

**EMOTION BASED MUSIC RECOMMENDATION
SYSTEM
USING DEEP LEARNING**

MINI PROJECT REPORT

Submitted by

AARTHI A - 210419104002

MADHUSREE M P - 210419104095

in partial fulfillment for the award of the degree

BACHELOR OF ENGINEERING

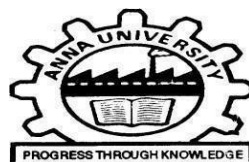
in

COMPUTER SCIENCE AND ENGINEERING



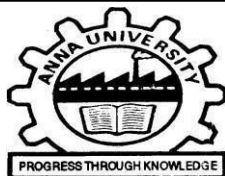
CHENNAI INSTITUTE OF TECHNOLOGY

CHENNAI-600 069



ANNA UNIVERSITY:CHENNAI - 600025

JUNE 2022



ANNA UNIVERSITY:CHENNAI-600025

JUNE 2022

BONAFIDE CERTIFICATE

Certified that this project report “EMOTION BASED MUSIC RECOMMENDATION SYSTEM USING DEEP LEARNING” is the bonafide work of “Aarthi A (210419104002) and Madhusree M P (210419104095)” who carried out the project work under my supervision.

SIGNATURE

Dr. S.PAVITHRA,M.E Ph.D

Head of the Department.

Department of Computer Science

And Engineering.

Chennai Institute of Technology.

Kundrathur,Chennai- 600 069.

SIGNATURE

Ms.V.Gayathri, M.E

Assistant Professor

Department of Computer Science

and Engineering.

Chennai Institute of Technology.

Kundrathur,Chennai- 600 069.

Submitted for the project viva examination held on _____

INTERNAL EXAMINER

EXTERNAL EXAMINER

TABLE OF CONTENTS

CHAPTER	TITLE	PAGE NO
	ACKNOWLEDGEMENT	I
	ABSTRACT	II
I	INTRODUCTION	1
	1.1. ACTION BASED MUSIC RECOMMENDATION SYSTEM	1
	1.2. SYSTEM OVERVIEW	3
	1. 3. SCOPE OF THE PROJECT	3
II	SYSTEM ANALYSIS	4
	2. 1. EXISTING SYSTEM	4
	2. 2. PROPOSED SYSTEM	5
	2. 3. REQUIREMENT SPECIFICATION	5
III	SYSTEM DESIGN	6
	3.1 SYSTEM ARCHITECTURE	6
	3.2 DATA PROCESSING	8
	3.3 CONVOLUTION NEURAL NETWORK	10
	3.4 UML DESIGN	15
IV	MODULES	19
	4.1 EXTRACTING EFFECTIVE FEATURES	19
	4.2 FEATURE-POINT DETECTION	20
	4.3 LIP FEATURE DETECTION	22

	4.4 EYE FEATURE DETECTION	23
	4.5 INTEGRAL IMAGES	23
	4.6 EMOTION EXTRACTION	24
	4.7 MOOD RECOGNIZER	26
	4.8 PLAYLIST CLASSIFIER	28
V	CONCLUSION	32
	5.1 CONCLUSION	32
	5.2 FUTURE ENHANCEMENT	32
	APPENDIX 1	33
	APPENDIX 2	49
	REFERENCES	50

ACKNOWLEDGEMENT

We express our gratitude to our chairman **Shri.P.SRIRAM**, and all trust members of Chennai Institute of Technology for providing the facility and opportunity to do this project as a part of our undergraduate course.

We thank our Principal **Dr.A.RAMESH M.E.**, for his valuable suggestion and guidance for the development and completion of this project

We sincerely thank our Head of the Department **Dr.S.PAVITHRA M.E., Ph.D.** Department of Computer Science Engineering for having provided us valuable guidance, resources and timely suggestions through our work.

We sincerely thank our project guide **Ms.V.Gayathri M.E** Assistant professor, Department of Computer Science Engineering for having provided us valuable guidance , resources and timely suggestions through our work.

We wish to extend our sincere thanks to **All Faculty members** And **Lab Instructors** of the Department of Computer Science Engineering for their valuable suggestions and their kind co-operation for the successful completion of our project.

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. 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. The user would not have to waste any time searching or looking for songs and the best track matching the user's mood is detected, and songs would be shown to the user according to his/her mood. The project is developed using the (SVM) Algorithm which enhances the Accuracy of the Application. The (CNN) Convolution Neural Networks Technology is used to integrate the images based on the pixel rate using the Rectangular Figures to detect the emotions using the metrics. The Music recommendation phase is the most important aspect of the proposed application because of the K-Means Algorithm that encompasses the audio features and the lyrical understanding and classifies the playlist based on the emotions. Music recommendations have existed for a long time. The K-Means and Scalar Vector Machine Algorithm proved to help change the mood of a person if it is a negative one such as sad, or angry. These K-Means and Scalar Vector Machine Algorithm reduced the overall computational time and the cost of the designed system to the optimized level.

CHAPTER 1

INTRODUCTION

1. 1. EMOTION BASED MUSIC RECOMMENDATION

Music plays a very primary role in elevating an individual's life as it is an important medium of entertainment for music lovers and listeners. In today's world, there is increasing advancement in the field of multimedia and technology, various music players have been developed with features like fast forward, reverse, variable playback speed, genre classification, streaming playback with multicast streams and volume modulation, etc. These features might satisfy the user's basic requirements, but the user has got to face the task of manually browsing the playlist of songs and choosing songs that supported their current mood and behavior. Emotion-based music player is a novel approach that helps the user to play songs according to the emotions of the user automatically. It recognizes the facial emotions of the user and plays the songs according to their emotion.

The emotions are recognized using a machine learning method EMO algorithm. The human face is an important organ of an individual's body and it especially plays an important role in the extraction of an individual's behaviors and emotional state. The webcam captures the image of the user. It then extracts the facial features of the user from the captured image. Facial expression is categorized into 2, smiling and not smiling. The foremost concept of this project is to automatically play songs based on the emotions of the user. It aims to provide user-preferred music with respect to the emotions detected. Each sub-directory contains songs that correspond to the emotion. Songs in the sub folders can be changed/replaced or deleted by the programmer depending on the requirements of the user. At times it is possible that users might like different kinds of songs in a certain mood. For example, when a user's emotion

is detected to be Sad, it is totally the user's choice what kind of mood he/she wants. There are two possibilities in this scenario:

- a) User wants to continue his/her sad mood.
- b) User wants to elevate his/her mood and wants to be happy.

Therefore, depending on the choice of users the songs in the sub directories can be changed.

1.1.1 Face Detection

Face detection has progressed from rudimentary computer vision techniques to advances in machine learning (ML) to increasingly sophisticated artificial neural networks (ANN) and related technologies; the result has been continuous performance improvements. It now plays an important role as the first step in many key applications

-- including face tracking, face analysis, and face recognition. Face detection has a significant effect on how sequential operations will perform in the application.

In face analysis, face detection helps identify which parts of an image or video should be focused on to determine age, gender, and emotions using facial expressions. In a facial recognition system -- which maps an individual's facial features mathematically and stores the data as a faceprint -- face detection data is required for the algorithm that discerns which parts of an image or video are needed to generate a faceprint. Once identified, the new faceprint can be compared with stored faceprints to determine if there is a match.

1.2. SYSTEM OVERVIEW

Face detection is a type of computer vision technology that is able to identify people's faces within digital images. This is very easy for humans, but computers need precise instructions.

Facial recognition involves identifying the face in the image as belonging to person *X* and not person *Y*. It is often used for biometric purposes, like unlocking your smartphone.

Facial analysis tries to understand something about people from their facial features, like determining their age, gender, or the emotion they are displaying.

Facial tracking is mostly present in video analysis and tries to follow a face and its features (eyes, nose, and lips) from frame to frame.

1.3. SCOPE OF THE PROJECT

Project Emo player (an emotion-based music player) is a novel approach that helps the user to automatically play songs based on the emotion of the user. It recognizes the facial emotions of the user and plays the songs according to their emotion. The emotions are recognized using a machine learning method Support Vector Machine(SVM)algorithm. The human face is an important organ of an individual's body and it especially plays an important role in the extraction of an individual's behaviors.

CHAPTER 2

SYSTEM ANALYSIS

2.1. EXISTING SYSTEM

Currently, there are many existing music player applications. Some of the interesting applications among them are:

- Saavan and Spotify - These application gives good user accessibility features to play songs and recommends user with other songs of similar genre.
- Moodfuse - In this application, user should manually enter mood and genre that wants to be heard and moodfuse recommends the songs-list.
- Steromood - User should select his mood manually by selecting the moods from the list and the application plays music from YouTube.

2.1.1. DISADVANTAGES OF EXISTING SYSTEM

- Existing algorithms are slow, increase cost of the system by using additional hardware (e.g. EEG systems and sensors) and have quite very less accuracy.
- This paper presents an algorithm that not only automates the process of generating an audio playlist, but also to classify those songs which are newly added and the main task is to capture current mood of person and to play song accordingly.
- This enhances the system's efficiency, faster and automatic. The main goal is to reduce the overall computational time and the cost of the designed system.

2. 2. PROPOSED SYSTEM

The proposed system can detect the facial expressions of the user and based on the 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.

2. 2. 1. Advantages of the Proposed System

- Ease of use.
- Mixed mood detection.
- Improved accuracy.
- Reduced computational time.

2.3. REQUIREMENTS SPECIFICATION

2.3. 1. Hardware Requirements

- Processor : Pentium Dual Core 2.00GHZ
- Hard Disk : 500 GB
- RAM : 4GB (minimum)
- Keyboard : 110 keys enhanced

2. 3. 2. Software Requirements

- MATLAB 8.6 Version
- tensorflow 2:0: This is the latest version of the deep learning framework developed and maintained by Google
- Scikit learn: To use ML related utilities.
- Pandas and NumPy: A must for all ML tasks in python. We will use this to manipulate data.
- Matplotlib: Visualizing model performance and data

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

In Figure 3.1 shows the step by step process of How the System is being Designed based on the algorithm

3.1.1 Upload Image

An image should be uploaded which may contain any of the four emotions that our model will be able to detect. The four emotions are: Happy, Sad, Energetic and Calm.

3.1.2 Analyse Image

The Analyse phase where the metrics are analysed the uploaded image is analysed by the convolution neural network technology the Data is initially collected and then trained in the training phase then the datasets are tested in the inference phase based on this analysis the new data or image that is uploaded is detected and analysed based on the trained datasets.

If the analysis Phase is not completed successfully because any improper dataset gathering like the image with low possibility of metrics then the analysis phase is stoped and returns to the uploading phase to reupload the image.

3.1.3 Mood Detection

The mood is recognized with the help of the analysis. There are different moods based on the analysis that is being collected from the data collection the mood includes happy, sad, neutral, angry, anxious are the basic human emotions these human emotions are detected by the features of face such as if the person smiles then the emotion is recognized as happiness, If the user wrinkles the eyes and lips then the emotion is taken into account as sadness same as follows for the neutral if there is no any metrical change in the parameters gathered from the face then the emotion is recognized as neutral this metrics decides the emotions that is detected based on the face actions.

If the Emotions are disturbed by the user by from the Data gathered then it reaches to the capturing face itself to capture the image, then to the analysis then comes to the Mood Detection Phase.

3.1.4 Provide Playlists

The next phase is the playlist displaying phase the mood is recognized and the window pops up to display the playlist in the player in the VLC media Player the the the playlist consists of the list of songs gathered based on the emotions.

3.1.5 Play selected Songs

The Playlist is displayed and the song that the most connecting to the emotions are played selectively this is because of the algorithm. The songs that are played are the classified based on the emotions gathered in the analysis and mood detection Phase.

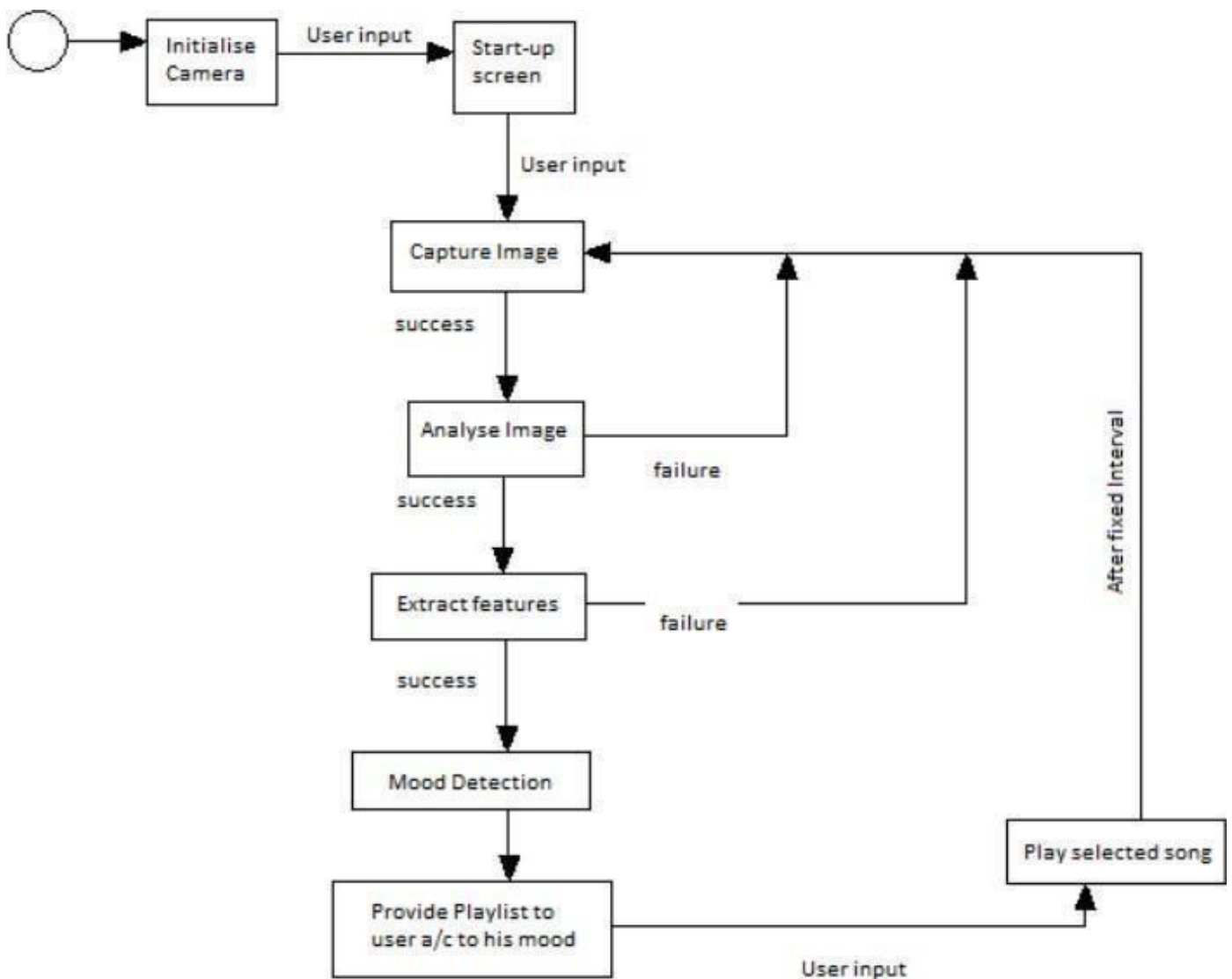


FIGURE 3.1 Architecture of the proposed system

3.2 DATA PROCESSING

1. Image pyramid
2. Histogram of Oriented Gradients
3. Linear Classifier The data that are obtained are decomposed into the sampling image using image pyramid into multiple scales.

The use of this technique is simply to extract features while reducing the noise and the other factors. The low pass image pyramid technique (also known as Gaussian pyramid) consists of smoothing the frame and subsampling it by decreasing its resolution, the process needs to be repeated a few times in order to obtain a perfect result that in the end of the process we obtain a frame similar to the original one but with a decreased resolution and an increased smoothing level.

Commonly used to detect objects in images in the field of image processing, HOG is a feature descriptor, a technique that counts occurrences of gradient orientation in a localized portion of an image. The main objective of using this technique is to describe the face within the image with a set of distributions of intensity gradients.

3.2.1 Data Set

The Raw dataset is downloaded one by one from Google images for seven emotions. Extra dataset is taken from Kaggle datasets for facial expression detection. There are three different data sets such as Data collection, Training data sets, Inference data sets. In data collection process the emotions are collected using open CV and the respective data is trained.

3.2.2 Trained Dataset

Before processing the model, the training and testing phases is undergone. Trained dataset are those which are taught to the model or which learn At the time of training, system takes dataset of faces with their respective expression; eye should be in centre location mostly and learns a set of weights, which splits the facial expressions for classification.

For training, the sequence:

1. Spatial normalization
2. Synthetic samples generation
3. Image cropping
4. Down-sampling
5. Intensity normalization.

3.2.3 Test Data

At the time of testing, classifier takes images of face with respective eye center locations, and it gives output as predicted expression by using the weights learned during training.

3.3 Convolution Neural Network

In the Figure 3.3.1 There are two main parts to a CNN architecture

- A convolution tool that separates and identifies the various features of the image for analysis in a process called as Feature Extraction
- A fully connected layer that utilizes the output from the convolution process and predicts the class of the image based on the features extracted in previous stages.

3.3.2 Convolution Layers

There are three types of layers that make up the CNN which are the convolutional layers, pooling layers, and fully-connected (FC) layers. When these layers are stacked, a CNN architecture will be formed. In addition to these three layers, there are two more important parameters which are the dropout layer and the activation function which are defined below.

3.3.3 Convolutional Layer

This layer is the first layer that is used to extract the various features from the input images. In this layer, the mathematical operation of convolution is performed between the input image and a filter of a particular size $M \times M$. By sliding the filter over the input image, the dot product is taken between the filter and the parts of the input image with respect to the size of the filter ($M \times M$).

The output is termed as the Feature map which gives us information about the image such as the corners and edges. Later, this feature map is fed to other layers to learn several other features of the input image.

3.3.4 Pooling Layer

In most cases, a Convolutional Layer is followed by a Pooling Layer. The primary aim of this layer is to decrease the size of the convolved feature map to reduce the computational costs. This is performed by decreasing the connections between layers and independently operates on each feature map. Depending upon method used, there are several types of Pooling operations.

In Max Pooling, the largest element is taken from feature map. Average Pooling calculates the average of the elements in a predefined sized Image section. The total sum of the elements in the predefined section is computed in Sum Pooling. The Pooling Layer usually serves as a bridge between the Convolutional Layer and the FC Layer.

3.3.5 Fully Connected Layer

The Fully Connected (FC) layer consists of the weights and biases along with the neurons and is used to connect the neurons between two different layers. These layers are usually placed before the output layer and form the last few layers of a CNN Architecture.

In this, the input image from the previous layers are flattened and fed to the FC layer. The flattened vector then undergoes few more FC layers where the mathematical functions operations usually take place. In this stage, the classification process begins to take place.

3.3.6 Dropout

Usually, when all the features are connected to the FC layer, it can cause overfitting in the training dataset. Overfitting occurs when a particular model works so well on the training data causing a negative impact in the model's performance when used on a new data.

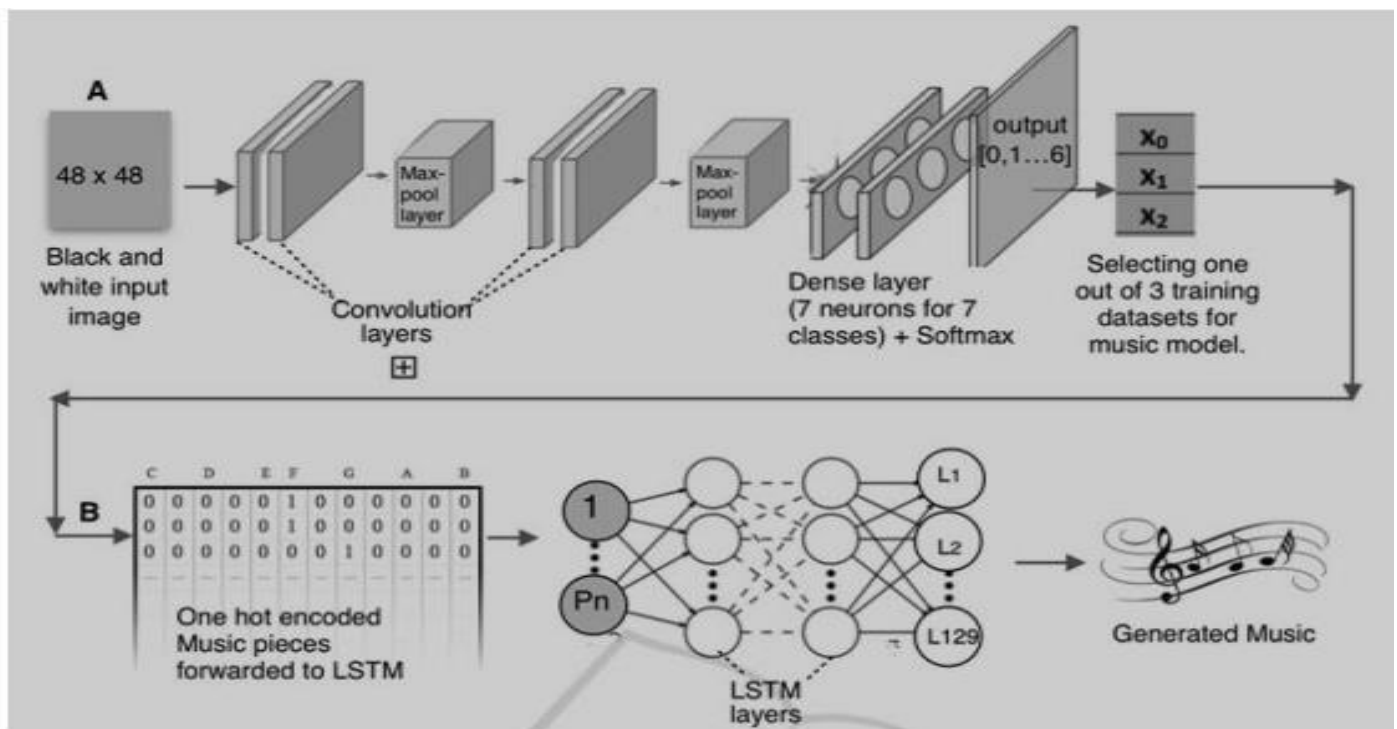
To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during training process resulting in reduced size of the model. On passing

a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network.

3.3.7 Activation Functions

Finally, one of the most important parameters of the CNN model is the activation function. They are used to learn and approximate any kind of continuous and complex relationship between variables of the network. In simple words, it decides which information of the model should fire in the forward direction and which ones should not at the end of the network.

It adds non-linearity to the network. In Figure 4.3.6 commonly used activation functions such as the ReLU, Softmax, tanH and the Sigmoid functions. Each of these functions have a specific usage. For a binary classification CNN model, sigmoid and softmax functions are preferred an for a multi-class classification, generally softmax us used.



CONVOLUTION NEURAL NETWORK LAYERED ARCHITECTURE

FIG 3.3.6

3.3.8 LeNet-5 CNN Architecture

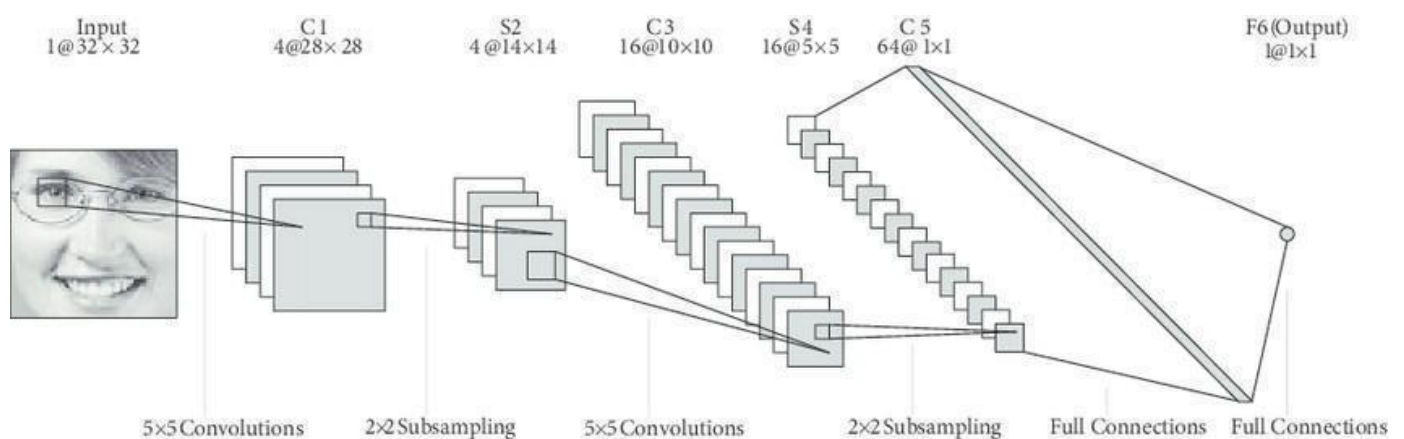
In the Figure 3.3.7 It consists of 7 layers. The first layer consists of an input image with dimensions of 32×32 . It is convolved with 6 filters of size 5×5 resulting in dimension of $28 \times 28 \times 6$. The second layer is a Pooling operation which filter size 2×2 and stride of 2. Hence the resulting image dimension will be $14 \times 14 \times 6$.

Similarly, the third layer also involves in a convolution operation with 16 filters of size 5×5 followed by a fourth pooling layer with similar filter size of 2×2 and stride of 2. Thus, the resulting image dimension will be reduced to $5 \times 5 \times 16$.

Once the image dimension is reduced, the fifth layer is a fully connected convolutional layer with 120 filters each of size 5×5 . In this layer, each of the 120 units in this layer will be connected to the 400 ($5 \times 5 \times 16$) units from the previous layers. The sixth layer is also a fully connected layer with 84 units. The final seventh layer will be a softmax output layer with 'n' possible classes depending upon the number of classes in the dataset

Architecture of proposed Convolution Neutral Network. It has five layers: first layer is convolution, second layer is sub-sampling, the third layer is convolution, fourth layer is sub-sampling, fifth layer is fully connected layer and final responsible for classifying facial image

FIG 3.3.7



ARCHITECTURE OF COVOLUTION

3.3.9 SVM Classifier

SVM is the high-tech binary classifier based on the large margin principle. Given labelled instances from two different classes, SVM classifier finds the optimal separating hyper plane which maximizes the distance between support vectors and the hyper plane. Instances closest to the hyper plane whose labels are most likely to be confused are the support vectors. Therefore, the SVM has better classification performance as it focuses on the difficult instances. Both K-NN and SVM are applicable to single feature vector representations as well as pair wise similarity values. In the second case, a kernel matrix is built from pair wise similarity values which can be used directly by the SVM.

3.3.10 K-Nearest Neighbours

K-Nearest Neighbours is one of the most basic yet essential classification algorithms in Machine Learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining and intrusion detection. It is widely disposable in real-life scenarios since it is non-parametric, meaning, it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

3.4 UML DESIGN

3.4.1 UseCase Diagram

In In the Figure 3.4.1 the useCases for the emotion based music Player is explained
The user inputs are gathered by the capturing phase and the analysis is done and the
emotion is detected and music is played.

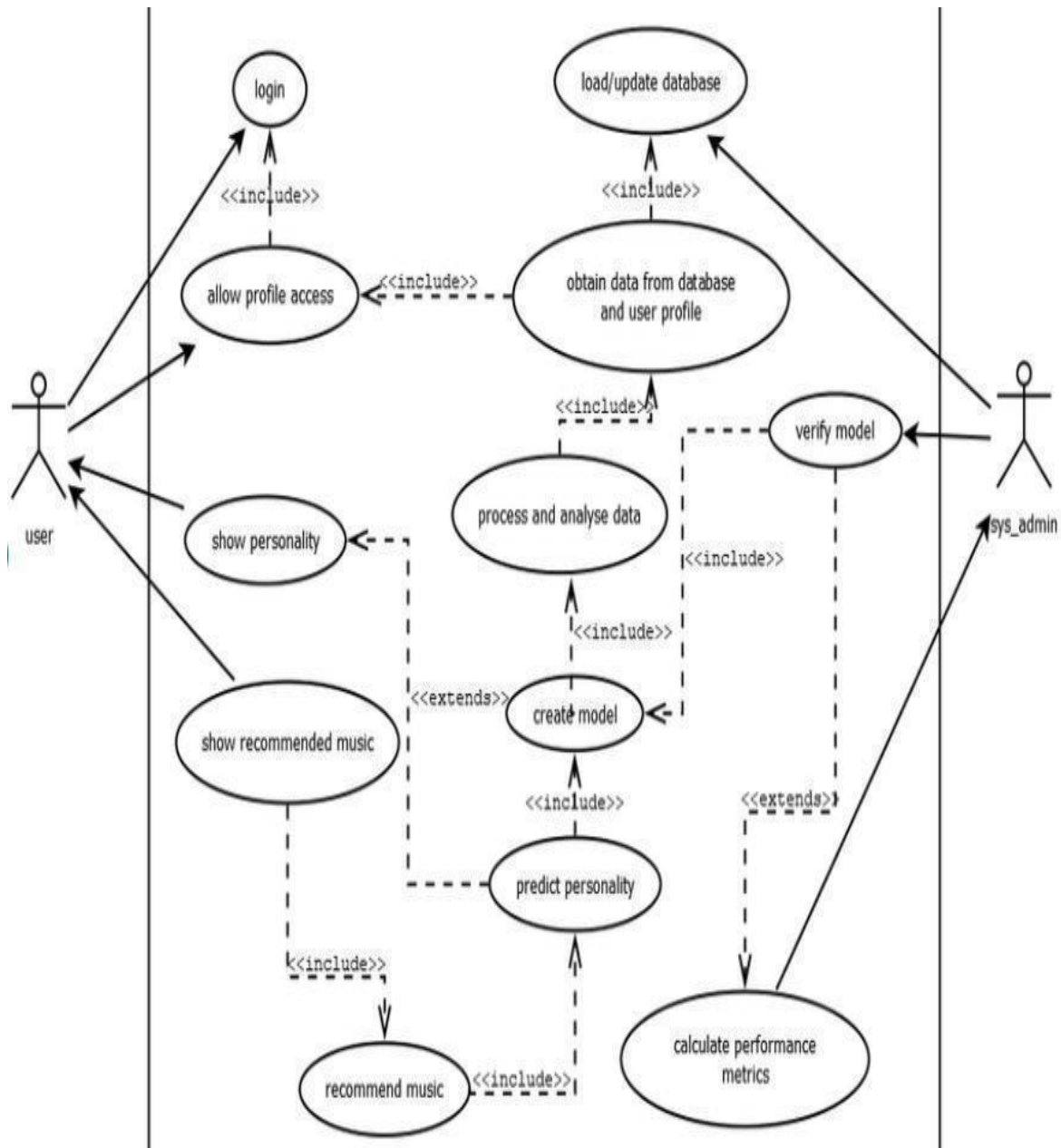


FIGURE 3.4.1

UseCase Diagram For the Proposed System

3.4.2 CLASS DIAGRAM

In the Figure 3.4.2 The gesture are captured and the Data is analysed and the Identify the Data based on the trained DataSets and the Player is activated and the player configurations takes place.

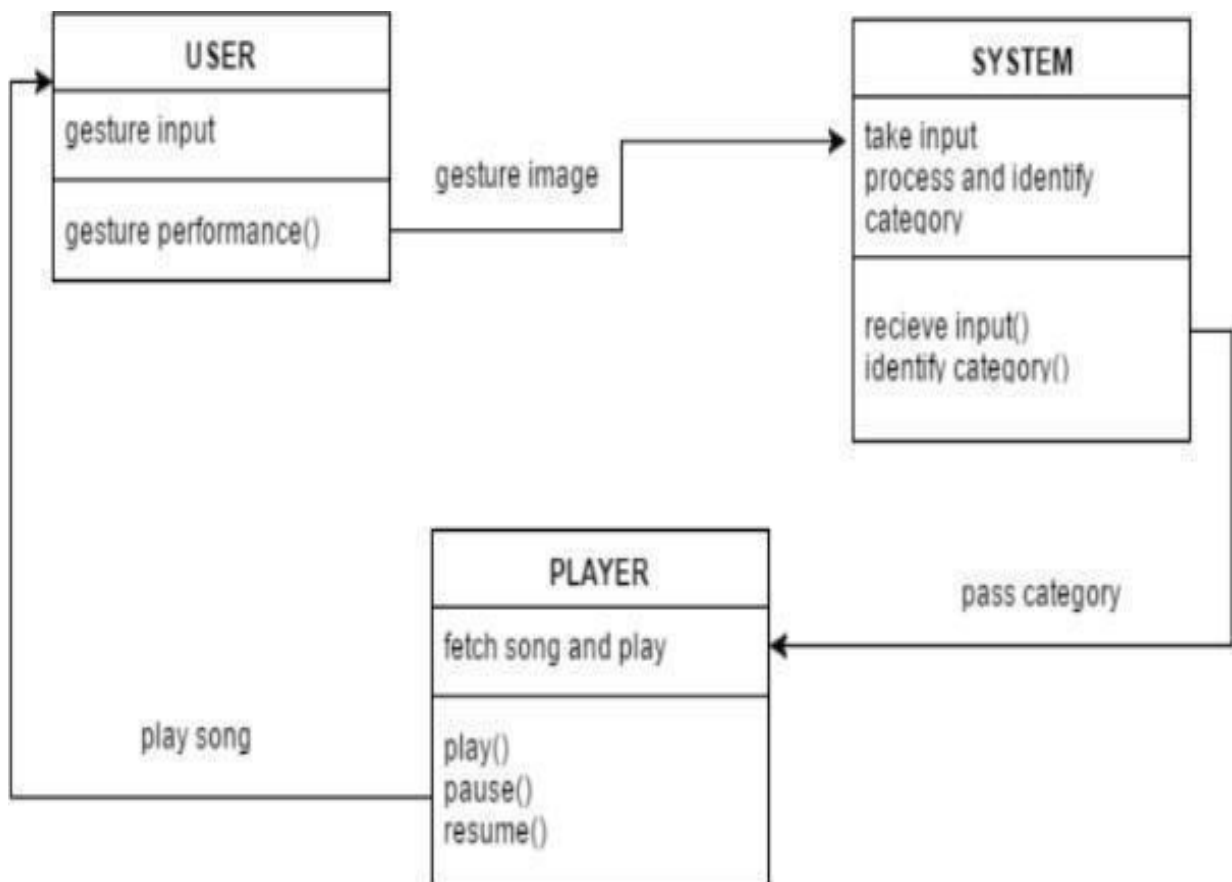


FIGURE 3.4.2

Class Diagram For the Proposed System

3.4.3 SEQUENCE DIAGRAM

The Figure is a type of interaction diagram because it describes how—and in what order—a group of objects works together. In the Figure explains the flow of the application First the the user intraction is the first process then the web cam is the next procedure the

then process of identifying the datasets after comes the player that is the music player system this songs are played from the local storage.

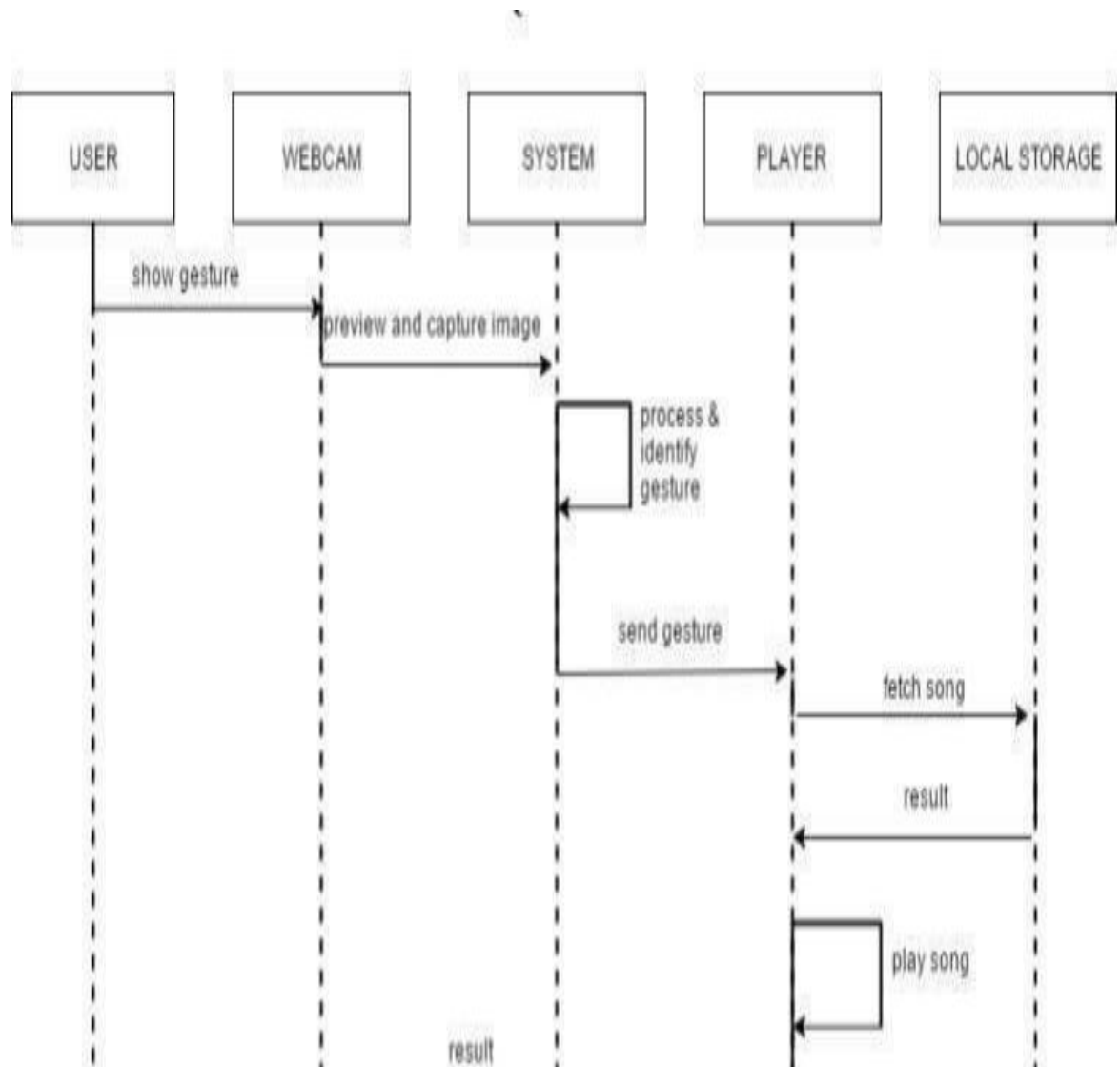


FIGURE 3.4.3

Sequence Diagram For the Proposed System

In the Figure 3.4.3 the flow starts from the user to show the gestures and moves on to the webcam for capturing the image then the system identifies the image for analysis, after the analysis the particular metrics are captured from the image for the classification of emotion. Then the gesture is sent to the player and the playlist is created to fetch the song from local storage and the media player starts running to play the song. This is viewed by the user.

3.4.4 ACTIVITY DIAGRAM

In the Figure 3.4.4 step by step flow for the application is explained first the user emotion is gathered then the music classifier and generate the playlist based the classified algorithm and play the music when the condition gets satisfied .

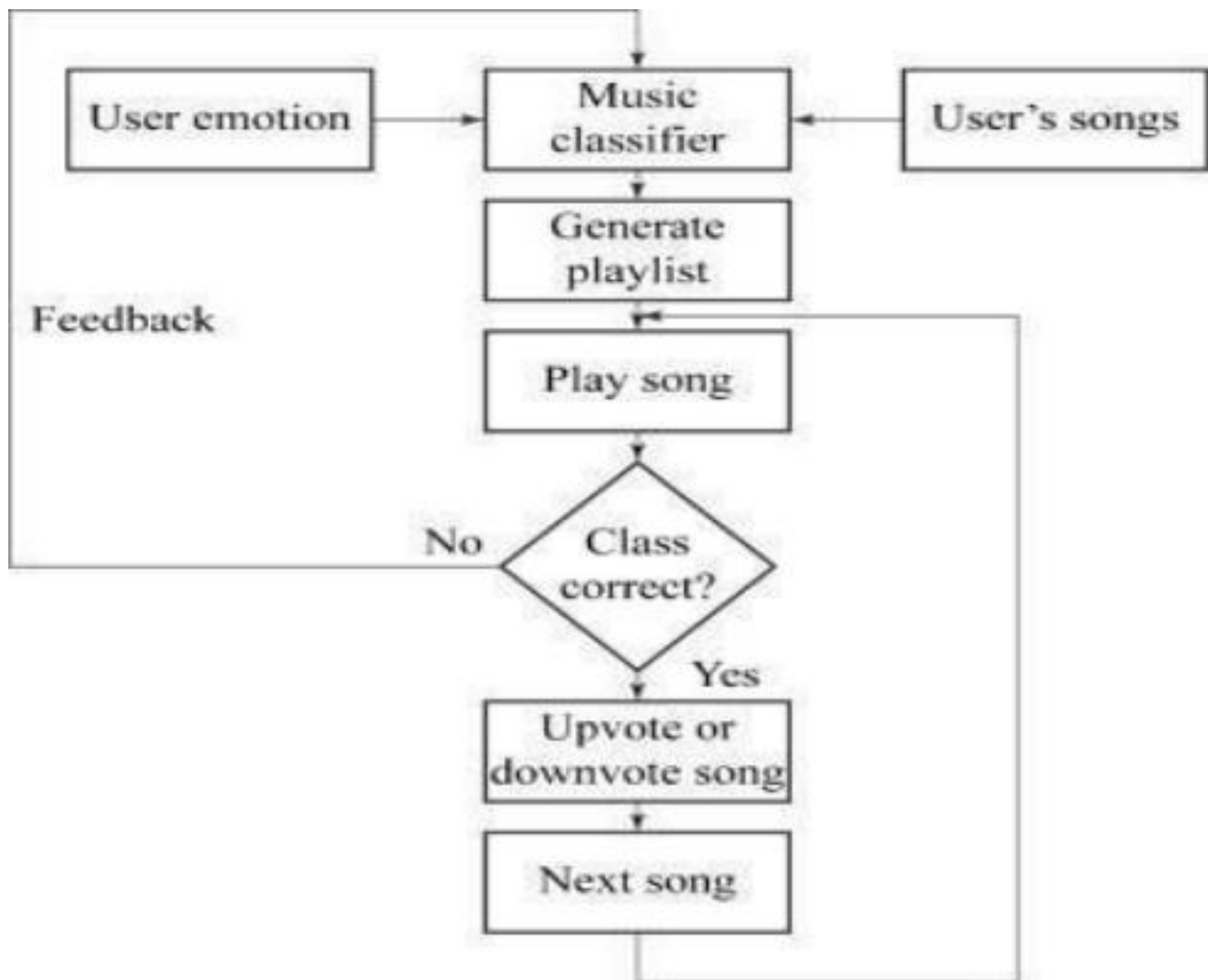


FIGURE 3.4.4

Activity Diagram For the Proposed System

CHAPTER 4

MODULE DESCRIPTION

4.1 Extracting Effective Features

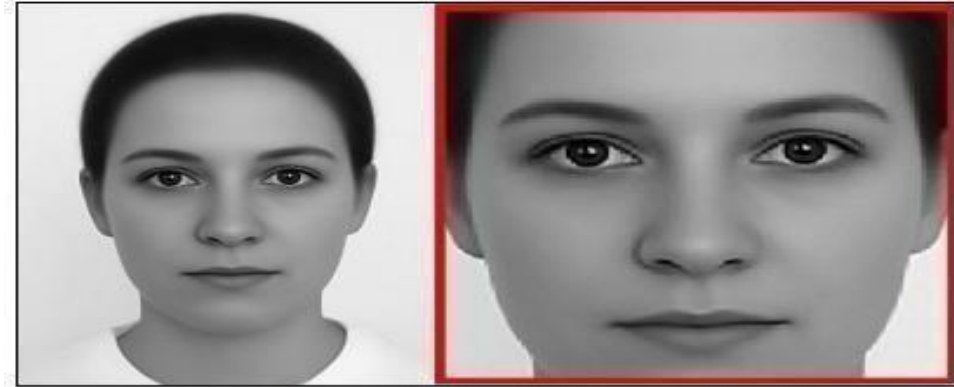


FIG 4.1

Extracting the Effective Features

In Figure 4.1 the Extracting Effective Features In the modules, first System will capture the image from webcam or suitable device. Then the input image is first checked for the facial features. In case if the image does not contain human features, then it does not detect it. If the input image contains Human features, then it detects the features

All human faces share some similarities. If you look at a photograph showing a person's face, you will see, for example, that the eye region is darker than the bridge of the nose. The cheeks are also brighter than the eye region. We can use these properties to help us understand if an image contains a human face.

A simple way to find out which region is lighter or darker is to sum up the pixel values of both regions and comparing them. The sum of pixel values in the darker region will be smaller than the sum of pixels in the lighter region.

In Figure 4.1.1 pixels are often represented in the RGB color model. RGB stands for **R**ed **G**reen **B**lue. Each pixel is a mix of those three colors. RGB is great at modeling all the colors humans perceive by combining various amounts of red, green, and blue.

In grayscale (black and white) images, each pixel is a single number, representing the amount of light, or intensity, it carries. In many applications, the range of intensities is from 0 (black) to 255 (white). Everything between 0 and 255 is various shades of gray.

4.2 Feature-Point Detection



FIG 4.2
Feature -Point Detection

In the Figure 4.2 the Feature-Point Detection In this module, the feature points are detected automatically. For face detection, first we convert binary format image from RGB format image. For converting binary image, we calculate the average value of RGB for each pixel and if the average value is below than 110, we replace it by black pixel and otherwise we replace it by white pixel. By this method, we get a binary image from RGB image. Then next stage is to find the forehead from the binary image. System will start scan from the middle of the image, after that it will look for continuous white pixels after a continuous black pixel. In this we want to find the maximum width of the white pixel by searching vertical both left and right site.

Then, if the new width is smaller half of the previous maximum width, then we break the scan because in that case we will reach to the eyebrow. Then we cut the face from the starting position of the forehead and its height will be 1.5 multiple of its width. In this processed image we will only have eyes, nose and lip. Then we will cut the RGB image according to the binary image. This stage can also be achieved using Applying certain operations to an image produces information that could be considered features as well. Computer vision and image processing have a large collection of useful features and feature extracting operations.

4.2.1 RECTANGULAR FIGURES

In the Figure 4.2.1, there are 4 basic types of rectangular features:

1. Horizontal feature with two rectangles
2. Vertical feature with two rectangles
3. Vertical feature with three rectangles
4. Diagonal feature with four rectangles

The value of the feature is calculated as a single number: the sum of pixel values in the black area minus the sum of pixel values in the white area. For uniform areas like a wall, this number would be close to zero and won't give you any meaningful information.

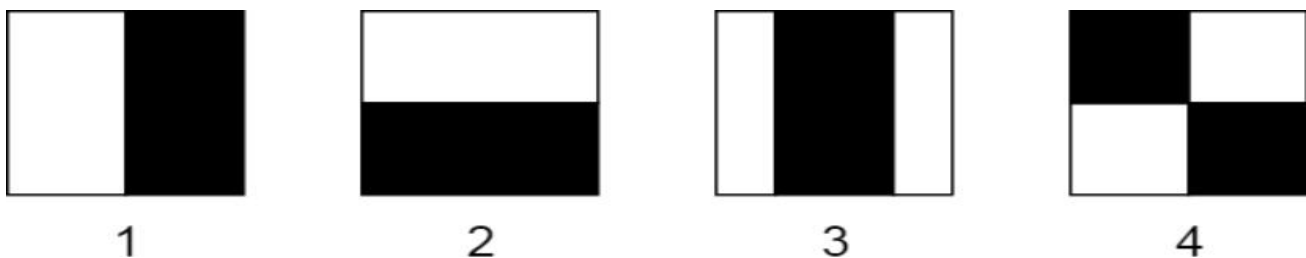


FIG 4.2.1

rectangle features

The Rectangular feature needs to give you a large number, meaning that the areas in the black and white rectangles are very different. There are known features that perform very well to detect human faces.

4.2 Lip Feature Detection

In Figure 4.3 the Lip Feature Detection Up till now we will have an image which contains lip portion of the face. Now the next step is to extract the expression features from lip .To extract the feature we just have to measure the distance between upper lip & lower lip. Also the system will consider the position of contour points of lip. By lip feature we can significantly determine two features happy & surprise mood, anger, sad.

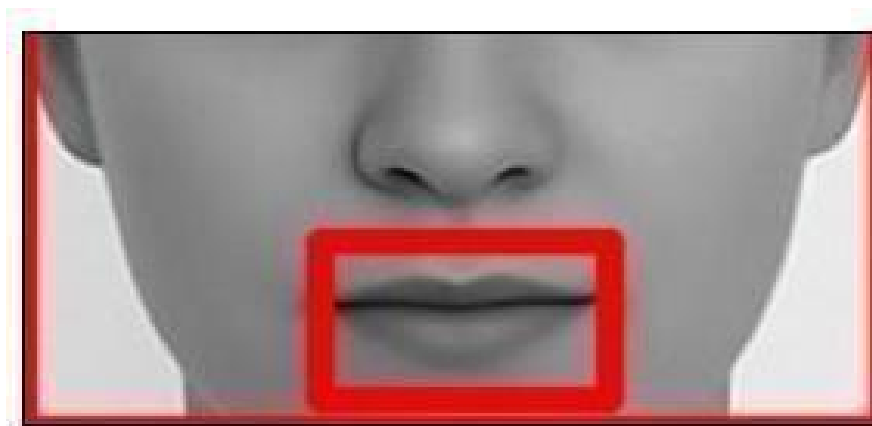


FIG 4.3

Lip Detection using the metrics



FIG 4.3.1

Rectangular Nose Detection

In Figure 4.3.1 the nose region is darker than the region below. This property to find which areas of an image give a strong response (large number) for a specific feature gives a strong response when applied to the bridge of the nose. The Figure 4.3.1 combine many of these features to understand if an image region contains a human face.

4.3 Eye Feature Detection

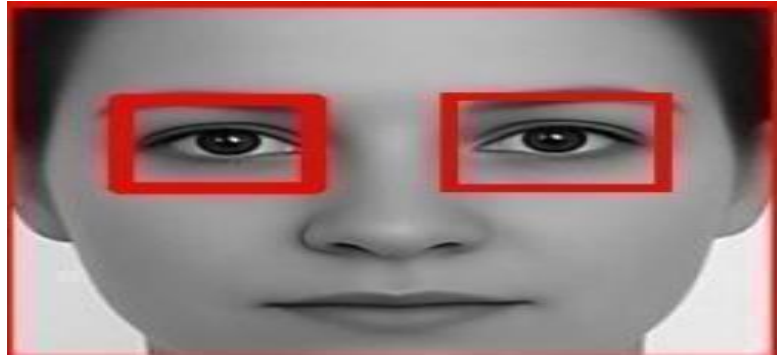


FIG 4.4

Detecting the Eye Features

In the Figure 5.4 the Eye Feature Detection Even with an upper face we can obtain certain facial feature like sleepy and surprised. For this again we have to calculate distance major axis and minor axis considering that eye as an ellipse.



FIG 4.4.1

Eye Detection using Rectangular Figures

In the Figure 4.4.1 the eye region is darker than the region below. This property to find which areas of an image give a strong response

4.4 Integral Images

An integral image (also known as a summed-area table) is the name of both a data structure and an algorithm used to obtain this data structure. It is used as a quick and efficient way to calculate the sum of pixel values in an image or rectangular part of an image.

In Figure 4.5, the value of each point is the sum of all pixels above and to the left, including the target pixel image can be calculated in a single pass over the original image. This reduces

summing the pixel intensities within a rectangle into only three operations with four numbers, regardless of rectangle size

1	3	7	5		1	4	11	16
12	4	8	2		13	20	35	42
0	14	16	9		13	34	65	81
5	11	6	10		18	50	87	113

FIG 4.5

Calculating an integral image from pixel values

In Figure 4.1.1 The sum of pixels in the rectangle $ABCD$ can be derived from the values of points A , B , C , and D , using the formula $D - B - C + A$.

1	4	11	16
13	20	35	42
13	34	65	81
18	50	87	113

A B
 C D
 $ABCD = ?$

FIG 4.5.1 The sum of pixels

4.5 Emotion Extraction

In the Figure 4.6 user is captured with the help of a camera/webcam. Once the picture captured, the frame of the captured image from webcam feed is converted to a grayscale image to improve the performance of the classifier, which is used to identify the face present in the picture. Once the conversion is complete, the image is sent to the classifier algorithm which, with the help of feature extraction techniques can extract the face from the frame of the web

camera feed. From the extracted face, individual features are obtained and are sent to the trained network to detect the emotion expressed by the user. These images will be used to train the classifier so that when a completely new and unknown set of images is presented to the classifier, it is able to extract the position of facial landmarks from those images based on the knowledge that it had already acquired from the training set and return the coordinates of the new facial landmarks that it detected. The network is trained with the help of CK extensive data set. This is used to identify the emotion being voiced by the user

To increase the accuracy and an aim to obtain real time performance only features of eyes and mouth are appropriate enough to depict the emotions accurately. For extracting the features of mouth and eyes certain calculations and measurements are taken into consideration. Equations (1), (2), (3) and (4) illustrate the bounding box calculations for extracting features of a mouth.

$$X(\text{start pt of mouth}) = X(\text{mid pt of nose}) - (X(\text{end pt of nose}) - (X_{\text{startpt of nose}}))$$

$$(1) X(\text{end pt of mouth}) = X(\text{mid pt of nose}) + ((X_{\text{endpt of nose}}) - (X_{\text{startpt of nose}}))$$

$$(2) Y(\text{start pt of mouth}) = Y(\text{mid pt of nose}) + 15$$

$$(3) Y(\text{end pt of mouth}) = Y(\text{start pt of mouth})$$

$$+ 103 \quad (4)$$

Where $(X(\text{start pt of mouth}), Y(\text{start pt of mouth}))$ and $(X(\text{end pt of mouth}), Y(\text{end pt of mouth}))$ illustrates start and end points of the bounding box for mouth respectively, $(X(\text{mid pt of nose}), Y(\text{mid pt of nose}))$ illustrates midpoint of noise and $((X_{\text{endpt of nose}}, (X_{\text{startpt of nose}}))$ illustrates end and start point of noise. Classification is performed using Support Vector Machine (SVM) which classifies it into 7 classes of emotions



FIGURE 4.6 Facial Metric to extract the Emotions

4.6 Mood Recognizer

In standard classification, a training data set where. The purpose is classification rule that can best predict the labels for unseen data. Knearest neighbour (K-NN) , support vector machine (SVM) and GMM classifier are most popular choices for classifiers

K-NN is one of the most accepted classifiers used for both general classification problems and in mood based music classification as well. K-NN uses training data directly for the classification of testing data. We can predict label of the testing instance by majority voting on the labels of the nearest instances in the training set.

KNN is an example of a non-parametric classifier. If we denote $D^n = x_1, x_2, \dots, x_n$ a set of n labelled prototypes then the nearest neighbor rule for classifying an unknown vector x is to assign it the label of its closest neighbouring point in the set of labelled prototypes .

It is possible to show that for an unlimited number of prototypes the error rate of this classifier is never worse than twice the optimal Bayes rate. The KNN rule classifies x by assigning the label most regularly represented among the k nearest samples. Normally k is odd to avoid ties in voting. Various methods are used to make algorithms computation faster and storage requirements smaller as this algorithm has heavy time and space requirements

A classifier that is used to detect or obtain the facial landmarks from the face of the user is trained on HELEN dataset. HELEN dataset contains more than 2000 images. These images will be used to train the classifier so that when a completely new and unknown set of images is presented to the classifier, it is able to extract the position of facial landmarks from those images based on the knowledge that it had already acquired from the training set and return the coordinates of the new facial landmarks that it detected. The network is trained with the help of CK extensive data set. This is used to identify the emotion being voiced by the user. Once this has been detected, an appropriate song is selected by the music player that would best match the mood of the user. The overall idea behind making the system is to enhance the experience of the user and ultimately relieve some stress or lighten the mood of the user. The

user does not have to waste any time in searching or to look up for songs and the best track matching the user's mood is detected and played automatically by the music player. The image of the user is captured with the help of a webcam

User	Emotion	Facial Expression	Accuracy
1	Happy	Happy	100
2	Sad	Happy	0
3	Happy	Happy	100
4	Sad	Sad	100

Table 4.7

Emotion Accuracy based inner Emotion

In this Table 4.7.1 the users were given instructions as to what is to be done to perform the prediction of the emotion expressed which provided the following results. Sometimes in cases where the inner emotion is sad and facial expression is happy it resulted in a fail case.

User	Emotion	Facial Expression	Accuracy
1	Happy	Sad	0
2	Sad	Happy	0
3	Sad	Sad	100
4	Happy	Sad	0

Table 4.7.1

Emotion Accuracy Inner emotions matched with Facial expression

In the Table 4.7.1 the users were not given any instructions as to what is to be do and thus the inner emotions or the emotions recognized failed, there were also cases where in the emotion matched with the facial expressions of the user. That is the inner emotions is also detected even the facial expression fails to show.

4.7 Playlist Classifier

Feature extraction is a process to extract important features from data. It includes identifying linguistic data and avoiding any kind of noise. Audio features are classified into 3 categories high-level, mid-level, and low-level audio features.

- High-level features are related to music lyrics like chords, rhythm, melody, etc.
- Mid-level features include beat level attributes, pitch-like fluctuation patterns, and MFCCs.
- Low-level features include energy, a zero-crossing rate which are statistical measures that get extracted from audio during feature extraction.

So to generate these features we use a certain set of steps and are combined under a single name as MFCC that helps extract mid-level and low-level audio features. below are the steps discussed for the working of MFCCs in feature extraction.

1. Audio files are of a certain length(duration) in seconds or as long as in minutes. And the pitch or frequency is continuously changing so to understand this we first divide the audio file into small-small frames which are near about 20 to 40 ms long.
2. After dividing into frames we try to identify and extract different frequencies from each frame. When we divide in such a small frame so assume that one frame divides down in a single frequency.
3. separate linguistic frequencies from the noise
4. To discard any type of noise, take discrete cosine transform (DCT) of the frequencies. students who are from engineering backgrounds might know cosine transform and have studied this in discrete mathematics subjects.

MFCC brings all these for us which we have already imported from the python speech feature library. we will iterate through each category folder, read the audio file, extract the MFCC feature, and dump it in a binary file using the pickle module.

4.8.1 Train-test split the dataset

In Figure 4.8.1 Implementation of a function that accepts a filename and copies all the data in form of a dataframe. After that based on a certain threshold and split the data into train and test sets because to a mix of different genres in both sets. There are different approaches to do train test split so a random module and running a loop till the length of a dataset and generate a random fractional number between 0-1 and if it is less than 66 then a particular row is appended in the train test else in the test set.

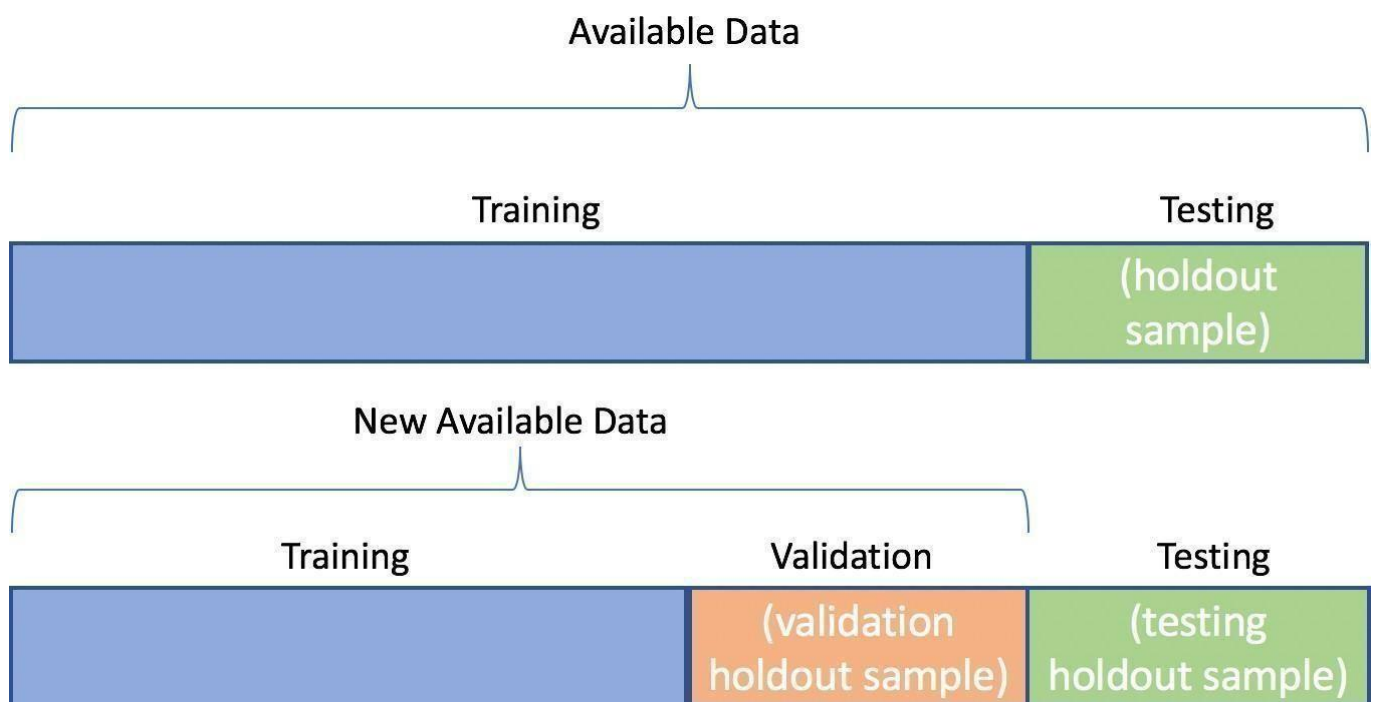


FIGURE 4.8.1

Training and testing Dataset calculation

4.8.2 Calculate the distance between two instance

The data is feeded to KNN algorithms and make all predictions and receive accuracy on the test dataset. The a functional programming approach in a stepwise manner so we only need to call the functions. The first is to get neighbors, extract class and check the accuracy of the model.

4.8.3 Test the Classifier with the new Audio File

The model is implemented and trained the model and There need to be a check for the new data to check how much our model is compliant in predicting the new audio file. So the labels (classes) in numeric form, and to check the class name there implemented a dictionary where the key is numeric label and value is the category name.

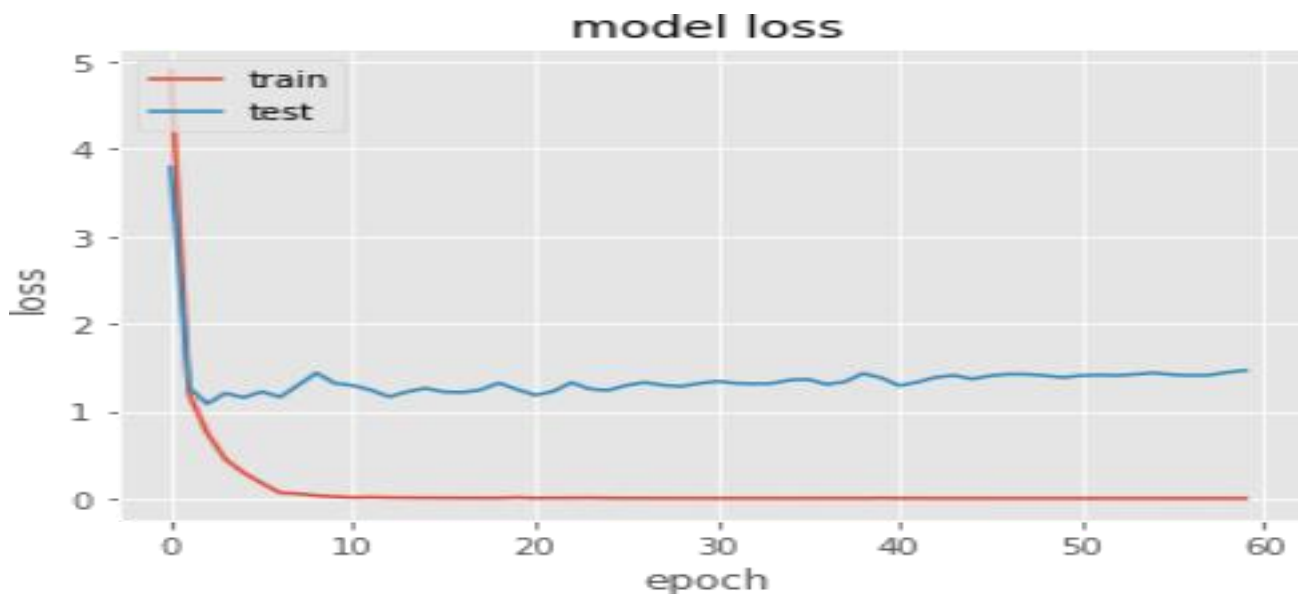


FIGURE 4.8.3

Audio event recognition test Data

In the Figure 4.8.3 Audio event recognition and sound classification is a difficult machine learning task. One of the most challenging aspects is finding a dataset for labels you want to classify. With the limited amount of data, we decided to broaden to datasets other than strictly animal samples. After finding some datasets, the following hurdles must be overcome:

- files obscured with background noise (either from other audio events or transducer noise)
- limited number of samples per label
- unbalanced data sets
- inconsistent data: varying loudness and time durations
- incorrect or unverified labels

4.8 Play the Selected Song in the Media Player

There are plenty of ways to play audio files with Python. It really depends on your application,

but the easiest way, by far, is to use the bindings for VLC to control VLC with Python, and play your files.

With VLC, you don't need to worry about codecs and file support. It also doesn't require too many complicated methods, and/or objects. So, for simple audio playback, VLC is best.

4.9.1 Play A Song

Playing a file from an existing object is even easier. You only need to call the **play** method on the object, and Python will begin playing it. When the playback finishes, it will stop.

4.9.2 Stopping And Pause

The VLC bindings make it easy to stop or pause a file .There is a pause method that will pause playback if the file is playing. If the player is already paused, calling the method again will resume playback.

4.9.3 Looping And “Playlists”

create pseudo-playlists with this, and loop through the songs that you've added. It would only take a basic for loop. In Figure the vlc media player is opened through the python package and the song played.



FIGURE 4.9.3

VLC MEDIA PLAYER IS OPENED AND SONG PLAYED

CHAPTER 5

CONCLUSION AND FUTURE ENHANCEMENT

5.1. CONCLUSION

The Emotion Detected By the (SVM) Support vector Machine Algorithm proved to be the best algorithm to analyze the datasets that are trained. The computational time is reduced by the best optimized packages which optimize the data that is trained with accurate metrics. The developed System is efficient to detect the emotion and accuracy is prominent to select the song as well as the playlist based on the K-Means Algorithm which is fed for the trained datasets provides an efficiency in accuracy under the playlist gathering. The Integral Image Calculation provided an efficient and optimization in computation Time. The Goal is achieved by reducing the computation time by using the python packages under tensorflow and the Convolution Architecture to reduce the raw data that gets generated in the data collection phase this makes the application to run in a stable mode.

5. 2. FUTURE ENHANCEMENT

In future Music Player can be enhanced with Google play music, so songs which are not present in local storage can also be played and to access the whole application in speech based. The Emotion Based Music System will be of great advantage to users looking for music based on their mood and emotional behavior. It will help reduce the searching time for music thereby reducing unnecessary time and hence increasing the overall accuracy and efficiency of the system.

APPENDIX 1

Import Libraries:

```
In [4]: import pandas as pd
import numpy as np
import json
import re
import sys
import itertools
import cv2
from PIL import Image

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.preprocessing import MinMaxScaler
from skimage import io
import matplotlib.pyplot as plt

import spotipy
from spotipy.oauth2 import SpotifyClientCredentials
from spotipy.oauth2 import SpotifyOAuth
import spotipy.util as util

import warnings
warnings.filterwarnings("ignore")
```

```
In [2]: %matplotlib inline
```

Load the dataset:

```
In [6]: spotify_df = pd.read_csv('kaggleSPOTIFY.csv')
```

```
In [7]: spotify_df.dropna(subset=['consolidates_genre_lists'], inplace=True)
```

Reduction of missing values:

```
In [8]: spotify_df.isna().sum()
```

```
Out[8]: Unnamed: 0      0
valence      0
year      0
acousticness  0
artists      0
danceability  0
duration_ms  0
energy      0
explicit     0
id           0
instrumentalness  0
key          0
liveness     0
loudness     0
mode         0
name         0
popularity   0
release_date  0
speechiness  0
tempo        0
artists_upd_v1  0
artists_upd_v2  0
artists_upd    0
artists_song   0
consolidates_genre_lists  0
dtype: int64
```

```
In [9]: spotify_df.head()
```

Data Training:

```
In [17]: def base_model():
        model = Sequential()
        model.add(Dense(8,input_dim=10,activation='relu'))
        model.add(Dense(4,activation='softmax'))
        model.compile(loss='categorical_crossentropy',optimizer='adam',
                      metrics=['accuracy'])
        return model
```

```
In [18]: #Configure the model
        estimator = KerasClassifier(build_fn=base_model,epochs=300,batch_size=200,verbose=0)
```

```
In [19]: #Evaluate the model using KFold cross validation
        kfold = KFold(n_splits=10,shuffle=True)
        results = cross_val_score(estimator,X,encoded_y,cv=kfold)
        print("Baseline: %.2f%% (%.2f%%)" % (results.mean()*100,results.std()*100))
```

WARNING:tensorflow:5 out of the last 5 calls to <function Model.make_test_function.<locals>.test_function at 0x00000248775F3160> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

WARNING:tensorflow:6 out of the last 6 calls to <function Model.make_test_function.<locals>.test_function at 0x000002486F5F0820> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has experimental_relax_shapes=True option that relaxes argument shapes that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

Baseline: 79.74% (3.85%)

```
In [24]: estimator.fit(X_train,y_train)
        y_preds = estimator.predict()
```


Pipeline:

The purpose of the pipeline is to assemble several steps that can be cross-validated together while setting different parameters.

```
In [34]: estimator.fit(X_train, y_train)
pip.fit(X2, encoded_y)

In Out[36]: Pipeline(steps=[('minmaxscaler', MinMaxScaler()),
                             ('keras',
                              <tensorflow.python.keras.wrappers.scikit_learn.KerasClassifier object at 0x0000024B732B80D0>)])

In In [40]: def predict_mood(preds):
#         pipe = Pipeline([('minmaxscaler', MinMaxScaler()), ('keras', KerasClassifier(build_fn=base_model, epochs=300, batch_size=200, va
#         # Fit the Pipeline
#         pipe.fit(X2, encoded_y)

#         preds_features = np.array(preds[:]).reshape(-1,1).T

# Predict the features of the song
results = pipe.predict(preds_features)

mood = np.array(target['mood'][target['encode']==int(results)])

return str(mood[0])
#print(f"{name_song} by {artist} is a {mood[0].upper()} song")

In [44]: res = []

for i in range(len(mood_trans_np)):
    res.append(predict_mood(mood_trans_np[i]))

In [45]: spotify_df.shape

Out[45]: (156410, 25)
```

Training and Testing of Mood Detection:

```
In [ ]: from google.colab import drive
drive.mount('/content/drive')
```

```
In [ ]: import pandas as pd
import numpy as np
data = pd.read_csv('/content/drive/My Drive/projects/train.csv')
test_data = pd.read_csv('/content/drive/My Drive/projects/test.csv')
```

```
In [ ]: def stringToList(input):
    lst = []
    for i in input:
        l = list(map(float, i.split(" ")))
        lst.append(l)
    return lst
```

```
In [ ]: x_train = data['pixels']
y_train = np.array(data['emotion'])

x_test = np.array(test_data['pixels'])

x_train = np.array(stringToList(x_train))/255.0
x_train = np.reshape(x_train, (28709, 48,48,1))
x_test = np.array(stringToList(x_test))
x_test = np.reshape(x_test, (7178,48,48,1)) /255.0
#y_train = np.reshape(y_train, (1,y_train.shape[0]))

# x_train = np.array(map(float, data['pixels']))
# x_train
```

Processing the data:

```
return lst
```

```
In [ ]: x_train = data['pixels']
y_train = np.array(data['emotion'])

x_test = np.array(test_data['pixels'])

x_train = np.array(stringToList(x_train))/255.0
x_train = np.reshape(x_train, (28709, 48,48,1))
x_test = np.array(stringToList(x_test))
x_test = np.reshape(x_test, (7178,48,48,1)) /255.0
#y_train = np.reshape(y_train, (1,y_train.shape[0]))

# x_train = np.array(map(float, data['pixels']))
# x_train
# print(x_train.shape)
# type(x_train[0][0])
# #x_train /= np.float(255)
# y_train = data['emotion']
```

```
In [ ]: print(x_train.shape)

(28709, 48, 48, 1)
```

```

In [ ]: model = Sequential()
        #1st convo
        model.add(Conv2D(96, (3,3), input_shape = (48,48,1)))
        model.add(Activation('relu'))
        #pooling
        model.add(BatchNormalization())
        model.add(MaxPooling2D(pool_size=(2,2)))

        #2nd convo
        model.add(Conv2D(256, (3,3)))
        model.add(Activation('relu'))
        #pooling
        model.add(BatchNormalization())
        model.add(MaxPooling2D(pool_size=(2,2)))

        #3rd convo
        model.add(Conv2D(384, (3,3)))
        model.add(Activation('relu'))
        #pooling
        model.add(BatchNormalization())
        model.add(MaxPooling2D(pool_size=(2,2)))

        #4th convo
        model.add(Conv2D(256, (3,3)))
        model.add(Activation('relu'))
        #pooling
        model.add(BatchNormalization())
        model.add(MaxPooling2D(pool_size=(2,2)))

```

```

        #passing through dense layer
        model.add(Flatten())

        #1st dense layer
        model.add(Dense(1024))
        model.add(Activation('relu'))
        #dropout
        model.add(BatchNormalization())
        model.add(Dropout(0.4))

        #2nd dense layer
        model.add(Dense(1024))
        model.add(Activation('relu'))
        #dropout
        model.add(BatchNormalization())
        model.add(Dropout(0.4))

        #3rd dense layer
        model.add(Dense(256))
        model.add(Activation('relu'))
        #dropout
        model.add(Dropout(0.4))
        model.add(BatchNormalization())
        model.add(Activation('relu'))

        #output layer
        model.add(Dense(7))
        model.add(Activation('softmax'))

```

```
#output layer
model.add(Dense(7))
model.add(Activation('softmax'))

model.compile(loss="sparse_categorical_crossentropy",
              optimizer = "adam",
              metrics = ['sparse_categorical_accuracy'])
model.fit(x_train, y_train, batch_size=32, epochs=10, validation_split=0.1)
```

```
Epoch 1/10
808/808 [=====] - 620s 766ms/step - loss: 2.0099 - sparse_categorical_accuracy: 0.2363 - val_loss: 1.5
107 - val_sparse_categorical_accuracy: 0.3932
Epoch 2/10
808/808 [=====] - 618s 765ms/step - loss: 1.4504 - sparse_categorical_accuracy: 0.4393 - val_loss: 1.3
908 - val_sparse_categorical_accuracy: 0.4538
Epoch 3/10
808/808 [=====] - 619s 766ms/step - loss: 1.2398 - sparse_categorical_accuracy: 0.5286 - val_loss: 1.4
242 - val_sparse_categorical_accuracy: 0.4580
Epoch 4/10
808/808 [=====] - 633s 784ms/step - loss: 1.1549 - sparse_categorical_accuracy: 0.5696 - val_loss: 1.6
212 - val_sparse_categorical_accuracy: 0.4800
Epoch 5/10
808/808 [=====] - 639s 791ms/step - loss: 1.0613 - sparse_categorical_accuracy: 0.6037 - val_loss: 1.3
038 - val_sparse_categorical_accuracy: 0.4897
Epoch 6/10
808/808 [=====] - 640s 793ms/step - loss: 0.9516 - sparse_categorical_accuracy: 0.6517 - val_loss: 1.4
444 - val_sparse_categorical_accuracy: 0.4803
Epoch 7/10
808/808 [=====] - 639s 791ms/step - loss: 0.8370 - sparse_categorical_accuracy: 0.6965 - val_loss: 1.1
014 - val_sparse_categorical_accuracy: 0.5904
Epoch 8/10
808/808 [=====] - 621s 769ms/step - loss: 0.7241 - sparse_categorical_accuracy: 0.7405 - val_loss: 1.1
654 - val_sparse_categorical_accuracy: 0.5556
```

```
In [ ]: from keras.models import load_model
loaded_model = load_model('/content/drive/My Drive/projects/model.h5')
```

```
In [ ]: loaded_model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 46, 46, 96)	960
activation (Activation)	(None, 46, 46, 96)	0
batch_normalization (Batch Normalization)	(None, 46, 46, 96)	384
max_pooling2d (MaxPooling2D)	(None, 23, 23, 96)	0
conv2d_1 (Conv2D)	(None, 21, 21, 256)	221440
activation_1 (Activation)	(None, 21, 21, 256)	0
batch_normalization_1 (Batch Normalization)	(None, 21, 21, 256)	1024
max_pooling2d_1 (MaxPooling2D)	(None, 10, 10, 256)	0
conv2d_2 (Conv2D)	(None, 8, 8, 384)	885120
activation_2 (Activation)	(None, 8, 8, 384)	0
batch_normalization_2 (Batch Normalization)	(None, 8, 8, 384)	1536
max_pooling2d_2 (MaxPooling2D)	(None, 4, 4, 384)	0
conv2d_3 (Conv2D)	(None, 2, 2, 256)	884992

Converting pixel to image:

```
In [ ]: import matplotlib.pyplot as plt

single_image = np.array(x_test[69]*255)
print(single_image.shape)
single_image = np.reshape(single_image,(48,48),order = 'C')
print(single_image.shape)
plt.imshow(single_image, cmap='gray', vmin=0, vmax=255)
# s_img = np.array(single_image, shape=(48,48))
```

```
(48, 48, 1)
(48, 48)
```

```
Out[58]: <matplotlib.image.AxesImage at 0x7f4a3a91e4a8>
```



Mood Detection from videos and images

```
In [9]: def stringToList(input):
        lst = []
        for i in input:
            l = list(map(float, i.split(" ")))
            lst.append(l)
        return lst
```

```
In [8]: def moodNamePrintFromLabel(n):
        if n == 0: result = 'Angry'
        elif n == 1: result = 'Disgust'
        elif n == 2: result = 'Fear'
        elif n == 3: result = 'Happy'
        elif n == 4: result = 'Sad'
        elif n == 5: result = 'Surprise'
        elif n == 6: result = 'Neutral'
        return result
```

```
In [6]: from google.colab import drive
        drive.mount('/content/drive')
```

Mounted at /content/drive

```
In [7]: import pandas as pd
        import numpy as np
```

```
In [10]: data = pd.read_csv('/content/drive/My Drive/projects/train.csv')
        test_data = pd.read_csv('/content/drive/My Drive/projects/test.csv')
```

Processing the data:

```
In [28]: X = data['pixels']
Y = np.array(data['emotion'])

x_test_data = np.array(test_data['pixels'])

X = np.array(stringToList(X))/255.0
X = np.reshape(X, (28709, 48,48,1))

x_test_data = np.array(stringToList(x_test_data))
x_test_data = np.reshape(x_test_data, (7178,48,48,1)) /255.0

In [ ]: print(X.shape)
print(Y.shape)

(28709, 48, 48, 1)
(28709,)
```

Data splitting to test and training set:

```
In [ ]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size = 0.2)

In [ ]: print(x_train.shape)
print(y_train.shape)

(22967, 48, 48, 1)
(22967,)
```

```
In [ ]: import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation, Flatten, Conv2D, Dropout, MaxPooling2D, BatchNormalization

In [ ]: model = Sequential()
#1st convo
model.add(Conv2D(96, (3,3), input_shape = (48,48,1)))
model.add(Activation('relu'))
#pooling
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(BatchNormalization())

#2nd convo
model.add(Conv2D(256, (3,3)))
model.add(Activation('relu'))
```



```

In [ ]: model = Sequential()
        #1st convo
        model.add(Conv2D(96, (3,3), input_shape = (48,48,1)))
        model.add(Activation('relu'))
        #polling
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(BatchNormalization())

        #2nd convo
        model.add(Conv2D(256, (3,3)))
        model.add(Activation('relu'))
        #polling
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(BatchNormalization())

        #3rd convo
        model.add(Conv2D(384, (3,3)))
        model.add(Activation('relu'))
        #polling
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(BatchNormalization())

        #4th convo
        model.add(Conv2D(256, (3,3)))
        model.add(Activation('relu'))
        #polling
        model.add(MaxPooling2D(pool_size=(2,2)))
        model.add(BatchNormalization())

        #passing through dense layer
        model.add(Flatten())

        #passing through dense layer
        model.add(Flatten())

        #1st dense layer
        model.add(Dense(1024))
        model.add(Activation('relu'))
        #dropout
        model.add(BatchNormalization())
        model.add(Dropout(0.4))

        #2nd dense layer
        model.add(Dense(1024))
        model.add(Activation('relu'))
        #dropout
        model.add(BatchNormalization())
        model.add(Dropout(0.4))

        #3rd dense layer
        model.add(Dense(256))
        model.add(Activation('relu'))
        #dropout
        model.add(Dropout(0.4))
        model.add(BatchNormalization())
        model.add(Activation('relu'))

        #output layer
        model.add(Dense(7))
        model.add(Activation('softmax'))

        model.compile(loss="sparse_categorical_crossentropy",
                      optimizer = "adam",
                      metrics = ['sparse_categorical_accuracy'])
        model.fit(x_train, y_train, batch_size=32, epochs=10, validation_split=0.1)

```

Loading the saved mood detection model from google drive:

```
In [11]: from keras.models import load_model
loaded_model = load_model('/content/drive/My Drive/projects/model.h5')

In [ ]: score = loaded_model.evaluate(x_test, y_test, verbose=1)
180/180 [=====] - 37s 203ms/step - loss: 0.6828 - sparse_categorical_accuracy: 0.7654
```

Taking input as image and videos and detecting faces and getting the face portion. Then passing these faces to model and getting the mood detection output.

```
In [112]: import cv2
from PIL import Image
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow

def videoToMoodDetection(video_path):
    #video_cap = cv2.VideoCapture(0)
    video_cap = cv2.VideoCapture(video_path)

    while (video_cap.isOpened()):
        ret, frame = video_cap.read()

        Gray_img = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

        faces = face_cascade.detectMultiScale(Gray_img, 1.3, 5)

        for (x,y,w,h) in faces:
            pxl_lst = []
```

```
            #resize for passing to the model
            resized_img = np.reshape(resized_img, (1,48,48,1))/255.0

            #passing to model
            result = np.argmax(loaded_model.predict(resized_img), axis=-1)
            if result is not None:
                print(moodNamePrintFromLabel(result))

def imageToMoodDetection(img_path):
    img = cv2.imread(img_path)

    Gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(Gray_img, 1.3, 3)

    for (x,y,w,h) in faces:
        pxl_lst = []
        for i in range(y, y+h):
            lst = []
            for j in range(x, x+w):
                lst.append(Gray_img[i][j])
            pxl_lst.append(lst)
        single_face = np.array(pxl_lst)
        single_face = np.reshape(single_face, (h,w,))

        resized_img = cv2.resize(single_face, (48,48), interpolation = cv2.INTER_AREA)
        cv2_imshow(resized_img)
        print(resized_img.shape)
        resized_img = np.reshape(resized_img, (1,48,48,1))/255.0

        #passing to model
        result = np.argmax(loaded_model.predict(resized_img), axis=-1)
        if result is not None:
```




```


if result is not None:
    print(moodNamePrintFromLabel(result))

In [110]: imageToMoodDetection('/content/DSC_2693.NEF.jpg')

[[43 34 29 ... 26 31 37]
 [33 29 27 ... 24 29 35]
 [24 25 27 ... 22 25 30]
 ...
 [50 50 51 ... 46 45 47]
 [50 51 53 ... 50 49 49]
 [53 51 51 ... 47 48 49]]



(48, 48)
Happy
[[ 68 45 39 ... 38 38 38]
 [ 65 44 40 ... 38 38 38]
 [ 60 46 43 ... 38 38 38]
 ...
 [192 198 199 ... 156 148 150]
 [182 188 184 ... 150 145 146]
 [162 161 169 ... 152 147 144]]



(48, 48)
Sad
[[ 29 33 36 ... 127 144 178]
 [ 27 33 32 ... 138 143 174]
 [ 29 29 28 ... 144 138 153]]

```

```

In [115]: videoToMoodDetection('/content/video.mp4')


Neutral

Neutral

Neutral

Happy


```

Converting Pixel to image:

```
In [ ]: import matplotlib.pyplot as plt

single_image = np.array(x_test_data[13]*255)
print(single_image.shape)
single_image = np.reshape(single_image,(48,48),order = 'C')
print(single_image.shape)
plt.imshow(single_image, cmap='gray', vmin=0, vmax=255)
# s_img = np.array(single_image, shape=(48,48))

(48, 48, 1)
(48, 48)
```

Out[31]: <matplotlib.image.AxesImage at 0x7f6b2ba65e48>



Music Recommender

```
In [3]: #another useful command to make data exploration easier
# NOTE: if you are using a massive dataset, this could slow down your code.
pd.set_option('display.max_columns', None)
pd.set_option("max_rows", None)
```

```
In [4]: df = pd.read_csv('kaggleMusicMoodFinal.csv')
```

```
In [5]: spotify_df = df.copy()
```

```
In [6]: spotify_df.shape
```

Out[6]: (156410, 27)

```
In [7]: float_cols = spotify_df.dtypes[spotify_df.dtypes == 'float64'].index.values
```

```
In [8]: ohe_cols = 'popularity'
```

```
In [9]: # create 5 point buckets for popularity
spotify_df['bucket_popularity'] = spotify_df['popularity'].apply(lambda x: int(x/5))
```

```
In [10]: spotify_df['consolidates_genre_lists_upd'] = spotify_df['consolidates_genre_lists'].apply(lambda x: [re.sub(' ', '_', i) for i in x])
```

```
In [11]: spotify_df.head()
```

Out[11]:

	Unnamed: 0	Unnamed: 0.1	valence	year	acousticness	artists	danceability	duration_ms	energy	explicit	id	Instrumentalness	key	l
0	0	0	0.177	1989	0.568	조성모	0.447	237688	0.2150	0	2ahebdwe2cNXT4eL34T7oW	0.000001	10	

```

In [12]: #simple function to create OHE features
#this gets passed later on
def One_Hot_Encode_Prep(df, column, new_name):
    """
    Create One Hot Encoded features of a specific column

    Parameters:
        df (pandas dataframe): Spotify Dataframe
        column (str): Column to be processed
        new_name (str): new column name to be used

    Returns:
        tf_df: One hot encoded features
    """

    tf_df = pd.get_dummies(df[column])
    feature_names = tf_df.columns
    tf_df.columns = [new_name + "|" + str(i) for i in feature_names]
    tf_df.reset_index(drop = True, inplace = True)
    return tf_df

#function to build entire feature set
def Feature_Set_Using_Tfidf(df, float_cols):
    """
    Process spotify df to create a final set of features that will be used to generate recommendations

    Parameters:
        df (pandas dataframe): Spotify Dataframe
        float_cols (list(str)): List of float columns that will be scaled

    Returns:
        final: final set of features
    """

```

```

#tfidf genre lists
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(df['consolidates_genre_lists_upd'].apply(lambda x: " ".join(x)))
genre_df = pd.DataFrame(tfidf_matrix.toarray())
genre_df.columns = ['genre' + "|" + i for i in tfidf.get_feature_names()]
genre_df.reset_index(drop = True, inplace=True)

#explicit_oh = One_Hot_Encode_Prep(df, 'explicit', 'exp')
year_oh = One_Hot_Encode_Prep(df, 'year', 'year') * 0.5
popularity_oh = One_Hot_Encode_Prep(df, 'bucket_popularity', 'pop') * 0.15

#scale float columns
floats = df[float_cols].reset_index(drop = True)
scaler = MinMaxScaler()
floats_scaled = pd.DataFrame(scaler.fit_transform(floats), columns = floats.columns) * 0.2

#concatenate all features
final = pd.concat([genre_df, floats_scaled, popularity_oh, year_oh], axis = 1)

#add song id
final['id'] = df['id'].values

return final

```

```

def Get_Spotify_Playlist_DataFrame(playlist_name,id_dic, df):
    """
    Pull songs from a specific playlist.

    Parameters:
        playlist_name (str): name of the playlist you'd like to pull from the spotify API
        id_dic (dic): dictionary that maps playlist_name to playlist_id
        df (pandas dataframe): spotify dataframe

    Returns:
        playlist: all songs in the playlist THAT ARE AVAILABLE IN THE KAGGLE DATASET
    """

    #generate playlist dataframe
    playlist = pd.DataFrame()
    playlist_name = playlist_name

    #print(len(sp.playlist(id_dic[playlist_name]))['tracks']['items']))

    for ix, i in enumerate(sp.playlist(id_dic[playlist_name]))['tracks']['items']:
        #print(i['track']['artists'][0]['name'])
        playlist.loc[ix, 'artist'] = i['track']['artists'][0]['name']
        playlist.loc[ix, 'name'] = i['track']['name']
        playlist.loc[ix, 'id'] = i['track']['id'] # i['uri'].split(':')[2]
        playlist.loc[ix, 'url'] = i['track']['album']['images'][1]['url']
        playlist.loc[ix, 'date_added'] = i['added_at']

    playlist['date_added'] = pd.to_datetime(playlist['date_added'])

    #playlist = playlist[playlist['id'].isin(df['id'].values)].sort_values('date_added',ascending = False)

    return playlist

```

```

def visualize_songs(df):
    """
    Visualize cover art of the songs in the inputted dataframe

    Parameters:
        df (pandas dataframe): Playlist Dataframe
    """

    temp = df['url'].values
    plt.figure(figsize=(15,int(0.625 * len(temp))))
    columns = 5

    for i, url in enumerate(temp):
        plt.subplot(len(temp) / columns + 1, columns, i + 1)

        image = io.imread(url)
        plt.imshow(image)
        plt.xticks(color = 'w', fontsize = 0.1)
        plt.yticks(color = 'w', fontsize = 0.1)
        plt.xlabel(df['name'].values[i], fontsize = 12)
        plt.tight_layout(h_pad=0.4, w_pad=0)
        plt.subplots_adjust(wspace=None, hspace=None)

    plt.show()

```

```

def generate_playlist_feature(complete_feature_set, playlist_df, weight_factor):
    """
    Summarize a user's playlist into a single vector

    Parameters:
        complete_feature_set (pandas dataframe): Dataframe which includes all of the features for the spotify songs
        playlist_df (pandas dataframe): playlist dataframe
        weight_factor (float): float value that represents the recency bias. The larger the recency bias, the most priority recency

    Returns:
        playlist_feature_set_weighted_final (pandas series): single feature that summarizes the playlist
        playlist_features_not_in_dataframe (pandas dataframe):
    """

    playlist_features_in_dataframe = complete_feature_set[complete_feature_set['id'].isin(playlist_df['id'].values)].drop('id',
    playlist_features_in_dataframe = playlist_features_in_dataframe.merge(playlist_df[['id', 'date_added']], on = 'id', how = 'inner')
    playlist_features_not_in_dataframe = complete_feature_set[~complete_feature_set['id'].isin(playlist_df['id'].values)].drop('id',
    playlist_feature_set = playlist_features_in_dataframe.sort_values('date_added', ascending = False)
    most_recent_date = playlist_feature_set.iloc[0, -1]

    for ix, row in playlist_feature_set.iterrows():
        playlist_feature_set.loc[ix, 'months_from_recent'] = int((most_recent_date.to_pydatetime() - row.iloc[-1].to_pydatetime()).days / 30)

    playlist_feature_set['weight'] = playlist_feature_set['months_from_recent'].apply(lambda x: weight_factor ** (-x))

    playlist_feature_set_weighted = playlist_feature_set.copy()
    #print(playlist_feature_set_weighted.iloc[:, :-4].columns)
    playlist_feature_set_weighted.update(playlist_feature_set_weighted.iloc[:, :-4].mul(playlist_feature_set_weighted.weight, 0))
    playlist_feature_set_weighted_final = playlist_feature_set_weighted.iloc[:, :-4]
    #playlist_feature_set_weighted_final['id'] = playlist_feature_set['id']

    return playlist_feature_set_weighted_final.sum(axis = 0), playlist_features_not_in_dataframe

```

```

def Recommend_Playlist(df, features, nonplaylist_features):
    """
    Pull songs from a specific playlist.

    Parameters:
        df (pandas dataframe): spotify dataframe
        features (pandas series): summarized playlist feature
        nonplaylist_features (pandas dataframe): feature set of songs that are not in the selected playlist

    Returns:
        non_playlist_df_top_40: Top 40 recommendations for that playlist
    """

    non_playlist_df = df[df['id'].isin(nonplaylist_features['id'].values)]
    non_playlist_df['sim'] = cosine_similarity(nonplaylist_features.drop('id', axis = 1).values, features.values.reshape(1, -1))
    non_playlist_df_top_40 = non_playlist_df.sort_values('sim', ascending = False).head(40)
    non_playlist_df_top_40['url'] = non_playlist_df_top_40['id'].apply(lambda x: sp.track(x)['album']['images'][1]['url'])

    return non_playlist_df_top_40

```

```

#client id and secret for my application
client_id = ''
client_secret = ''

```



```

#client id and secret for my application
client_id = ''
client_secret = ''

In [14]: scope = 'user-library-read'

if len(sys.argv) > 1:
    username = sys.argv[1]
else:
    print("Usage: %s username" % (sys.argv[0],))
    sys.exit()

In [15]: auth_manager = SpotifyClientCredentials(client_id=client_id, client_secret=client_secret)
sp = spotipy.Spotify(auth_manager=auth_manager)

In [16]: token = util.prompt_for_user_token(scope, client_id=client_id, client_secret=client_secret, redirect_uri='http://localhost:8400/')

In [17]: sp = spotipy.Spotify(auth=token)

In [18]: #gather playlist names and images.
#images aren't going to be used until I start building a UI
id_name = {}
list_photo = {}
for i in sp.current_user_playlists()['items']:

    id_name[i['name']] = i['uri'].split(':')[2]
    list_photo[i['uri'].split(':')[2]] = i['images'][0]['url']

In [19]: id_name
Out[19]: {'Mixed': '2UL11PFnuIW69TsyTwDwt', 'EDM': '43KYp7yRMBTMIPFP358dqH'}

```

```

In [19]: id_name
Out[19]: {'Mixed': '2UL11PFnuIW69TsyTwDwt', 'EDM': '43KYp7yRMBTMIPFP358dqH'}

In [20]: def ChooseDataset(x):
    if x == "Disgust":
        return spotify_df[spotify_df['Mood'].isin(['Energetic', 'Happy', 'Calm'])]
    if x == "Angry":
        return spotify_df[spotify_df['Mood'].isin(['Energetic', 'Calm'])]
    if x == "Fear":
        return spotify_df[spotify_df['Mood'].isin(['Happy', 'Calm'])]
    if x == "Happy":
        return spotify_df[spotify_df['Mood'].isin(['Sad', 'Happy', 'Calm'])]
    if x == "Sad":
        return spotify_df[spotify_df['Mood'].isin(['Sad', 'Happy', 'Calm'])]
    if x == "Surprise":
        return spotify_df[spotify_df['Mood'].isin(['Energetic', 'Happy', 'Sad'])]
    return spotify_df

In [21]: def Recommend_Top40(x):
    #.....
    #.....
    #.....

    O_df = ChooseDataset(x)

    # Feature Engineering from main dataframe
    complete_feature_set = Feature_Set_Using_TDIDF(O_df, float_cols=float_cols)#.mean(axis = 0)

    # collecting spotify user playlist dataframe
    one_playlist_from_spotify = Get_Spotify_Playlist_DataFrame('EDM', id_name, O_df)

    # Linear vector for recommendation
    features_in_the_playlist, features_not_in_the_playlist = generate_playlist_feature(complete_feature_set, one_playlist_from_sp

```

```
In [5]: def moodNamePrintFromLabel(n):
        if n == 0: result = "Angry"
        elif n == 1: result = "Disgust"
        elif n == 2: result = "Fear"
        elif n == 3: result = "Happy"
        elif n == 4: result = "Sad"
        elif n == 5: result = "Surprise"
        elif n == 6: result = "Neutral"

        return result

def imageToMoodDetection(img_path):

    img = cv2.imread(img_path)

    Gray_img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    faces = face_cascade.detectMultiScale(Gray_img, 1.3, 3)

    for (x,y,w,h) in faces:
        pxl_lst = []
        for i in range(y, y+h):
            lst = []
            for j in range(x, x+w):
                lst.append(Gray_img[i][j])
            pxl_lst.append(lst)
        single_face = np.array(pxl_lst)
        single_face = np.reshape(single_face,(h,w,))

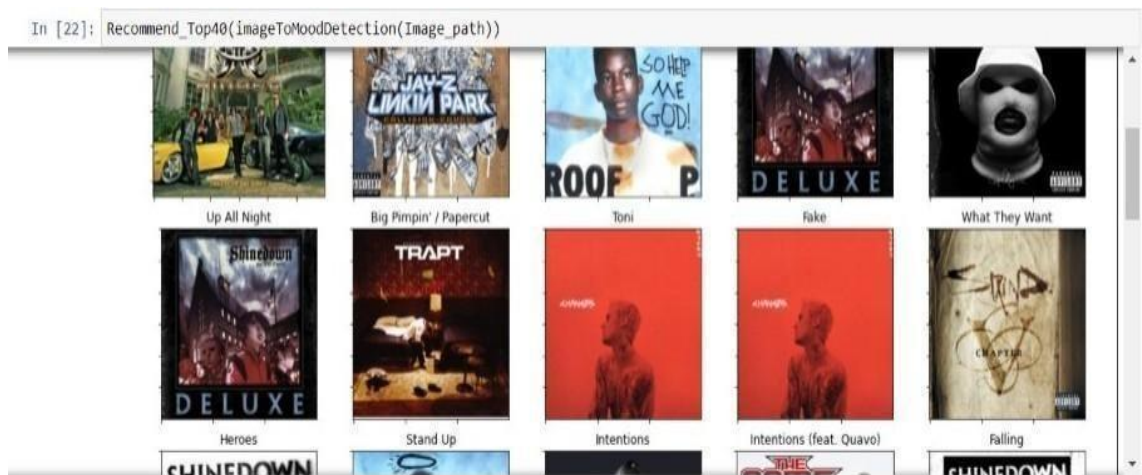
        resized_img = cv2.resize(single_face, (48,48), interpolation = cv2.INTER_AREA)

        resized_img = np.reshape(resized_img,(1,48,48,1))/255.0

        #passing to model
        result = np.argmax(loader_model.predict(resized_img), axis=-1)
```

APPENDIX 2

RECOMMENDED SONGS FROM SPOTIFY EMD PLAYLIST FOR NEUTRAL MOOD



REFERENCES

- 1) Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on, vol. 1, pp. 511-518 IEEE, 2001 (2001)
- 2) H. Immanuel James, J. James Anto Arnold, J. Maria Masilla Ruban, M. Tamilarasan, R. Saranya” EMOTION BASED MUSIC RECOMMENDATION SYSTEM”: pISSN: 2395-0072 , IRJET 2019
- 3) Hafeez Kabani, Sharik Khan , Omar Khan , Shabana Tadvi”Emotion Based Music Player” International Journal of Engineering Research and General Science Volume 3, Issue 1, January-February , 2015
- 4) Shlok Gikla, Husain Zafar, Chuntan Soni, Kshitija Waghurdekar”SMART MUSIC INTEGRATING AND MUSIC MOOD RECOMMENDATION”2017 International Conference on Wireless Communications, Signal Processing and Networking(WiSpPNET)
- 5) T.-H. Wang and J.-J.J. Lien, “Facial Expression Recognition System Based on Rigid and Non-Rigid Motion Separation and 3D Pose Estimation” J. Pattern Recognition , vol. 42, no. 5,pp. 962-977, 2009
- 6) Srushti Sawant, Shraddha Patil, Shubhangi Biradar,”EMOTION BASED MUSIC SYSTEM”, International Journal of Innovations & Advancement in Computer Science,IJIACS ISSN 2347-8616 volue 7, Issue 3 March 2018
- 7) Sudha Veluswamy, Hariprasad Kanna, Balasubramanian Anand, Anshul Sharma “METHOD AND APPARATUS FOR RECOGNIZING AN EMOTION OF AN

INDIVIDUAL BASED ON FACIAL ACTION UNITS”US2012/0101735A1

8) Markus Mans Folke Andreasson”GENERATING MUSIC PLAYLIST BASED ON FACIAL EXPRESSION”US8094891B2

9) Mutasem K. Alsmadi”FACIAL EXPRESSION RECOGNITION”US2018/0211102A1

10) Vaid S, Singh P and Kaur C 2015 EEG signal analysis for BCI interface: A review In2015 fifth international conference on advanced computing & communication technologies 143- 147 IEEE

11) Liu C, Xie S, Xie X, Duan X, Wang W and Obermayer K 2018 Design of a video feedback SSVEP-BCI system for car control based on improved MUSIC method In2018 6th International Conference on Brain-Computer Interface (BCI) 1-4 IEEE