correct output should be.

Levels

1. Unit testing. White-box testing is done during unit testing to ensure that the code is working as intended, before any integration happens with previously tested code. White-box testing during unit testing catches any defects early on and aids in any defects that happen later on after the code is integrated with the rest of the application and therefore prevents any type of errors later on.

2. Integration testing. White-box testing at this level are written to test the interactions of each interface with each other. The Unit level testing made sure that each code was tested and working accordingly in an isolated environment and integration examines the correctness of the behaviour in an open environment through the use of white-box testing for any interactions of interfaces that are known to the programmer.

3. Regression testing. White-box testing during regression testing is the use of recycled white-box test cases at the unit and integration testing levels.

White-box testing's basic procedures involve the understanding of the source code that you are testing at a deep level to be able to test them. The programmer must have a deep understanding of the application to know what kinds of test cases to create so that every visible path is exercised for testing. Once the source code is understood then the source code can be analysed for test cases to be created. These are the three basic steps that white-box testing takes in order to create test cases:

1. Input, involves different types of requirements, functional specifications, detailed designing of documents, proper source code, security specifications. This is the preparation stage of white-box testing to layout all of the basic information.

2. Processing Unit, involves performing risk analysis to guide whole testing process, proper test plan, execute test cases and communicate results. This is the phase of building test cases to make sure they thoroughly test the application the given results are recorded accordingly.

3. Output, prepare final report that encompasses all of the above preparations and results.

Black Box Testing

Black-box testing is a method of software testing that examines the functionality of an application (e.g. what the software does) without peering into its internal structures or workings (see white-box testing). This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance. It typically comprises most if not all higher level testing, but can also dominate unit testing as well

Test procedures

Specific knowledge of the application's code/internal structure and programming knowledge in general is not required. The tester is aware of what the software is supposed to do but is not aware of how it does it. For instance, the tester is aware that a particular input returns a certain, invariable output but is not aware of how the software produces the output in the first place.

Test cases

Test cases are built around specifications and requirements, i.e., what the application is supposed to do. Test cases are generally derived from external descriptions of the software, including specifications, requirements and design parameters. Although the tests used are primarily functional in nature, non-functional tests may also be used. The test designer selects both valid and invalid inputs and determines the correct output without any knowledge of the test object's internal structure.

Test design techniques

Typical black-box test design techniques include:

- Decision table testing
- All-pairs testing
- State transition tables
- Equivalence partitioning
- Boundary value analysis

Unit testing

In computer programming, unit testing is a method by which individual units of source code, sets of one or more computer program modules together with associated control data, usage procedures, and operating procedures are tested to determine if they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. In procedural programming, a unit could be an entire module, but is more commonly an individual function or procedure. In object-oriented programming, a unit is often an entire interface, such as a class, but could be an individual method. Unit tests are created by programmers or occasionally by white box testers during the development process.

Ideally, each test case is independent from the others. Substitutes such as method stubs, mock objects, fakes, and test harnesses can be used to assist testing a module in isolation. Unit tests are typically written and run by software developers to ensure that code meets its design and behaves as intended. Its implementation can vary from being very manual (pencil and paper)to being formalized as part of build automation.

Testing will not catch every error in the program, since it cannot evaluate every execution path in any but the most trivial programs. The same is true for unit testing. Additionally, unit testing by definition only tests the functionality of the units themselves. Therefore, it will not catch integration errors or broader system-level errors (such as functions performed across multiple units, or non-functional test areas such as performance).

Unit testing should be done in conjunction with other software testing activities, as they can only show the presence or absence of particular errors; they cannot prove a complete absence of errors. In order to guarantee correct behaviour for every execution path and every possible input, and ensure the absence of errors, other techniques are required, namely the application of formal methods to proving that a software component has no unexpected behaviour.

Software testing is a combinatorial problem. For example, every Boolean decision statement requires at least two tests: one with an outcome of "true" and one with an outcome of "false". As a result, for every line of code written, programmers often need 3 to 5 lines of test code.

This obviously takes time and its investment may not be worth the effort. There are also many problems that cannot easily be tested at all – for example those that are nondeterministic or involve multiple threads. In addition, code for a unit test is likely to be at least as buggy as the code it is testing. Fred Brooks in The Mythical Man-Month quotes: never take two chronometers to sea. Always take one or three. Meaning, if two chronometers contradict, how do you know which one is correct?

Another challenge related to writing the unit tests is the difficulty of setting up realistic and useful tests. It is necessary to create relevant initial conditions so the part of the application being tested behaves like part of the complete system. If these initial conditions are not set correctly, the test will not be exercising the code in a realistic context, which diminishes the value and accuracy of unit test results.

To obtain the intended benefits from unit testing, rigorous discipline is needed throughout the software development process. It is essential to keep careful records not only of the tests that have been performed, but also of all changes that have been made to the source code of this or any other unit in the software. Use of a version control system is essential. If a later version of the unit fails a particular test that it had previously passed, the version-control software can provide a list of the source code changes (if any) that have been applied to the unit since that time.

It is also essential to implement a sustainable process for ensuring that test case failures are reviewed daily and addressed immediately if such a process is not implemented and ingrained into the team's workflow, the application will evolve out of sync with the unit test suite, increasing false positives and reducing the effectiveness of the test suite.

Unit testing embedded system software presents a unique challenge: Since the software is being developed on a different platform than the one it will eventually run on, you cannot readily run a test program in the actual deployment environment, as is possible with desktop programs.[7]

Functional testing

Functional testing is a quality assurance (QA) process and a type of black box testing that bases its test cases on the specifications of the software component under test. Functions are tested by feeding them input and examining the output, and internal program structure is rarely considered (not like in white-box testing). Functional Testing usually describes what the system does.

Functional testing differs from system testing in that functional testing "verifies a program by checking it against ... design document(s) or specification(s)", while system testing "validate a program by checking it against the published user or system requirements" (Kane, Falk, Nguyen 1999, p. 52).

Functional testing typically involves five steps. The identification of functions that the software is expected to perform

1. The creation of input data based on the function's specifications
2. The determination of output based on the function's specifications
3. The execution of the test case
4. The comparison of actual and expected outputs

Performance testing

In software engineering, performance testing is in general testing performed to determine how a system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate, measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

Performance testing is a subset of performance engineering, an emerging computer science practice which strives to build performance into the implementation, design and architecture of a system.

Testing types

Load testing

Load testing is the simplest form of performance testing. A load test is usually conducted to understand the behaviour of the system under a specific expected load. This load can be the expected concurrent number of users on the application performing a specific number of transactions within the set duration. This test will give out the response times of all the important business critical transactions. If the database, application server, etc. are also monitored, then this simple test can itself point towards bottlenecks in the application software.

Stress testing

Stress testing is normally used to understand the upper limits of capacity within the system. This kind of test is done to determine the system's robustness in terms of extreme load and helps application administrators to determine if the system will perform sufficiently if the current load goes well above the expected maximum.

Soak testing

Soak testing, also known as endurance testing, is usually done to determine if the system can sustain the continuous expected load. During soak tests, memory utilization is monitored to detect potential leaks. Also important, but often overlooked is performance degradation. That is, to ensure that the throughput and/or response times after some long period of sustained activity are as good as or better than at the beginning of the test. It essentially involves applying a significant load to a system for an extended, significant period of time. The goal is to discover how the system behaves under sustained use.

Spike testing

Spike testing is done by suddenly increasing the number of or load generated by, users by a very large amount and observing the behaviour of the system. The goal is to determine whether

performance will suffer, the system will fail, or it will be able to handle dramatic changes in load.

Configuration testing

Rather than testing for performance from the perspective of load, tests are created to determine the effects of configuration changes to the system's components on the system's performance and behaviour. A common example would be experimenting with different methods of load-balancing.

Isolation testing

Isolation testing is not unique to performance testing but involves repeating a test execution that resulted in a system problem. Often used to isolate and confirm the fault domain.

Integration testing

Integration testing (sometimes called integration and testing, abbreviated I&T) is the phase in software testing in which individual software modules are combined and tested as a group. It occurs after unit testing and before validation testing. Integration testing takes as its input modules that have been unit tested, groups them in larger aggregates, applies tests defined in an integration test plan to those aggregates, and delivers as its output the integrated system ready for system testing.

Purpose

The purpose of integration testing is to verify functional, performance, and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

Test cases are constructed to test whether all the components within assemblages interact correctly, for example across procedure calls or process activations, and this is done after testing individual modules, i.e. unit testing. The overall idea is a "building block" approach, in which verified assemblages are added to a verified base which is then used to support the integration testing of further assemblages.

Some different types of integration testing are big bang, top-down, and bottom-up. Other Integration Patterns are: Collaboration Integration, Backbone Integration, Layer Integration, Client/Server Integration, Distributed Services Integration and High-frequency Integration.

Big Bang

In this approach, all or most of the developed modules are coupled together to form a complete software system or major part of the system and then used for integration testing. The Big Bang method is very effective for saving time in the integration testing process. However, if the test cases and their results are not recorded properly, the entire integration process will be more complicated and may prevent the testing team from achieving the goal of integration testing.

A type of Big Bang Integration testing is called Usage Model testing. Usage Model Testing can be used in both software and hardware integration testing. The basis behind this type of integration testing is to run user-like workloads in integrated user-like environments. In doing the testing in this manner, the environment is proofed, while the individual components are proofed indirectly through their use.

Usage Model testing takes an optimistic approach to testing, because it expects to have few problems with the individual components. The strategy relies heavily on the component developers to do the isolated unit testing for their product. The goal of the strategy is to avoid redoing the testing done by the developers, and instead flesh-out problems caused by the interaction of the components in the environment.

For integration testing, Usage Model testing can be more efficient and provides better test coverage than traditional focused functional integration testing. To be more efficient and accurate, care must be used in defining the user-like workloads for creating realistic scenarios in exercising the environment. This gives confidence that the integrated environment will work as expected for the target customers.

Top-down and Bottom-up

Bottom Up Testing is an approach to integrated testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

All the bottom or low-level modules, procedures or functions are integrated and then tested. After the integration testing of lower level integrated modules, the next level of modules will

be formed and can be used for integration testing. This approach is helpful only when all or most of the modules of the same development level are ready. This method also helps to determine the levels of software developed and makes it easier to report testing progress in the form of a percentage.

Top-Down Testing is an approach to integrated testing where the top integrated modules are tested and the branch of the module is tested step by step until the end of the related module.

Sandwich Testing is an approach to combine top down testing with bottom up testing.

The main advantage of the Bottom-Up approach is that bugs are more easily found. With Top-Down, it is easier to find a missing branch link

Verification and validation

Verification and Validation are independent procedures that are used together for checking that a product, service, or system meets requirements and specifications and that it full fills its intended purpose. These are critical components of a quality management system such as ISO 9000. The words "verification" and "validation" are sometimes preceded with "Independent" (or IV&V), indicating that the verification and validation is to be performed by a disinterested third party.

It is sometimes said that validation can be expressed by the query "Are you building the right thing?" and verification by "Are you building it right?"In practice, the usage of these terms varies. Sometimes they are even used interchangeably.

The PMBOK guide, an IEEE standard, defines them as follows in its 4th edition

• **"Validation**. The assurance that a product, service, or system meets the needs of the customer and other identified stakeholders. It often involves acceptance and suitability with external customers. Contrast with verification."

• **"Verification.** The evaluation of whether or not a product, service, or system complies with a regulation, requirement, specification, or imposed condition. It is often an internal process. Contrast with validation."

• Verification is intended to check that a product, service, or system (or portion thereof, or set thereof) meets a set of initial design specifications. In the development phase, verification

procedures involve performing special tests to model or simulate a portion, or the entirety, of a product, service or system, then performing a review or analysis of the modelling results.

• Validation is intended to check that development and verification procedures for a product, service, or system (or portion thereof, or set thereof) result in a product, service, or system (or portion thereof, or set thereof) that meets initial requirements.

8.2 TEST CASES

A TEST Plan is a systematic approach to test a system as a machine or software. The plan typically contains a detailed understanding of what the eventual work flow will be UNIT LEVEL plan

S.NO	TEST CASE NAME	TEST CASE DESCRIPTION	EXPECTED OUTPUT	ACTUAL OUTPUT	TEST CASE STATUS	REMARKS
1	Face Recognition	Web camera gets on and captures the face	Face recognition is processed	Face recognition is processed	PASS	
2	Face not recognised	If face does not give properly to the camera, it does not recognise the face	Does not recognise the face	Does not recognise the face	PASS	Give your face properly to the camera
3	Sad Emotion	If the emotion is detected as sad, plays sad music	Plays sad music	Plays sad music	PASS	
4	Happy Emotion	If the emotion is detected as happy, plays happy music	Plays happy music	Plays happy music	PASS	
5	Surprise Emotion	If the emotion is detected as surprise, plays surprise music	Plays surprised music	Plays surprised music	PASS	
6	Disgust Emotion	If the emotion is detected as disgust, plays disgust music	Play disgusted music	Play disgusted music	PASS	
7	Neutral Emotion	If the emotion is detected as neutral, plays neutral music	Plays neutral music	Play neutral music	PASS	
8	Stress Emotion	If the emotion is detected as stress, plays stress music	Plays stress music	Plays stress music	PASS	
9	Scared Emotion	If the emotion is detected as scared, plays scared music	Plays scared music	Plays scared music	PASS	
10	Other than above emotions	If the emotion is detected other than above emotions then it will play the neutral music	Plays neutral music	Plays neutral music	PASS	

9. OUTPUT SCREENS

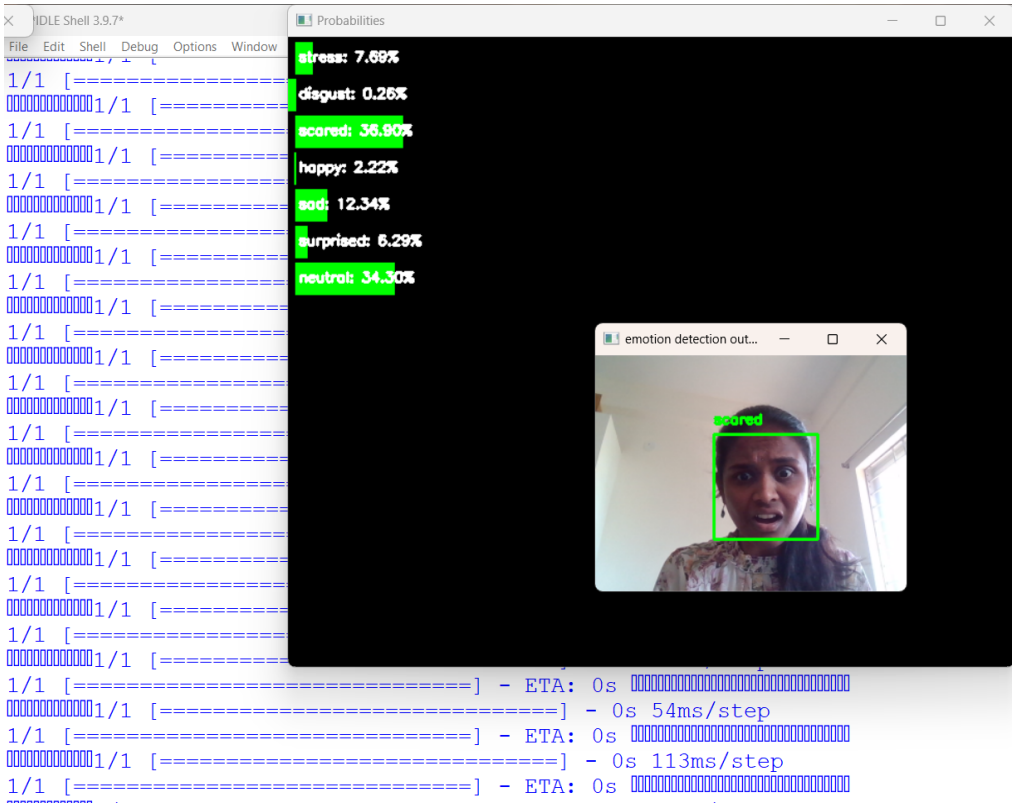

Figure 31: Output face Detected by the Camera (SCARED)

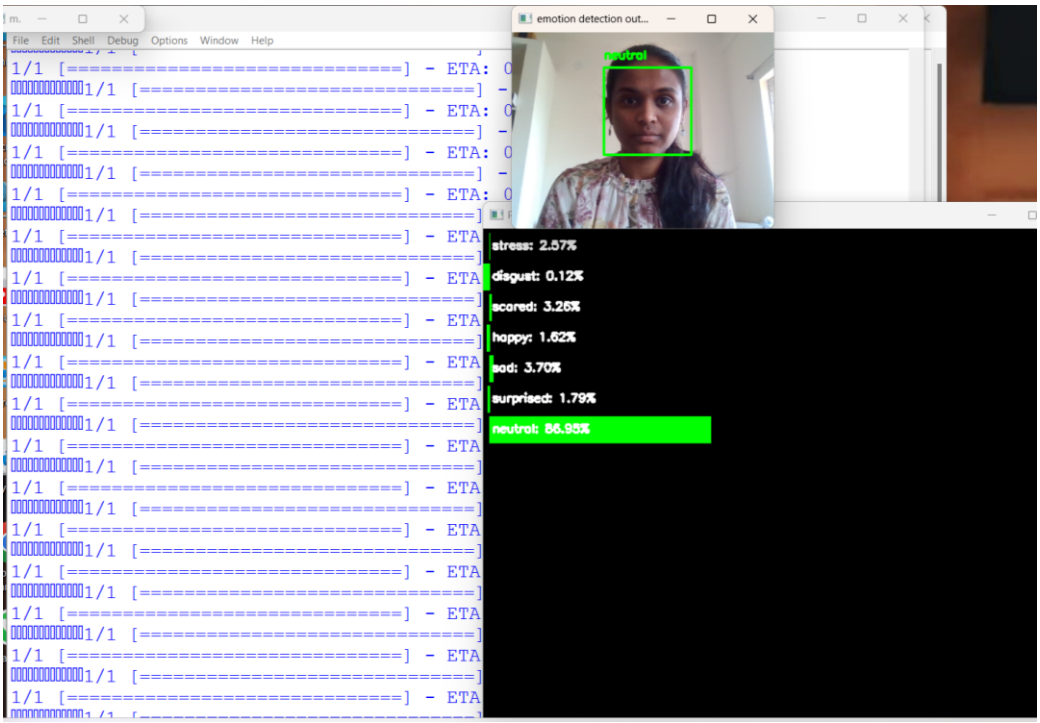

Figure 32: Output face Detected by the Camera (NEUTRAL)

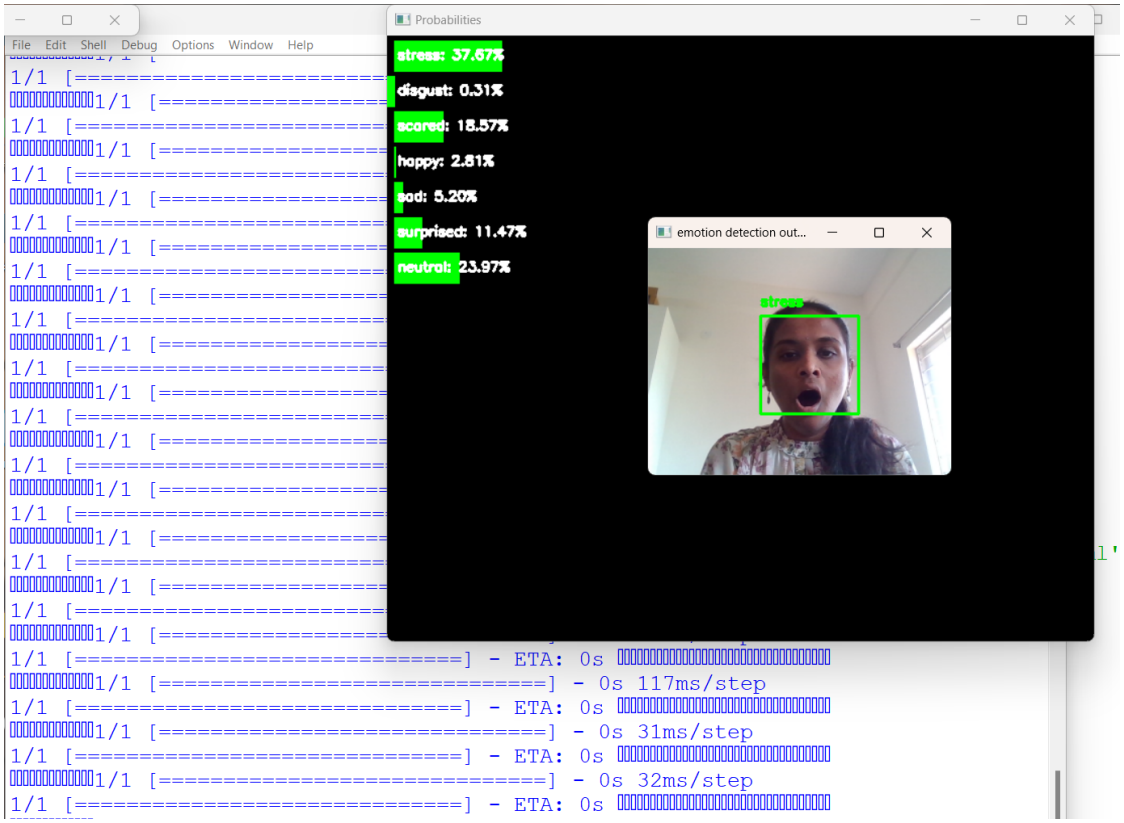

Figure 33: Output face Detected by the Camera (STRESS)

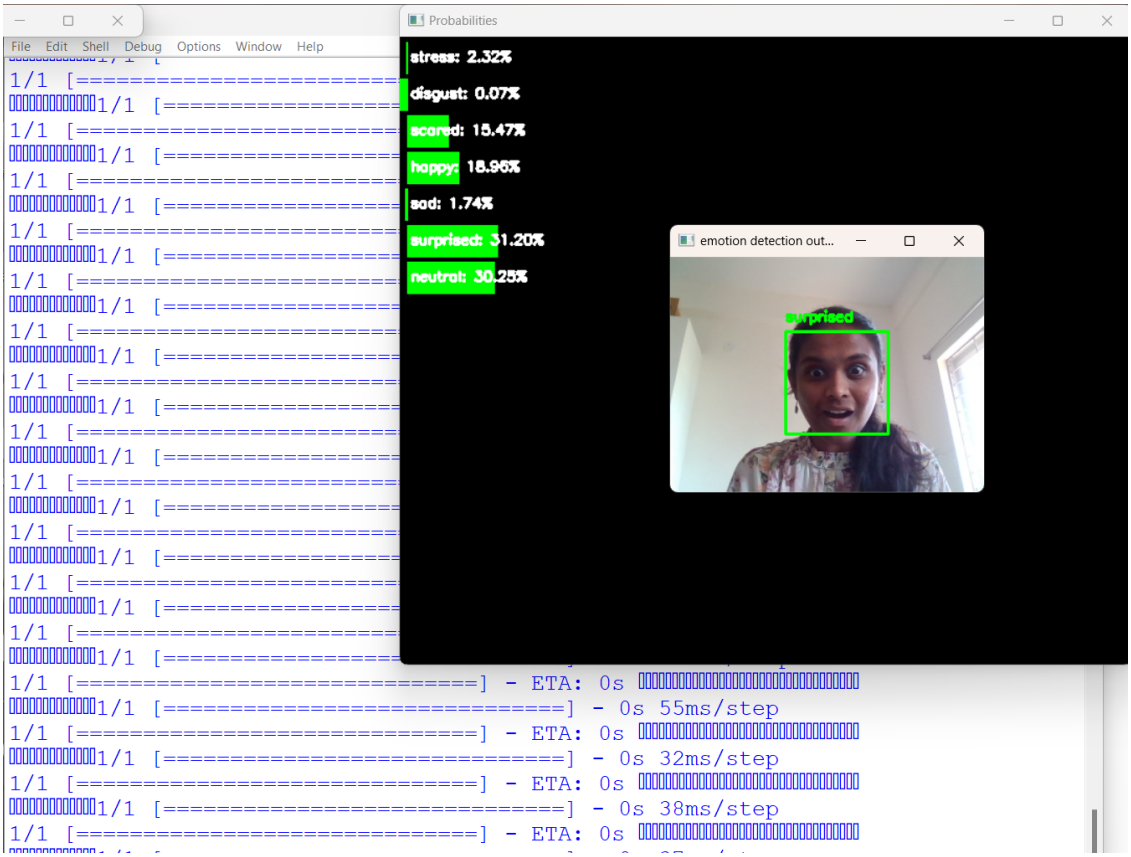

Figure 34: Output face Detected by the Camera (SURPRISED)

10 . CONCLUSION

10.1 PROJECT CONCLUSION

Experimental results have shown that the time required for audio feature extraction is negligibie and songs are stored pre-handed the total estimation time of the proposed system is proportional to the time required for extraction of facial features. Also, the various classes of emotion yield a better accuracy rate as compared to previous existing systems. The computational time taken is 1.000sec which is very less thus helping in achieving a better real time performance and efficiency.

The system thus aims at providing the Windows operating system users with a cheaper, additional hardware free and accurate emotion-based music system. The Emotion Based Music System will be of great advantage to users looking for music based on their mood and emotional behaviour. It will help reduce the searching time for music thereby reducing the unnecessary computational time and thereby in- creasing the overall accuracy and efficiency of the system. Also, with its additional features mentioned above, it will be a complete system for music lovers and listeners.

10.2 FUTURE ENHANCEMENTS

The proposed system might have many functions and it may be user friendly but the proposed system can have further advancement in future. The future scope in this system will be to create a mechanism that will be helpful in music therapy treatment and will provide the music therapist needed to treat patients suffering from disorders such as mental stress, anxiety, acute, depression and trauma. The proposed system is currently available on windows operating system, In future it will be available for the user using different operating system such as iOS, ubuntu etc. and mobile platform as well.

The proposed system tries to avoid unforeseen results generated in the future in extremely poor lighting conditions and very poor camera resolution. In the proposed work only one emotion is detected at a time so that it can be further enhanced to detect mixed emotion.

11 . REFERENCES

[1] P. Póo Argüelles, “Parálisis cerebral infantil,” 2008.

[2] I. C. Valdovinos Lopez, “Evaluación diagnóstica del niño con parálisis cerebral en el tercer nivel de atención,” 2009.

[3] J. M. Valdez, Medicina (BA): organo de la Sociedad Argentina de Investigación Clínica. La Sociedad.

[4] G. Sumathy and A. Renjith, “A survey of vision and speech stimulation for cerebral palsy rehabilitation,” in 2014 International Conference on Control, Instrumentation, Communication and Computational Technologies (ICCICCT), 2014, pp. 1315–1319.

[5] M. González, D. Mulet, E. Perez, C. Soria, and V. Mut, “Vision based interface: an alternative tool for children with cerebral palsy,” in 2010 Annual International Conference of the IEEE Engineering in Medicine and Biology, 2010, pp. 5895–5898.

[6] H. Rahmati, O. M. Aamo, O. Stavdahl, R. Dragon, and L. Adde, “Videobased early cerebral palsy prediction using motion segmentation,” in 2014 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2014, pp. 3779–3783.

[7] M. Betke, J. Gips, and P. Fleming, “The Camera Mouse: visual tracking of body features to provide computer access for people with severe disabilities,” IEEE Trans. Neural Syst. Rehabil. Eng., vol. 10, no. 1, pp. 1–10, Mar. 2002.

[8] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 22, no. 8, pp. 747–757, 2000.