EMOTION BASED MUSIC RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

Submitted by

MEGHAA RHENITH R

PADMAPRIYA G

SERENA J E

SNEHA B

in partial fulfilment for the award of the degree of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

LOYOLA-ICAM
COLLEGE OF ENGINEERING AND TECHNOLOGY

CHENNAI – 600034

ANNA UNIVERSITY: 600025

APRIL 2022

ANNA UNIVERSITY CHENNAI: 600025

BONAFIDE CERTIFICATE

Certified that this project report **"Emotion Based Music Recommendation System"** is the bonafide work of **MEGHAA RHENITH R (311119104044), PADMAPRIYA G(311119104048), SERENA J E (311119104055)** and **SNEHA B(311119104057)** who carried out the project work under my supervision.

SIGNATURE

HEAD OF THE DEPARTMENT
Department of Computer Science and Engineering
Loyola-ICAM College of Engineering and Technology
Loyola Campus,
Nungambakkam,
Chennai-600034

SIGNATURE

SUPERVISOR
Assistant Professor
Department of Computer Science and Engineering
LICET
Loyola Campus,
Nungambakkam,
Chennai-600034

Submitted for the project viva voce held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Firstly, we are deeply obliged to the almighty for this opportunity and for leading us to the successful completion of the project. The experience has been very fruitful.

We express our sincere gratitude to the beloved Director **Rev Dr S Maria Wenisch SJ**, the Dean of Students **Dr D Caleb Chanthi Raj** and the Dean of Engineering Education **Mr Nicolas Juhel**, for their continuous support and Encouragement.

We are deeply obliged to the Principal **Dr L Antony Michael Raj** for his inspiring guidance and invaluable advice during the project work. We also express our sincere thanks to the Head of the Department and Mini Project Coordinator **Ms. Sathia Priya R**, the teaching and non-teaching faculty of our department for their advice, support and guidance.

We extend our gratitude to our project guide **Dr G R Jainish** for providing valuable insights and resources. Without her supervision and constant help, this project would not have been possible.

Last but not the least, we are extremely grateful to our family and friends, who have been a constant source of support during the preparation of this project work.

ABSTRACT

Songs, as a medium of expression, have always been a popular choice to depict and understand human emotions. Music is the form of art known to have a greater connection with a person's emotion. It has got a unique ability to lift up one's mood. If a user receives a recommendation based on his/her mood, it will also improve his/her listing experience. The human face plays an important role in knowing an individual's mood. The required inputs are extracted from the human face directly using a camera. One of the applications of this input can be for extracting the information to deduce the mood of an individual. This data can then be used to get a list of songs that comply with the mood derived from the input provided earlier. This eliminates the time-consuming and tedious task of manually segregating or grouping songs into different lists and helps in generating an appropriate playlist based on an individual's emotional features. Reliable emotion based classification systems can go a long way in helping us parse their meaning.

In this proposed model, we present an effective Emotion Based Music Recommendation System , which recommends music based on the real-time mood of the user. Emotion Detection is done by taking an image of the user's face as an input and identifying their mood using convolutional neural networks. Based on the emotion detected, a playlist is recommended to the user which consists of songs suitable to the mood of the user.

Keywords: Emotion detection, music recommendation, convolutional neural networks, camera.

TABLE OF CONTENT

TABLE OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 PROBLEM DEFINITION

Where words fail, music speaks. In the present-day environment, there is a lot of stress and fatigue, which leads to more despair and ill health. Medical studies demonstrate that interactive and high-quality games and shows, as well as music tailored to one's mood, can lift one's spirits. By reducing stress hormones, it can aid in the treatment of depression along with helping in physical aches and pains induced by the high level of stress.

However, additional research shows that listening to random music that is unrelated to one's mood can have a stressful effect on people. As a result, listening to music to unwind after work can improve one's health.A great song should lift your heart, warm the soul and make you feel good. But music is an ocean and in that it is often confusing for a person to decide which music he/she has to listen to from a massive collection of existing options.

An Emotion-Based Music Player provides a better platform for all music listeners by automating song selection and updating playlists regularly based on the user's identified emotion. These assist users in organizing and playing songs based on their mood, making them less stressed. So, the goal is to design a recommender system on the music and emotion domain . Human face plays an important role in the extraction of an individual's emotional state.

1.2 EXISTING APPLICATIONS

a) Music Recommendation System using Content and Collaborative Filtering Methods

Rapid development of mobile devices and the internet has made it possible for us to access different music resources freely. While the Music industry may favor certain types of music more than others, it is important to understand that there isn't a single human culture on earth that has existed without music. In this paper, they have designed, implemented and analyzed a song recommendation system. They have used the Song Dataset provided to find correlations between users and songs and to learn from the previous listening history of users to provide recommendations for songs which users would prefer to listen to most. The dataset contains over ten thousand songs and listeners are recommended the best available songs based on the mood, genre, artist and top charts of that year. With an interactive UI they show the listener the top songs that were played the most and top charts of the year. Listeners also have the option to select his/her favorite artist and genres on which songs are recommended to them using the dataset.

b) First Mood-Based Music Recommendation System

A music recommendation system that uses an adaptation of the "Valence-Arousal Plane" alongside vector-distance measurements to match tracks that convey a similar type of emotion/mood. This approach is fundamentally different from collaborative filtering methods since it attempts to extract the inner qualities of a song without relying on user data. This is especially helpful for those who do not operate a large music platform and have billions of relevant user data points. Another plus is that this method's implementation is fast and free, from

collecting the data to implementing the actual recommendation algorithm. Finally, recommending music across genre categories is a desirable quality of any real-world recommender system, since this supports the user in engaging with new kinds of music and developing their personal music taste.

1.3 NEED FOR THE SYSTEM

People tend to express their emotions, mainly by their facial expressions.The aim is to develop an Emotion Based Music Recommendation System, which is an web application meant for users to minimize their efforts in managing large playlists.The proposed model will extract the user's facial expressions to determine the current mood to the user.Once the emotion is detected, a playlist of songs suitable to the mood of the user will be presented to the individual.

The project helps to lighten the mood of the user, by playing songs that match the requirements of the user by capturing the image of the user. By developing a recommendation system, we can assist a user to make a decision regarding which music one should listen to. The user would 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 songs would be shown to the user according to his/her mood.The initial setup of the functionality of the project was to detect emotion using Matlab. The geometric manipulation procedure is used here to measure the emotion levels. But then it was observed that Matlab had some serious limitations. The processing requirement capacities for Matlab were very high. Also, detection of speed in real-time processing was not an easy task.

The processing capacity of Matlab was only 4-5 frames per second. In the case of a system with a low RAM, this was even 9% lower. This is where OpenCV comes into use. Open-source computer vision library (OpenCV) is specially and

specifically designed for computational efficiency which is strongly focused on real-time applications. It helps in efficiently building highly sophisticated vision-based applications. OpenCV is very useful in satisfying the low processing power and high-speed requirements of the given application.

The proposed methodology “Emotion based music recommendation system” provides similar functionality but with better results and along with emotion detection using facial expressions.

CHAPTER 2

LITERATURE SURVEY

2.1 Analysis of Facial Expression and Recognition Based on Statistical Approach

[1] presents the methodology for an efficient facial expression analysis and classification. statistical parameters such as entropy, skewness and kurtosis are used to extract the features. In the first step the average face is extracted, then we compute the difference image. Statistical information is retrieved from different images. Entropy retrieves one value from each image, skewness and kurtosis retrieve 106 values respectively. Combinations of all these values are used as a feature vector for each image which helps for classification. Skewness and kurtosis retrieves 106 values from image respectively. By using Feedforward neural network classification is done. Scaled Conjugate Gradient Backpropagation algorithm is used to train and test the network. The implementation of neural networks and the training method were done with the Neural Networks Toolbox of MATLAB 7.11.0.584. The SCG training algorithm outperformed in this experiment by classifying the input data in 773 epochs with the average training time of 15 seconds. The confusion matrix gives the accuracy of the classification problem. 180 images of 10 subjects are classified into six categories with 92.2% accuracy. The diagonal elements in the confusion matrix show the classified groups. For the input we gave 30 images of every class. The Resulting Confusion matrix illustrates that, out of 30 samples 27 samples fall in anger class and remaining one in disgust and two are in class surprise. Similarly classification is done with all classes as shown in the confusion matrix. Therefore total average accuracy of classification is 92.2% and misclassification or error rate is 7.8%.

From this project , it will be really useful to us as we are making a real-time system which uses Facial expression and recognition.

2.2 Emotion Recognition From Facial Expressions and Its Control Using Fuzzy Logic

In this project the fuzzy relational model for emotion detection is done by examination of a large facial database which reveals that the degree of a specific human emotion, such as happiness or anger, greatly depends on the degree of MO (Mouth Orbit), EO (Eye Orbit), and the length of EBC. For the segmentation of the mouth region, a color-sensitive segmentation algorithm is most appropriate, a color-sensitive FCM clustering algorithm has been selected for the segmentation of the mouth region.The segmentation of the eye regions is successfully performed by the traditional thresholding method. The hair region in the human face is segmented by the thresholding technique. Segmentation of the mouth and eye regions is required for the determination of MO and EO, respectively. Constriction in the forehead region can be explained as a collection of white and dark patches called hilly and valley regions, respectively. To determine the length of EBC on the forehead region, variation in intensity along the x-axis of the selected rectangular region is scanned. The maximum x-width that includes variation in intensity is defined as the length of EBC. The measurements obtained on MO, EO, and EBC are encoded into three distinct fuzzy sets: HIGH, LOW, and MODERATE. From this project , it will be really useful to us as we are making a real-time system which uses Facial expression and recognition.

2.3 Facial Expression Recognition with Identity and Emotion Joint Learning

In this project the Facial Expression Recognition (FER) is a well defined task, aiming to recognize facial expressions with discrete categories or continuous levels from still images or videos. Different subjects express the same specific facial expression in different ways due to the inter-subject variability and their facial biometric shapes.In the proposed model a transfer learning method from face recognition to FER using CNN directly. The model extracts the highlevel identity feature from the face recognition network and considers it as an auxiliary input feature for our FER model. This model concatenates both the high-level emotion and identity features as Tandem Facial Expression (TFE) features and feeds it to the subsequent fully connected layers to form a new network. In this project ,the CNN architecture to discover latent identity and emotion features. First, we pre-train latent emotion and identity features separately using two different CNNs for emotion and DeepID. This work evaluates the proposed method on two popular FER datasets, namely Extended Cohn-Kanade (CK+) database and FER+ database. These two FER datasets are used for deep-learned emotion feature extraction and joint learning, while the identity features are learned from the CASIA-WebFace database.In this dataset, the face images are collected from the Internet, containing 10,575 subjects and 494,414 images. It is usually considered as a standard large scale training dataset for face recognition challenges .For the DeepID network, we follow the same parameter setting in , with a 160-dimensional representation in the fully-connected layers. A dropout layer is used after the DeepID layer, with a probability of 0.4 to reduce overfitting .For the deep ResNet, we have two different settings for the number of layers. From this project , it will be really useful to us as we are making a real-time system which uses Facial expression recognition with Emotion joint learning.

2.4 Music Recommendation System Using Emotion Triggering Low-level Features

This project proposes a preference analysis method based on empirical evaluation scores for the selection of general and reliable low level features of music that triggers emotions.This proposed model implemented a personalized music recommendation system based on the low-level features selected by the proposed method.It uses emotion triggering low-level feature selection method based on large number of subjective audience ratings.The low-level features that are selected by the proposed method depend on the training set of songs and evaluation results. To check if there is any peculiar competition in the 16 competitions of training set, in which the evaluation of the competition may have been influenced by the specific set of songs performed or by the specific audience panel with special preference, we also constructed 16 sets of training sets by selecting competitions out of candidate competitions, in addition to the training set with all of the 16 candidate competitions. From the 16 training sets composed of 15 competitions, 10 to 19 features are selected from individual training sets with a total of 40 distinct features. From the training set with all the competitions, features are selected. Out of the 17 features extracted from the training set of competitions, all the features appear at least in one of the 16 training sets of competitions, features appear in more than four of 16 sets of 15 competitions, and 11 features appear in more than eight of 16 sets of 15 competitions. Out of 40 features extracted from the 16 training sets of 15 competitions, 20 features appeared in more than 4 training sets, and 13 features appeared in more than 8 training sets. Each module is used in two types of experiments, which are prediction of competition results based on audience panel evaluation and recommendation of the user's preferred songs.

2.5 A Brief Review of Facial Emotion Recognition Based on Visual Information

In this project,Facial emotions are important factors in human communication that help us understand the intentions of others. People infer the emotional states of other people, such as joy, sadness, and anger, using facial expressions and vocal tone.In conventional FER approaches, the reference has the highest performance than other algorithms. This study tried to compute difference information between the peak expression face and its intra class variation in order to reduce the effect of the facial identity in the feature extraction. Because the feature extraction is robust to face rotation and misalignment, this study achieves relatively accurate FER than other conventional methods .A complex CNN network proposed consists of two convolutional layers, each followed by max pooling and four Inception layers. This network has a single-component architecture that takes registered facial images as the input and classifies them into one of six basic or one neutral expression. The highest performance approach also consists of two parts. In the first part, the spatial image characteristics of the representative expression-state frames are learned using a CNN. In the second part, the temporal characteristics of the spatial feature representation in the first part are learned using an LSTM of the facial expression. Based on the accuracy of a complex hybrid approach using spatio-temporal feature representation learning, the FER performance is largely affected not only by the spatial changes but also by the temporal changes. These memories demanding and computational complexities make deep learning ill-suited for deployment on mobile platforms with limited resources

CHAPTER 3
SYSTEM ANALYSIS

3.1 SYSTEM REQUIREMENTS

The following specifications were those required by the system for the software's successful implementation and functioning.

3.1.1 REQUIREMENTS

SOFTWARE REQUIREMENTS

- PYTHON 3.7
- SPYDER (ANACONDA)
- EMOTION DETECTION ALGORITHM

BASIC HARDWARE REQUIREMENTS

- WEBCAM
- CPU - Intel Core i5
- GPU - 8GB for Nvidia 100
- 4GB RAM

PYTHON

The python programming language is a high-level, object-oriented, and interpreted programming language. Python is of best use when it comes to Rapid Application Development and this due to the high-level built-in data structures which are combined with dynamic typing and dynamic binding. This programming language has a simple and easy-to-learn syntax which emphasizes readability and reduces the overall cost of program maintenance. This high-level programming

language can also be used as a scripting or glue language to connect the existing components. It supports a lot of modules and packages and encourages program modularity and code reusability. Python has an interpreter and an extensive standard library that are made available in source or binary form without charge for all major platforms, and it is an open-source language.

SPYDER (ANACONDA 3)

Spyder is a free and open source scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.

3.1.2 PACKAGES

These are the packages that were downloaded and utilized in our project

- Flask==2.1.2
- spotipy==2.19.0
- pandas==1.4.2
- gunicorn==20.1.0
- opencv_python_headless==4.5.5.64
- numpy==1.22.4
- pillow==9.1.1
- tensorflow==2.9.0

Flask

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have a built-in abstraction layer for database handling, nor does it have form validation support. Instead, Flask supports the extensions to add such functionality to the application.

OpenCV

OpenCV is a huge open library for image analysis, computer-vision, and machine learning. Python, C++, Java, and many other programming languages have been accepted by OpenCV. It can scan pictures and videos to identify objects, faces, and even human handwriting.

Spotipy

Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform.

Pandas

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

Tensorflow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

Numpy

NumPy is a Python library that enables users to interact with arrays. It also provides numerous tools in linear algebra, the Fourier transform, and matrices, including a huge handful of international mathematical functions to handle these arrays.

Pillow

Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux.

CHAPTER 4

SYSTEM DESIGN

4.1 OBJECT ORIENTED DESIGN

Identifying the objects in a system would be what OO (Object Oriented) analysis and design have always been about identifying their relationships. Generating a design that could be transformed into executables utilising object oriented programming languages.

UML (Unified Modelling Language) is a standard language that uses, designs, constructs, and documents software components. The Unified Modelling Language (UML) is an effective instrument enabling ObjectOriented Analysis and Design. The Object Management Group (OMG) created it, and in January 1997, the OMG proposed UML 1.0 as a specification draft. It started as a way to capture the behaviour of vast software and non-software systems and has since grown into such an international standard. UML is created to be process generic, indicating it could be used in a variety of circumstances. It can be used for a spectrum of uses. Business analysts, software architects, and developers are all using UML as a common language. It can be used to describe, specify, build, and document the system's business processes, including its structural and behavioural artefacts.

UML is a modelling language used to create software blueprints. Diagrams are categorized into two divisions, which are further divided into subcategories:

- **Structural Diagrams**
- **Behavioral Diagrams**

4.2 STRUCTURAL DIAGRAMS

The static aspect of the system is portrayed by the structural diagrams. These static aspects are the components of a diagram that describes the main structure and are thus stable. Classes, interfaces, objects, components, and nodes are often used to represent them.

Some of the structural diagrams are:

- Class diagram
- Object diagram
- Component diagram
- Deployment diagram

4.3 BEHAVIORAL DIAGRAMS

Behavioral diagrams illustrate a system's complex nature. The changing/moving parts of a system are usually known for the dynamic aspect.

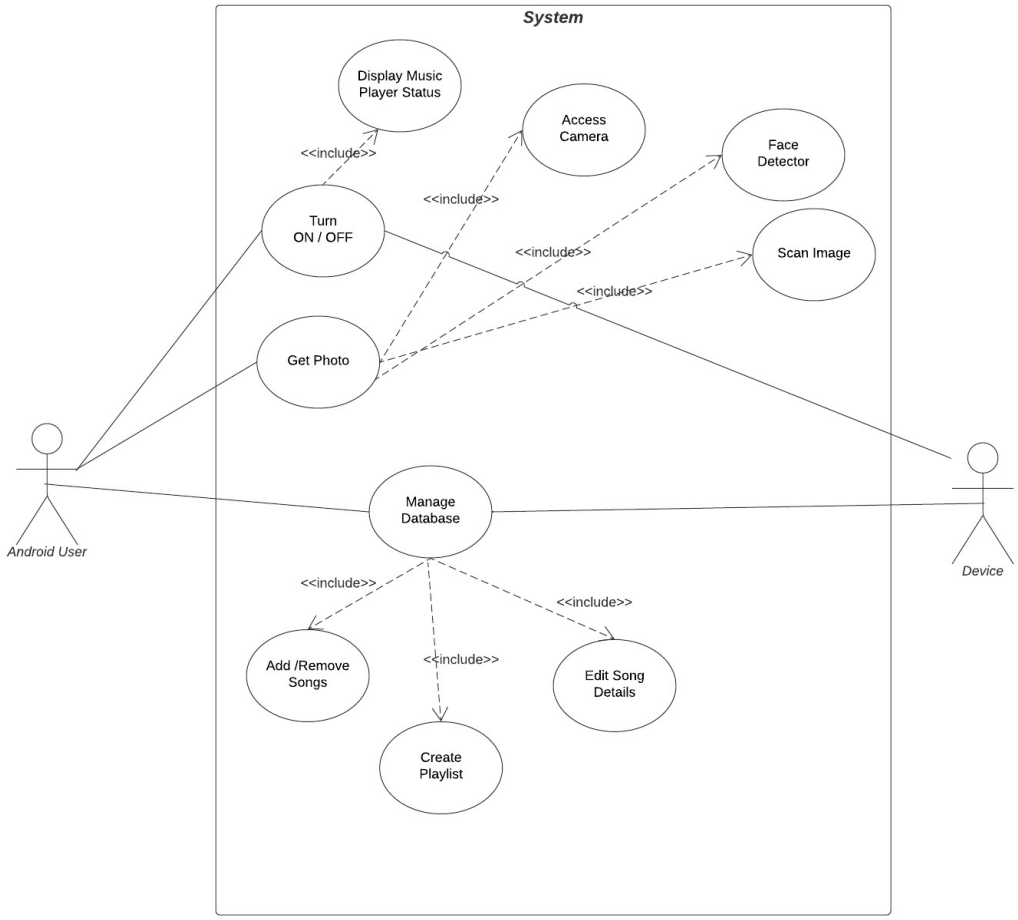
The following five types of behavioral diagrams are supported in UML:

- Use case diagram
- Sequence diagram
- Collaboration diagram
- Statechart diagram
- Activity diagram

4.4 USE CASE DIAGRAM

The use case diagrams show the system's behavior concerning the deployment environment. It illustrates the proposed system's users. A use case is a system analysis methodology for finding, explaining, and monitoring program needs.

A use case is a collection of conceivable sequences of interactions between systems and users in a specific environment, all of which are tied to a specific purpose. A use case is represented by an ellipse with the name of the use case. A stick figure with a name is used to represent an actor.

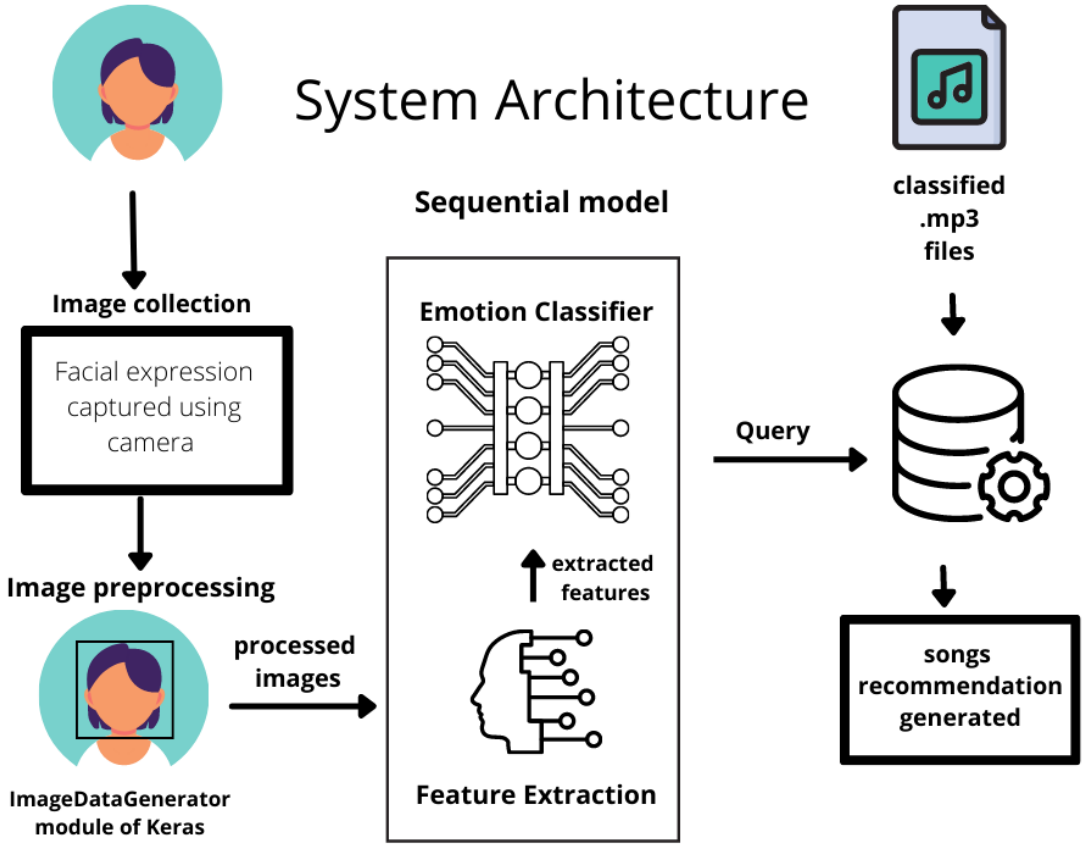


1.1 UseCase Diagram

4.5 ARCHITECTURE DIAGRAM

Application captures the user's image using the webcam and the face is detected from the image through preprocessing. Using the features extracted, emotion of the user is predicted by the sequential model. The emotions that the model can predict are happy, sad, neutral, surprised, angry, fearful and

disgusted. Based on the mood predicted, playlist is generated by querying the database.



1.2 Architecture Diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

5.1 TECH STACK

Keras:

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code. The code is hosted on GitHub, and community support forums include the GitHub issues page, and a Slack channel.

Deep Learning is becoming a very popular subset of machine learning due to its high level of performance across many types of data. A great way to use deep learning to classify images is to build a convolutional neural network (CNN). The Keras library in Python makes it pretty simple to build a CNN. Computers see images using pixels. Pixels in images are usually related. For example, a certain group of pixels may signify an edge in an image or some other pattern. Convolutions use this to help identify images.

Tensorflow :

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

TensorFlow was developed by the Google Brain team for internal Google use in research and production. The initial version was released under the Apache License 2.0 in 2015. Google released the updated version of TensorFlow, named TensorFlow 2.0, in September 2019.

TensorFlow can be used in a wide variety of programming languages, most notably Python, as well as Javascript, C++, and Java. This flexibility lends itself to a range of applications in many different sectors.

Spotipy :

Spotipy is a lightweight Python library for the Spotify Web API. With Spotipy you get full access to all of the music data provided by the Spotify platform. It enables and supports all the features of the Spotify API but allows them to be designed and run in Python. This allows for scalability and also will come in handy when we start dealing with large amounts of data. It is a well-built module and has an active team supporting development, so you don't have to worry about catastrophic bugs, but it is still better to get in the habit of creating virtual environments for your project.

Spotipy supports all of the features of the Spotify Web API including access to all endpoints, and support for user authorization. For details on. the capabilities you are encouraged to review the Spotify Web API documentation.

Flask :

Flask is a web application framework written in Python. It is developed by Armin Ronacher, who leads an international group of Python enthusiasts named Pocco. Flask is based on the Werkzeug WSGI toolkit and Jinja2 template engine. Both are Pocco projects.Flask is often referred to as a micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have a built-in abstraction layer for database handling, nor does it have form validation support. Instead, Flask supports the extensions to add such functionality to the application.

5.2 MODEL TRAINING

The main purpose of training a model is to find a set of weights and biases that have low loss across all examples. Training a model provides a way to learn and determine good values for all the weights and the bias from a particular example.

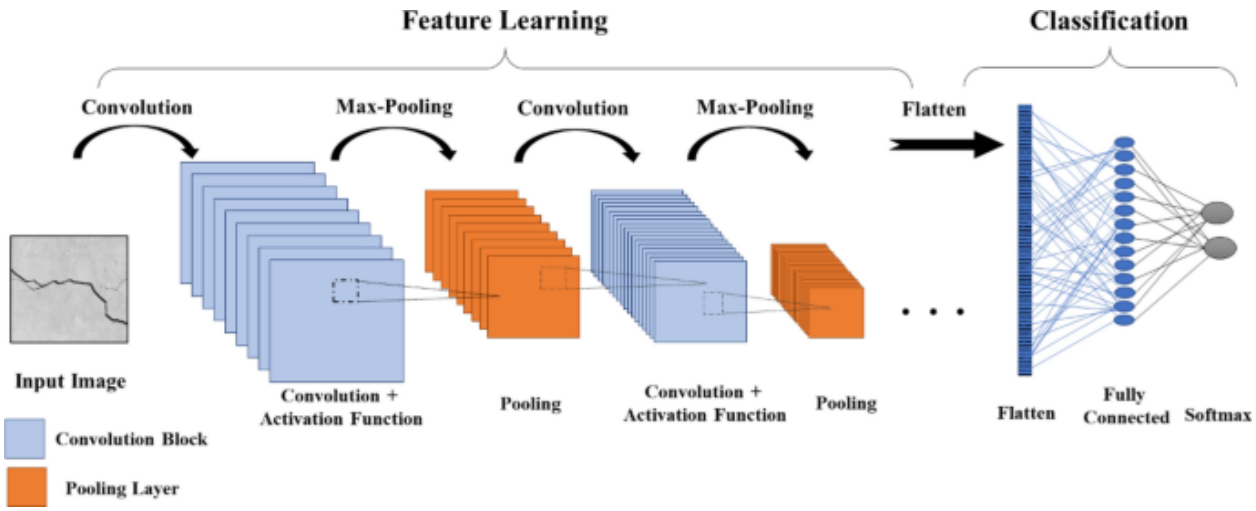
The model training stage consists of model building and compiling it using Keras. The model type used for the project is sequential which allows building convolutional neural networks layer by layer in Keras. Layers are added to the model using the add() function. The proposed model consists of conv2d, maxpooling2d, dropout, dense layers and fallen layer. Finally, the model training accuracy is obtained.

5.2.1 CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks(CNN) is a deep neural network, which is most commonly applied to analyzing visual imagery. CNN has numerous applications in image segmentation, object detection, medical image analysis, image captioning, image classification, semantic segmentation, image and video recognition, brain-computer interfaces, financial time series, natural language

processing, and recommender systems. CNNs were mainly inspired by the structure of the human brain, in which the biological processes and the connectivity pattern between neurons resembles the organization of the animal visual cortex.

CNNs generally consist of Input Layer, Hidden Layer, and Output Layer. The main layers of the CNN are as follows: Convolutional Layer, Pooling Layer, Fully connected layer, Dropout Layer, and Activation Functions.



1.3 Convolutional Neural Networks Architecture

Convolutional Layer

A Convolutional Layer could be defined as the core building block of CNN, which carries the main portion of the network's computational load. It is also called the feature extraction layer as the features of the image get extracted within this layer. The layers are made from several feature maps. Each unit of feature map is made from convolving a small region in input data which is called the local receptive field.

The layer generally performs a dot product between a matrix, which is the set of learnable parameters, also known as a kernel, and another matrix, which is the

restricted portion of the receptive field. The kernel is spatially smaller than an image but depthwise, it is deeper.The kernel slides across the height and width of the image during the forward pass. This produces a two-dimensional representation of the image known as an activation map. It is used to give the response of the kernel at each spatial position of the image. The sliding size of the kernel is called a stride. Then it slides the filter over the next receptive field of the same input image by a stride and does the same operation again. Here we have used kernel of size three. The process is repeated again and again until the whole image is covered. The output will be the input for the next layer. Convolution layer contains ReLU activation to make all the present negative values to zero. Conv2D is used as the convolutional layer with different filter sizes ranging from 32 to 128.

Pooling Layer

Pooling layers, also known as subsampling layers, are used immediately after the convolution layers and they usually make a condensed feature map from each feature map in convolutional layers. The pooling operation is processed on every slice of the representation individually. This helps to reduce the spatial size of the representation, which decreases the required amount of computation and weights. There are different kinds of pooling, such as local, global, max, average, etc. Local pooling combines small clusters, whereas Global pooling acts on all the neurons of the feature map. The two types of pooling commonly used are the max and average pooling layers. Max pooling uses the maximum value of each local cluster of neurons in the feature map, while average pooling takes the average value of the clusters of neurons of the feature map. Pooling layers consist of two hyperparameters, such as filters, and strides. Filters detect the changes in the intensity values of the images, in order to detect the spatial patterns of the same. Strides could be described as the number of pixels that have shifted over the input

matrix. The pooling layers plays an important role in reducing the computational time and overfitting issues in the CNN architecture. MaxPooling2D is used with pool size 2.

Fully Connected Layers

Fully connected layers are also called feed forward neural networks, which are generally used to connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multi-layer perceptron neural network. It can be one or more layers which are placed after a sequence of convolution and pooling layers. The flattened matrix goes through a fully connected layer to classify the images.

The fully connected layer in the CNN is used to represent the feature vector for the input. This feature vector holds information that is vital to the input. When the network gets trained, this feature vector is then further used for classification, regression, or input into other networks for translating into other types of output, thus finally, determining the loss and helping the network to get trained. It can also be used as an encoded vector. The fully connected layer consists of a final layer called a softmax layer, which converts the output of the last layer in the neural network into a probability distribution. The main purpose of fully connected layers are feature extraction and classification. CNNs capture better representation of data and hence there is no need to do feature engineering.

Dropout Layer

Dropout could be defined as an approach to regularization in neural networks which helps to reduce interdependent learning amongst the neurons. Usually, when all the features are connected to the Fully Connected 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 on the model's performance when used on test data.

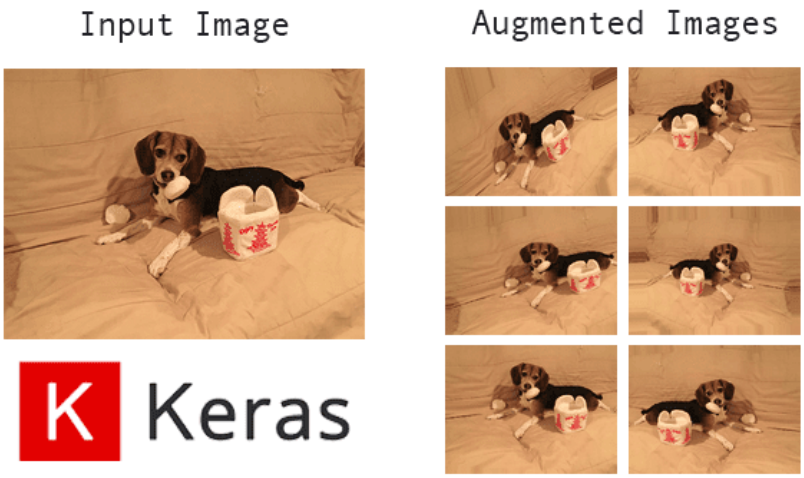
To overcome this problem, a dropout layer is utilised wherein a few neurons are dropped from the neural network during the training process resulting in reduced size of the model. Dropout layers usually prompt neural networks to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. Also, training time for each epoch can be made less by roughly doubling the number of iterations required to converge in the neural network. In the testing phase, the entire network can be considered and each activation can be reduced by a factor p. On passing a dropout of 0.3, 30% of the nodes are dropped out randomly from the neural network. In the proposed system, Dropout is set to 0.25 as anything above resulted in poor performance.

Activation Function

Activation functions are used to add non-linearity to the network. It is also called a transfer function. They are generally used to learn and approximate any kind of continuous and complex relationship between the variables of the network. They decide which information of the model should fire in the forward direction and which ones should not at the end of the network. It also defines how the weighted sum of the input can be transformed into an output from the nodes in a layer of the network. For the convolutional layer, ReLu is used as the activation function. Rectified Linear Activation function has been proven to work well in neural networks. Softmax is used for dense layer which makes the output sum up to 1 so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

5.3 IMAGE PREPROCESSING:

The aim of pre-processing is to improve the quality of the image so that we can analyze it in a better way. By preprocessing we can suppress undesired distortions and enhance some features which are necessary for the particular application we are working for. Those features might vary for different applications.All images can be represented using pixels. Images are essentially just matrices where the individual cells of the matrix contain pixel data. An image can be divided into a grid of pixels. Each element of this grid is used to hold pixel information, and the value of a pixel depends on the type of image that you're working with.This pixel representation of image matrices can be fed into a machine learning model such as those built using neural networks.Computers see an input image as an array of pixels, and it depends on the image resolution. Based on the image resolution, it will see height * width * dimension.Each pixel represents only intensity information, so you have one value in the range 0–1. A value of 1 means a pixel with the highest intensity, 0 represents a pixel with no intensity at all. Image Preprocessing is performed using the ImageDataGenerator module of Keras.

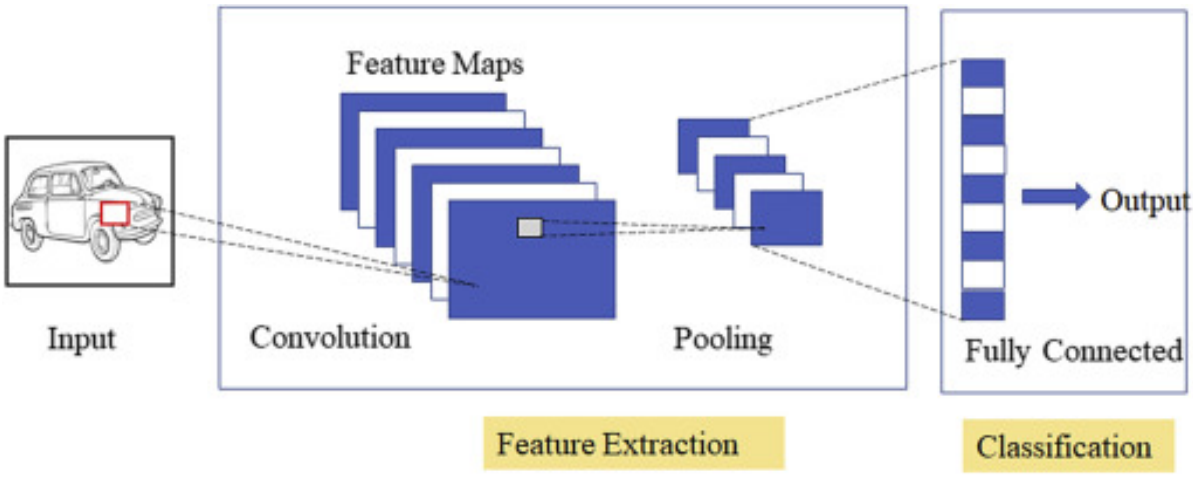


1.4 Data Augmentation using Keras ImageDataGenerator

5.4 FEATURE EXTRACTION

Feature extraction techniques play a very important role in image processing and pattern recognition domains. They are very much helpful in various image processing applications. They describe the relevant shape information contained in a pattern. This is done to make the task of classifying the pattern easier by a formal procedure.Various image preprocessing techniques are needed to be performed on the input images before getting the necessary features extracted from them. Feature extraction helps to get the best features from the segmented images by selecting and combining variables into features,thus effectively reducing the amount of data. The main goal of feature extraction is to obtain the most relevant information from the original data. As features are used to define the behavior of an image, they show their place in terms of storage taken, efficiency in classification and also in time consumption.

In the proposed system, we are using a Convolutional Neural Network for feature extraction. The true fact is that CNNs provide automatic feature extraction, which is the primary advantage[6]. The specified input data is initially forwarded to a feature extraction network, and then the resultant extracted features are forwarded to a classifier network.



1.5 Feature Extraction and Classification in CNN

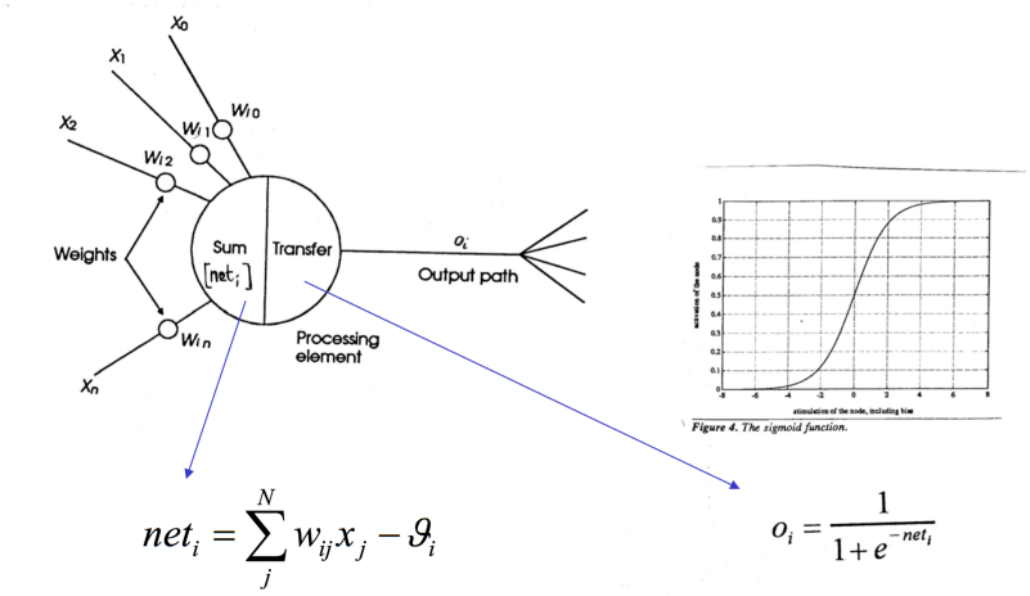
The feature extraction network comprises loads of convolutional and pooling layer pairs. Convolutional layer consists of a collection of digital filters to perform the convolution operation on the input data. The pooling layer is used as a dimensionality reduction layer and decides the threshold. During backpropagation, a number of parameters are required to be adjusted, which in turn minimizes the connections within the neural network architecture.

5.5 EMOTION CLASSIFICATION AND SONG RECOMMENDATION

Emotion Classification is the final step of the proposed system. Here, emotions are classified according to the facial expression of the user. It can also be described as the process of classifying images into different emotion categories based on their features. In our system, using the extracted features from the sequential model from the feature extraction stage, the classification is done into one of the seven classes, namely: Happy, Sad, Angry, Fearful, Disgust, Surprised and Neutral.

In the proposed system, the dense layer of sequential model is used as the classifier to classify images based on the output from convolutional layers.

Dense Layer is a simple layer of neurons in which each neuron receives input from all the neurons of the previous layer, thus called as dense.



1.6 Working of a single neuron.

A layer contains multiple numbers of such neurons.

Each Layer in the Neural Network contains neurons, which compute the weighted average of its input and this weighted average is passed through a nonlinear function, called “activation function”. Result of this activation function is treated as the output of that neuron. In a similar way, the process is carried out for all neurons of all layers.The output of the last layer will be considered as output for that image. Here, the output layer has seven neurons with each neuron representing one oh the emotions with “softmax activation” function. Softmax activation function is used when we have two or more than two classes. Softmax makes the output sum up to one so the output can be interpreted as probabilities. The model will then make its prediction based on which option has the highest probability.

Finally, based on the emotion detected the playlist suitable for that particular mood is generated to the user. The app will fetch the playlist of songs from Spotify through spotipy wrapper and recommend the songs by displaying them on the screen.

CHAPTER 6
PERFORMANCE METRICS

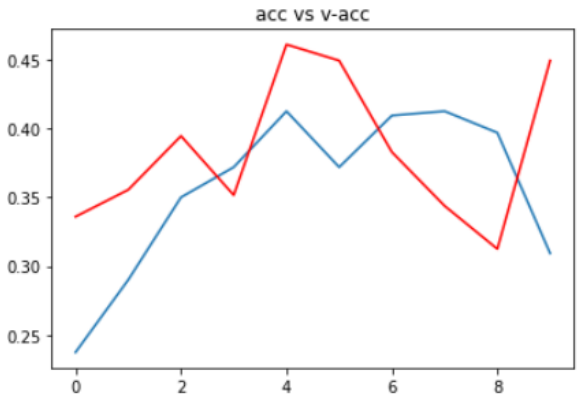
Performance Metrics are used to measure and summarize the quality of the trained classifier when tested with new data. It is mainly used to evaluate the generalization capability of the trained classifier.

6.1 ACCURACY

Accuracy is a metric that generally describes how the model performs across all classes. It is calculated as the ratio between the numbers of correct predictions to the total number of predictions.

TRAINING ACCURACY

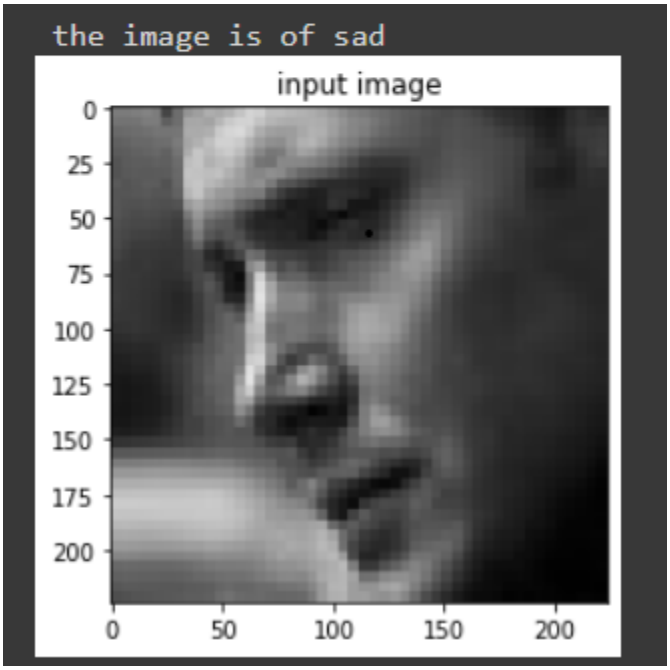
Model achieved a training accuracy of 66%.



1.7 Graph plotting training accuracy vs validation accuracy

TEST ACCURACY

Finally, a test image was input to the system, segmentation and classification was performed on that image and the accuracy was predicted. The image obtained an accuracy of 90% and was classified as sad.



1.8 Predicted output

CHAPTER 7
CONCLUSION AND FUTURE WORK

7.1 CONCLUSION

Paying attention to various factors, such as particular context, personal parameters, feelings and emotions, is highly important to a decision-making process of recommendations. Contemporary music recommendation systems face the gap in personalization, human feelings, contextual preferences and emotional factors while suggesting music. In this model, we proposed an emotion-driven recommendation system with respect to personalized preferences and particular life and activity contexts. The approach presented in this model is targeted to provide maximum benefits for people from the music listening experience.

7.2 FUTURE WORK

Further work on the implementation and testing of the recommendation engine, are considered for the next step when the appropriate amount of the data will be collected. Music creation by artificially intelligent systems with particular music attributes to move states of human emotions can be considered as the further elaboration work in this context. We intend to expand on the concept by adding a heart rate sensor to predict emotion based on heart rate variability(HRV) since sometimes, humans may involuntarily or deliberately conceal their real emotions (so-called social masking). The use of physiological signals can lead to more objective and reliable emotion recognition.

APPENDIX

APPENDIX A - SOURCE CODE

Module for video streaming, frame capturing, prediction and recommendation which are passed to main.py

camera.py

```
import numpy as np
import cv2
from PIL import Image
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from pandastable import Table, TableModel
from tensorflow.keras.preprocessing import image
import datetime
from threading import Thread
from Spotipy import *
import time
import pandas as pd

face_cascade=cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
ds_factor=0.6

emotion_model = Sequential()
```

```
emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))
emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))
emotion_model.load_weights('model.h5')

cv2.ocl.setUseOpenCL(False)

emotion_dict =
{0:"Angry",1:"Disgusted",2:"Fearful",3:"Happy",4:"Neutral",5:"Sad",6:"Surprised
"}
music_dist={0:"songs/angry.csv",1:"songs/disgusted.csv
",2:"songs/fearful.csv",3:"songs/happy.csv",4:"songs/neutral.csv",5:"songs/sad.csv
",6:"songs/surprised.csv"}
global last_frame1
last_frame1 = np.zeros((480, 640, 3), dtype=np.uint8)
global cap1
```

```
show_text=[0]


''' Class for calculating FPS while streaming. Used this to check performance of
using another thread for video streaming '''
class FPS:
    def __init__(self):
        # store the start time, end time, and total number of frames
        # that were examined between the start and end intervals
        self._start = None
        self._end = None
        self._numFrames = 0
    def start(self):
        # start the timer
        self._start = datetime.datetime.now()
        return self
    def stop(self):
        # stop the timer
        self._end = datetime.datetime.now()
    def update(self):
        # increment the total number of frames examined during the
        # start and end intervals
        self._numFrames += 1
    def elapsed(self):
        # return the total number of seconds between the start and
        # end interval
        return (self._end - self._start).total_seconds()
```

```
    def fps(self):
        # compute the (approximate) frames per second
        return self._numFrames / self.elapsed()


''' Class for using another thread for video streaming to boost performance '''
class WebcamVideoStream:

        def __init__(self, src=0):
            self.stream = cv2.VideoCapture(src,cv2.CAP_DSHOW)
            (self.grabbed, self.frame) = self.stream.read()
            self.stopped = False

        def start(self):
                # start the thread to read frames from the video stream
            Thread(target=self.update, args=()).start()
            return self

        def update(self):
            # keep looping infinitely until the thread is stopped
            while True:
                # if the thread indicator variable is set, stop the thread
                if self.stopped:
                    return
                # otherwise, read the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()
```

```
        def read(self):
                # return the frame most recently read
                return self.frame
        def stop(self):
                # indicate that the thread should be stopped
                self.stopped = True

''' Class for reading video stream, generating prediction and recommendations '''
class VideoCamera(object):

    def get_frame(self):
            global cap1
            global df1
            cap1 = WebcamVideoStream(src=0).start()
            image = cap1.read()
            image=cv2.resize(image,(600,500))
            gray=cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
            face_rects=face_cascade.detectMultiScale(gray,1.3,5)
            df1 = pd.read_csv(music_dist[show_text[0]])
            df1 = df1[['Name','Album','Artist']]
            df1 = df1.head(15)
            for (x,y,w,h) in face_rects:
                    cv2.rectangle(image,(x,y-50),(x+w,y+h+10),(0,255,0),2)
                    roi_gray_frame = gray[y:y + h, x:x + w]
                    cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray_frame, (48, 48)), -1), 0)
                    prediction = emotion_model.predict(cropped_img)
```

```
                    maxindex = int(np.argmax(prediction))
                    show_text[0] = maxindex
                    cv2.putText(image, emotion_dict[maxindex], (x+20, y-60),
cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)
                    df1 = music_rec()

                global last_frame1
                last_frame1 = image.copy()
                pic = cv2.cvtColor(last_frame1, cv2.COLOR_BGR2RGB)
                img = Image.fromarray(last_frame1)
                img = np.array(img)
                ret, jpeg = cv2.imencode('.jpg', img)
                return jpeg.tobytes(), df1

def music_rec():
        df = pd.read_csv(music_dist[show_text[0]],dtype=str)
        df = df[['Name','Album','Artist']]
        df = df.head(15)
        return df
```

Module for establishing connection to and getting tracks from Spotify using Spotipy wrapper

spotipy.py

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

```
import pandas as pd
import time

auth_manager = SpotifyClientCredentials(",")
sp = spotipy.Spotify(auth_manager=auth_manager)

def getTrackIDs(user, playlist_id):
    track_ids = []
    playlist = sp.user_playlist(user, playlist_id)
    for item in playlist['tracks']['items']:
        track = item['track']
        track_ids.append(track['id'])
    return track_ids

    def getTrackFeatures(id):
    track_info = sp.track(id)

    name = track_info['name']
    album = track_info['album']['name']
    artist = track_info['album']['artists'][0]['name']
    # release_date = track_info['album']['release_date']
    # length = track_info['duration_ms']
    # popularity = track_info['popularity']

    track_data = [name, album, artist] #, release_date, length, popularity
    return track_data
```

Code for creating dataframe of feteched playlist

```
emotion_dict =
{0:"Angry",1:"Disgusted",2:"Fearful",3:"Happy",4:"Neutral",5:"Sad",6:"Surprised
"}
music_dist={0:"0l9dAmBrUJLylii66JOsHB?si=e1d97b8404e34343",1:"1n6cpWo9
ant4WguEo91KZh?si=617ea1c66ab6446b
",2:"4cllEPvFdoX6NIVWPKai9I?si=dfa422af2e8448ef",3:"0deORnapZgrxFY4ns
Kr9JA?si=7a5aba992ea14c93",4:"4kvSlabrnfRCQWfN0MgtgA?si=b36add73b4a7
4b3a",5:"1n6cpWo9ant4WguEo91KZh?si=617ea1c66ab6446b",6:"37i9dQZEVXb
MDoHDwVN2tF?si=c09391805b6c4651"}
```

Utility module for video streaming of web camera with threads to enable real time detection

utils.py

```
''' Class for using separate thread for video streaming through web camera'''
import cv2
from threading import Thread
class WebcamVideoStream:

        def __init__(self, src=0):
                self.stream = cv2.VideoCapture(src,cv2.CAP_DSHOW)
                (self.grabbed, self.frame) = self.stream.read()
                self.stopped = False

        def start(self):
                        # start the thread to read frames from the video stream
```

```
            Thread(target=self.update, args=()).start()
            return self

        def update(self):
            # keep looping infinitely until the thread is stopped
            while True:
                # if the thread indicator variable is set, stop the thread
                if self.stopped:
                    return
                # otherwise, read the next frame from the stream
                (self.grabbed, self.frame) = self.stream.read()

        def read(self):
            # return the frame most recently read
            return self.frame
        def stop(self):
            # indicate that the thread should be stopped
            self.stopped = True
```

Module for image processing and training the model

train.py

```
from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
```

```
train_dir = 'data/train'
val_dir = 'data/test'
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_dir,
    target_size = (48,48),
    batch_size = 64,
    color_mode = "grayscale",
    class_mode = 'categorical'
)

val_generator = val_datagen.flow_from_directory(
    val_dir,
    target_size = (48,48),
    batch_size = 64,
    color_mode = "grayscale",
    class_mode = 'categorical'
)

emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3,3), activation='relu', input_shape
= (48,48,1)))
emotion_model.add(Conv2D(64, kernel_size=(3,3), activation='relu'))
```

```
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Conv2D(128, kernel_size=(3,3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2,2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))

emotion_model.compile(loss='categorical_crossentropy',optimizer=Adam(lr=0.000
1, decay=1e-6),metrics=['accuracy'])

emotion_model_info = emotion_model.fit_generator(
    train_generator,
    steps_per_epoch = 28709 // 64,
    epochs=75,
    validation_data = val_generator,
    validation_steps = 7178 // 64
)

emotion_model.save_weights('model.h5')
```

Main flask application file

app.py

```
from flask import Flask, render_template, Response, jsonify
import gunicorn
from camera import *

app = Flask(__name__)

headings = ("Name","Album","Artist")
df1 = music_rec()
df1 = df1.head(15)
@app.route('/')
def index():
    print(df1.to_json(orient='records'))
    return render_template('index.html', headings=headings, data=df1)

def gen(camera):
    while True:
        global df1
        frame, df1 = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n\r\n')

@app.route('/video_feed')
def video_feed():
    return Response(gen(VideoCamera()),
                    mimetype='multipart/x-mixed-replace; boundary=frame')
```

```
@app.route('/t')
def gen_table():
    return df1.to_json(orient='records')

if __name__ == '__main__':
    app.debug = True
    from waitress import serve
    serve(app, host="0.0.0.0", port=8080)
```

Web page for the application

index.html

```
<html>

<head>
  <title>Emotion Music Recommendation</title>

  <style>
    img {
      padding: 20px;
      display: inline-block;
      margin: auto;
      width: 85%;
    }
  </style>
```

```
  <link
href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/css/bootstrap.min.css
" rel="stylesheet"

integrity="sha384-eOJMYsd53ii+scO/bJGFsiCZc+5NDVN2yr8+0RDqr0Ql0h+rP
48ckxlpbzKgwra6" crossorigin="anonymous" />
  <link
href="https://fonts.googleapis.com/css2?family=Bigelow+Rules&display=swap"
rel="stylesheet">
  <link type="text/css" href="{{ url_for('static', filename='/css/style.css') }}"
rel="stylesheet" />
  <script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.bundle.mi
n.js"

integrity="sha384-JEW9xMcG8R+pH31jmWH6WWP0WintQrMb4s7ZOdauHnUt
xwoG2vI5DkLtS3qm9Ekf"
    crossorigin="anonymous"></script>
</head>

<body style = "margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
    'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
  background: #eff0f4">
```

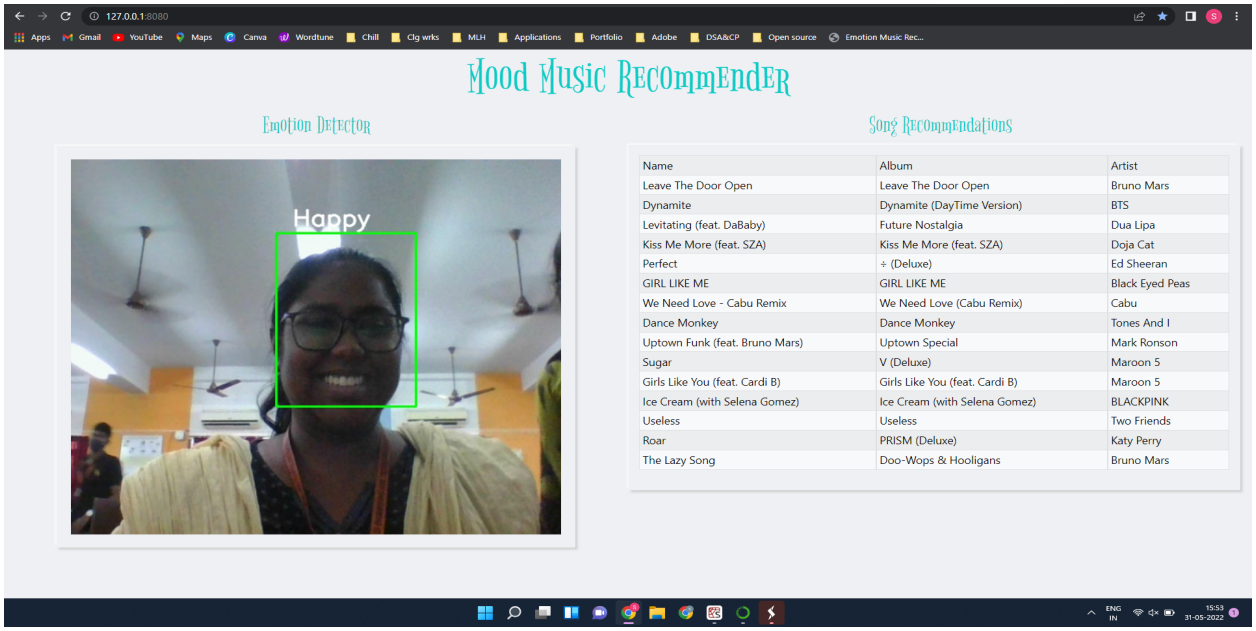
```
<div id="body">

 <div>

  <h1 align="center" style="
    font-family: 'Times New Roman';
    font-size: 62px;
    color: #0ccac4;">
    Emotion based Music Recommender
  </h1>

 </div>

 <div style="
    width: 50%;
    float: left;
    height: 100%%;
    margin: auto;
    padding-bottom:25px;
    text-align: center;
  ">
 <h2 align="center" style="
  font-family: 'Bigelow Rules';
  font-size: 36px;
  color: #0ccac4;">Emotion Detector
 </h2>
```

```
    <div style="
        margin: 10px;
        text-align: center;
        width: 51%%;
      ">

      <img class ="outer-shadow" id="bg" class="center img-fluid" src="{{
url_for('video_feed') }}" />
    </div>
    </div>
        <div style="
        width: 50%;
        float: left;
        height: 100%%;
        margin: auto;
        text-align: center;
      ">
        <h2 align="center" style="
          font-family: 'Bigelow Rules';
          font-size: 36px;
          color: #0ccac4;">Song Recommendations
        </h2>
     </div>
    <div class ="outer-shadow" id="ResultArea" style="
        padding: 15px;
        width: 49%;
        float: left;
```

```
        height: 100%%;
        margin: auto;
        text-align: center;
        margin-bottom:15px;
      ">
    </div>

  </div>

</body>

<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
<script type=text/javascript>

// Constantly Update Table
    setInterval(function() {
        $.getJSON('/t', function(data) {
            CreateHtmlTable(data);
            console.log(data,"DATA");
          });
        return false;
    }, 100);

    function CreateHtmlTable(data) {
      //Clear result div
      $("#ResultArea").html("");
      //Create table html tag
```

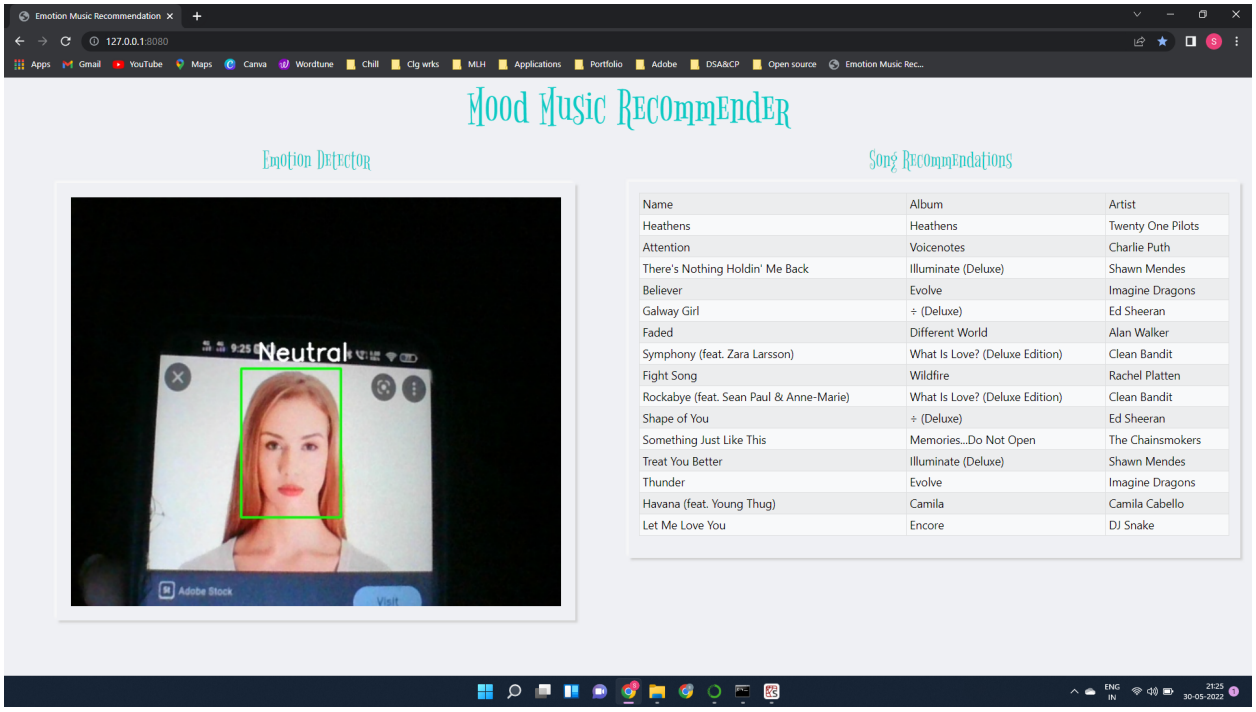
```
    var table = $("<table class = 'table table-striped table-light table-bordered
table-hover table-sm table-responsive'
id=DynamicTable></table>").appendTo("#ResultArea");
    //Create table header row
    var rowHeader =
    $("<tr></tr>").appendTo(table);
    $("<td></td>").text("Name").appendTo(rowHeader);
    $("<td></td").text("Album").appendTo(rowHeader);
    $("<td></td>").text("Artist").appendTo(rowHeader)
    //Get JSON data by calling action method in controller
    $.each(data, function (i, value) {
        //Create new row for each record
        var row =
        $("<tr></tr>").appendTo(table);
        $("<td></td>").text(value.Name).appendTo(row);
        $("<td></td>").text(value.Album).appendTo(row);
        $("<td></td>").text(value.Artist).appendTo(row);
    });
   }
  </script>
</html>
```

APPENDIX B - SNAPSHOTS



Name	Album	Artist
Leave The Door Open	Leave The Door Open	Bruno Mars
Dynamite	Dynamite (DayTime Version)	BTS
Levitating (feat. DaBaby)	Future Nostalgia	Dua Lipa
Kiss Me More (feat. SZA)	Kiss Me More (feat. SZA)	Doja Cat
Perfect	÷ (Deluxe)	Ed Sheeran
GIRL LIKE ME	GIRL LIKE ME	Black Eyed Peas
We Need Love - Cabu Remix	We Need Love (Cabu Remix)	Cabu
Dance Monkey	Dance Monkey	Tones And I
Uptown Funk (feat. Bruno Mars)	Uptown Special	Mark Ronson
Sugar	V (Deluxe)	Maroon 5
Girls Like You (feat. Cardi B)	Girls Like You (feat. Cardi B)	Maroon 5
Ice Cream (with Selena Gomez)	Ice Cream (with Selena Gomez)	BLACKPINK
Useless	Useless	Two Friends
Roar	PRISM (Deluxe)	Katy Perry
The Lazy Song	Doo-Wops & Hooligans	Bruno Mars

1.9 Emotion detected as "happy" from live webcam



Name	Album	Artist
Heathens	Heathens	Twenty One Pilots
Attention	Voicenotes	Charlie Puth
There's Nothing Holdin' Me Back	Illuminate (Deluxe)	Shawn Mendes
Believer	Evolve	Imagine Dragons
Galway Girl	÷ (Deluxe)	Ed Sheeran
Faded	Different World	Alan Walker
Symphony (feat. Zara Larsson)	What Is Love? (Deluxe Edition)	Clean Bandit
Fight Song	Wildfire	Rachel Platten
Rockabye (feat. Sean Paul & Anne-Marie)	What Is Love? (Deluxe Edition)	Clean Bandit
Shape of You	÷ (Deluxe)	Ed Sheeran
Something Just Like This	Memories...Do Not Open	The Chainsmokers
Treat You Better	Illuminate (Deluxe)	Shawn Mendes
Thunder	Evolve	Imagine Dragons
Havana (feat. Young Thug)	Camila	Camila Cabello
Let Me Love You	Encore	DJ Snake

1.10 Emotion detected as neutral

REFERENCES

[1] Londhe, Renuka & Vrushshen, P & Pawar,. (2012). Analysis of Facial Expression and Recognition Based On Statistical Approach. 2.

[2] A. Chakraborty, A. Konar, U. K. Chakraborty and A. Chatterjee, "Emotion Recognition From Facial Expressions and Its Control Using Fuzzy Logic," in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 39, no. 4, pp. 726-743, July 2009, doi: 10.1109/TSMCA.2009.2014645.

[3] M. Li, H. Xu, X. Huang, Z. Song, X. Liu and X. Li, "Facial Expression Recognition with Identity and Emotion Joint Learning," in IEEE Transactions on Affective Computing, vol. 12, no. 2, pp. 544-550, 1 April-June 2021, doi: 10.1109/TAFFC.2018.2880201.

[4] K. Yoon, J. Lee and M. Kim, "Music recommendation system using emotion triggering low-level features," in IEEE Transactions on Consumer Electronics, vol. 58, no. 2, pp. 612-618, May 2012, doi: 10.1109/TCE.2012.6227467.

[5] Ko BC. A Brief Review of Facial Emotion Recognition Based on Visual Information. Sensors (Basel). 2018 Jan 30;18(2):401. doi: 10.3390/s18020401. PMID: 29385749; PMCID: PMC5856145

[6] R. Miotto, F. Wang, S. Wang, X. Jiang, J.T. Dudley, Deep learning for healthcare: review, opportunities and challenges, Briefings in Bioinformatics (2017)