**SYMBIOSIS UNIVERSITY OF APPLIED SCIENCES**

**INDORE**

****

SOFT COMPUTING PROJECT REPORT

ON

**“Moodify: Emotion Based Music Playing System”**

Submitted to “Symbiosis University of Applied Sciences, Indore

As a Project report for the partial fulfillment of the award of degree of

**BACHELOR OF TECHNOLOGY**

**IN**

**SCHOOL OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY**

**Submitted To:** **Submitted By:**

Ms. Shruti Jain Jaydeep Patel (2020BTCS016)

Assistant Professor Ojas Sharma (2020BTCS019)

Tejas Bhati (2020BTCS042)

**Contents**

[INTRODUCTION 2](#_Toc2124650524)

[1.1 Introduction 3](#_Toc1899379953)

[1.2 Problem Definition 3](#_Toc1860430266)

[LITERATURE REVIEW 3](#_Toc449196965)

[PROPOSED METHODOLOGY 4](#_Toc1604769108)

[(BLOCK DIAGRAM OF THE WHOLE SYSTEM) 5](#_Toc757242090)

[DATA COLLECTION AND REFINEMENT 6](#_Toc698656693)

[4.1 Data Description 6](#_Toc1976351497)

[4.2 Data Collection Process 6](#_Toc1845504921)

[4.3 Data Refinement Requirements 7](#_Toc305504993)

[PROJECT IMPLEMENTATION AND CODE 7](#_Toc1891854531)

[RESULTS 10](#_Toc889110127)

[FUTURE SCOPE 15](#_Toc308196832)

[BIBLIOGRAPHY AND REFERENCE 15](#_Toc391090556)

# **INTRODUCTION**

## **1.1 Introduction**

There are several music streaming services and apps that use algorithms to generate playlists based on a listener's mood or activity. By using such services, a listener can easily access a playlist that matches their current state without manually selecting songs.

Another option is to use voice-activated assistants like Siri or Alexa to play music. This can be useful while performing some important task, as it allows you to play music without using your hands.

Ultimately, the key is to find a solution that allows an easy access to play music without distracting from the task at hand. By leveraging technology, we can enjoy the benefits of music without much compromising our productivity.

## **1.2 Problem Definition**

Music has the power to influence human emotions and moods. However, manually selecting and playing songs based on one's emotional state can be a risky while driving a car and time-consuming or disturbing while doing some work.

# **LITERATURE REVIEW**

Several research papers have proposed various approaches for creating smart music players that recommend songs based on the user's emotions or facial expressions. One such paper describes a music player called EMP, which incorporates emotion context reasoning within an adaptive music recommendation system. The system uses deep learning algorithms to identify the user's mood based on an image of their face and achieves high accuracy in classifying songs into different mood classes.

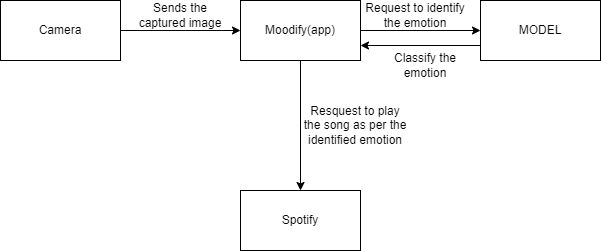
Another paper proposes a playlist generation system that uses facial expressions to automatically generate a playlist, reducing the computational time involved in obtaining results and the overall cost of the system. Several other papers describe similar systems that use computer vision technology to classify human expressions and play music tracks based on the present emotion detected. These systems employ the PCA algorithm and Euclidean Distance classifier to extract facial features and achieve moderate accuracy levels in recognizing expressions.

In addition, a paper proposes a smart music player that detects a person's mood based on the expression on their face using convolutional neural networks (CNN). The system collects multiple images to overcome the problem of facial expressions changing rapidly.

Lastly, another paper presents an algorithm that automates the process of generating an audio playlist based on the facial expressions of a user, reducing the time and labor involved in manual segregation of a playlist.

# **PROPOSED METHODOLOGY**

# **(BLOCK DIAGRAM OF THE WHOLE SYSTEM)**

****

The proposed methodology involves using a camera to capture an image of a person's face, sending the image to the Moodify application, which then classifies the emotion and requests to identify the emotion from the model. Once the emotion is identified, the application then requests to play a song as per the identified emotion from Spotify.

The use of a camera to capture a person's facial expression provides a visual input to the system, which is then used to classify the emotion of the person in the image. This classification process is carried out by the Moodify application, which makes use of a pre-trained emotion recognition model to classify the emotion in the image.

Once the emotion is identified, the Moodity application sends a request to Spotify to play a song that matches the identified emotion. Spotify then selects a song from its vast library of music that matches the identified emotion and plays it for the user.

This methodology has the potential to enhance the user's experience by automatically selecting and playing music that matches their current emotional state, without the need of manual input selection of songs. However, the success of this methodology depends on the accuracy of the emotion recognition model used by the Moodity application and the availability of songs that match the identified emotion in the Spotify library.

# **DATA COLLECTION AND REFINEMENT**

## **4.1 Data Description**

The FER\_2013 dataset is a collection of grayscale images of faces that are 48x48 pixels in size, placed in the folders labeled with one of seven possible facial expressions, i.e., anger, disgust, fear, happiness, sadness, surprise, or neutral. The dataset contains a total of 35,887 images, which are divided into three subsets: a training set of 28,709 images, a public test set of 3,589 images, and a private test set of 3,589 images.

## **4.2 Data Collection Process**

The images in the FER\_2013 dataset were collected from the Internet Movie Database (IMDb) using search queries for movie scenes that contained faces. The dataset was collected using a semi-automatic process, where initial images were selected using search queries, and then manual filtering was used to remove low-quality images.

After the initial collection, the images were labeled using an automatic face analysis tool that detected facial landmarks and inferred the presence of certain facial expressions. The tool used a combination of facial landmark detection, head pose estimation, and expression recognition algorithms to automatically label the images.

## **4.3 Data Refinement Requirements**

One of the main limitations of the FER\_2013 dataset is that the automatic labeling may not always be accurate. As a result, there is a need for manual refinement of the labels to improve the quality of the dataset. Additionally, the dataset does not include images of faces with glasses, facial hair, or other facial features that may affect expression recognition, which may limit the applicability of the dataset to certain real-world scenarios. Therefore, future work could involve collecting additional data that includes more diverse facial features and manual refinement of the labels to improve accuracy.

# **PROJECT IMPLEMENTATION AND CODE**

The given code implements a deep learning model called "Moodify" based on convolutional neural networks and is trained using the ImageDataGenerator class from Keras. The model is trained to recognize facial expressions/emotions from images and plays the music according to sentiment. It uses the FER2013 dataset, which contains facial images labeled with seven different emotions: Angry, Disgust, Fear, Happy, Neutral, Sad, and Surprise.

**#Moodify.ipynb**

# Importing libraries

from google.colab import drive

drive.mount('/content/drive')

import os

path = "/content/drive/MyDrive"

os.chdir(path)

import keras

from keras.models import Sequential, load\_model

from keras.layers import Conv2D, BatchNormalization, MaxPool2D, Flatten, Dense, Dropout

from keras.callbacks import EarlyStopping, ModelCheckpoint

from keras.preprocessing.image import ImageDataGenerator

import matplotlib.pyplot as plt

from PIL.Image import Image

import numpy as np

import pandas as pd

import zipfile

# Loading Dataset

from tensorflow.python import train

zip\_file = zipfile.ZipFile('/content/drive/MyDrive/Dataset.zip','r')

zip\_file.extractall('/content')

path = path="/content/Dataset"

train\_dir = path + "/FER\_2013/train"

test\_dir = path + "/FER\_2013/test"

train\_data = ImageDataGenerator(rescale = 1./255,

                                shear\_range = 0.2,

                                rotation\_range = 20,

                                zoom\_range = 0.2,

                                horizontal\_flip = True)

test\_data = ImageDataGenerator(rescale = 1./255)

train\_dataset = train\_data.flow\_from\_directory(train\_dir, target\_size=(48,48), batch\_size = 32, color\_mode="grayscale", class\_mode="categorical")

test\_dataset = train\_data.flow\_from\_directory(test\_dir, target\_size=(48,48), batch\_size = 32, color\_mode="grayscale", class\_mode="categorical")

# Model making

emotions = train\_dataset.num\_classes

moodDetector = Sequential()

moodDetector.add(Conv2D(filters=64, kernel\_size=(3,3) ,input\_shape=(48,48,1) ,activation='relu'))

moodDetector.add(BatchNormalization())

moodDetector.add(Conv2D(filters=128, kernel\_size=(3,3), activation='relu', kernel\_regularizer=keras.regularizers.l2(1e-4)))

moodDetector.add(MaxPool2D(pool\_size=(2,2)))

moodDetector.add(BatchNormalization())

moodDetector.add(Conv2D(filters=256, kernel\_size=(3,3), activation='relu', kernel\_regularizer=keras.regularizers.l1(1e-4)))

moodDetector.add(BatchNormalization())

moodDetector.add(Conv2D(filters=256, kernel\_size=(3,3), activation='relu', kernel\_regularizer=keras.regularizers.l2(1e-4)))

moodDetector.add(MaxPool2D(pool\_size=(2,2)))

moodDetector.add(BatchNormalization())

moodDetector.add(Conv2D(filters=512, kernel\_size=(3,3), activation='relu', kernel\_regularizer=keras.regularizers.l1(1e-4)))

moodDetector.add(BatchNormalization())

moodDetector.add(Conv2D(filters=512, kernel\_size=(3,3), activation='relu', kernel\_regularizer=keras.regularizers.l2(1e-4)))

moodDetector.add(MaxPool2D(pool\_size=(2,2)))

moodDetector.add(BatchNormalization())

moodDetector.add(Flatten())

moodDetector.add(Dense(1024, activation='relu', kernel\_regularizer=keras.regularizers.l2(1e-4)))

# moodDetector.add(Dropout(0.2))

moodDetector.add(Dense(512, activation='relu', kernel\_regularizer=keras.regularizers.l2(1e-4)))

moodDetector.add(Dropout(0.2))

moodDetector.add(Dense(emotions, activation='softmax'))

print(moodDetector.summary())

adam = keras.optimizers.Adam(learning\_rate=0.001)

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

# early\_stopping = EarlyStopping(monitor='val\_loss', patience=5, verbose=1, mode='min')

model\_checkpoint = ModelCheckpoint('mood\_detector.h5', save\_best\_only=True, monitor='val\_loss', verbose=1, mode='auto')

no\_of\_epochs = 20

history = moodDetector.fit(train\_dataset,

steps\_per\_epoch = train\_dataset.n//train\_dataset.batch\_size,

epochs=no\_of\_epochs,

# verbose=1,

validation\_data = test\_dataset,

validation\_steps = test\_dataset.n//test\_dataset.batch\_size,

callbacks=model\_checkpoint)

acc = history.history['accuracy']

val\_acc = history.history['val\_accuracy']

loss = history.history['loss']

val\_loss = history.history['val\_loss']

epochs = range(len(acc))

# Accuracy curve

plt.plot(epochs, acc, 'r', label='Training accuracy')

plt.plot(epochs, val\_acc, 'b', label='Validation accuracy')

plt.title('Training and validation accuracy')

plt.legend(loc='upper left')

plt.figure()

plt.show()

# Learning curve

plt.plot(epochs, loss, 'r', label='Training accuracy')

plt.plot(epochs, val\_loss, 'b', label='Validation accuracy')

plt.title('Training and validation accuracy')

plt.legend(loc='upper left')

plt.figure()

plt.show()

moodDetector.save("/content/drive/MyDrive/moodifyEngine.h5")

from keras.models import load\_model

import cv2

from google.colab.patches import cv2\_imshow

import numpy as np

label\_dictionary={0:"Angry",1:"Disgust",2:"Fear",3:"Happy",4:"Neutral",5:"Sad",6:"Surprise"}

moodDetector = load\_model("/content/drive/MyDrive/moodifyEngine.h5")

face\_detector = cv2.CascadeClassifier('/content/drive/MyDrive/haarcascade\_frontalface\_default.xml')

img = cv2.imread('/content/happy.jpeg')

# Preprocess image

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

faces = face\_detector.detectMultiScale(gray, 1.3, 5)

x,y,w,h = faces[0]

print(x)

cropped = gray[y:y+h,x:x+w]

resized = cv2.resize(cropped, (48, 48))

normalized = resized / 255.0

input\_data = normalized.reshape((1,48,48))

print(moodDetector.predict(input\_data))

# Make prediction

prediction = np.argmax(moodDetector.predict(input\_data), axis=-1)

# Print result

print("The predicted emotion is:", label\_dictionary.get(prediction[0]))

# Show image

# cv2\_imshow(img)

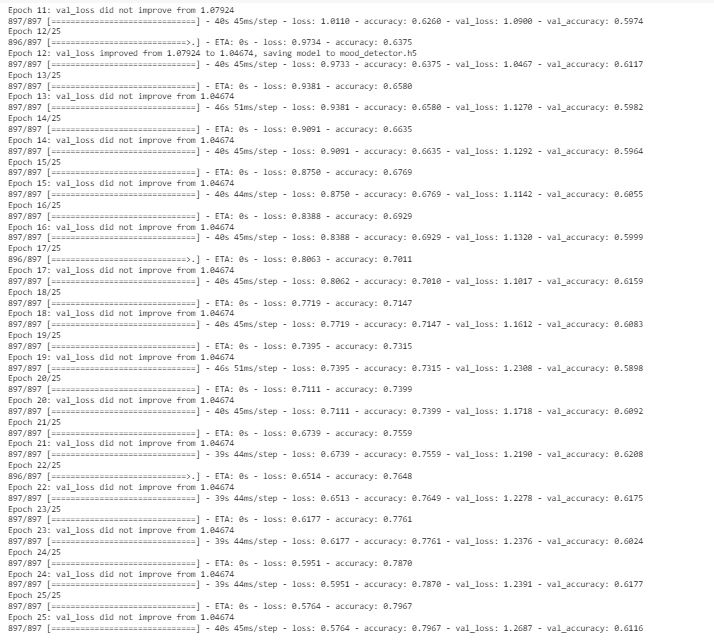
# cv2\_imshow(cropped)

cv2.waitKey(0)

cv2.destroyAllWindows()

# **RESULTS**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S.No.** | **Image** | **Actual Label** | **Correctly predicted by Model v1** | **Correctly predicted by Model v2** |
| 1 |  | Happy | 0 | 1 |
| 2 |  | Surprise | 0 | 0 |
| 3 |  | Angry | 0 | 0 |
| 4 |  | Neutral | 1 | 1 |
| 5 |  | Sad | 0 | 1 |
| 6 |  | Happy | 1 | 1 |
| 7 |  | Happy | 1 | 1 |
| 8 |  | Sad | 0 | 0 |
| 9 |  | Angry | 1 | 1 |
| 10 |  | Surprise | 1 | 1 |
| 11 |  | Neutral | 1 | 1 |
| 12 |  | Sad | 1 | 1 |
| 13 |  | Happy | 1 | 1 |
| 14 |  | Happy | 1 | 1 |
| 15 |  | Neutral | 1 | 0 |
| 16 |  | Angry | 0 | 1 |
| 17 |  | Neutral | 1 | 1 |
| 18 |  | Surprise | 0 | 0 |
| 19 |  | Angry | 0 | 0 |
| 20 |  | Happy | 0 | 1 |
| 21 |  | Neutral | 1 | 1 |
| 22 |  | Sad | 1 | 0 |
| 23 |  | Disgust | 0 | 0 |
| 24 |  | Disgust | 0 | 0 |
| 25 |  | Happy | 1 | 1 |
| 26 |  | Sad | 1 | 0 |
| 27 |  | Neutral | 1 | 1 |
| 28 |  | Angry | 0 | 0 |
| 29 |  | Surprise | 0 | 0 |
| 30 |  | Surprise | 0 | 0 |

Model V1

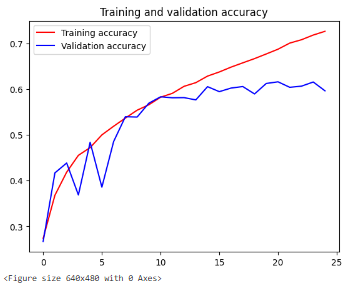


Fig - Accuracy curve

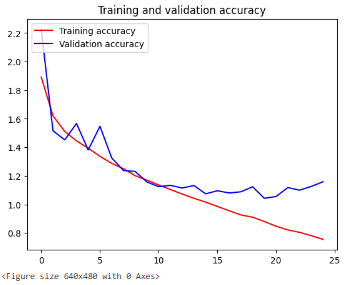
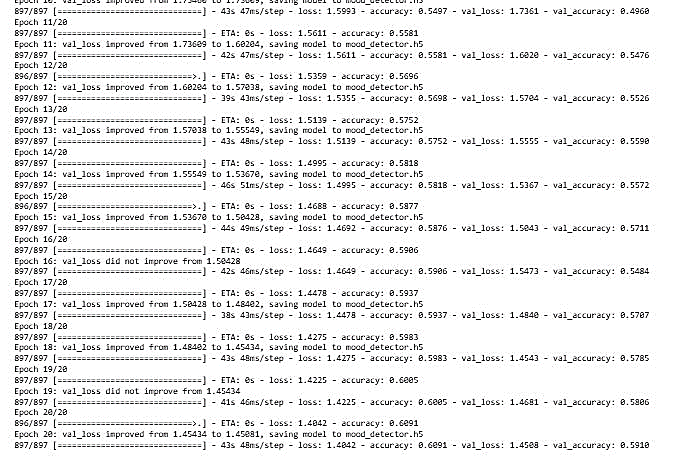


Fig 2 – Learning curve

Model V2

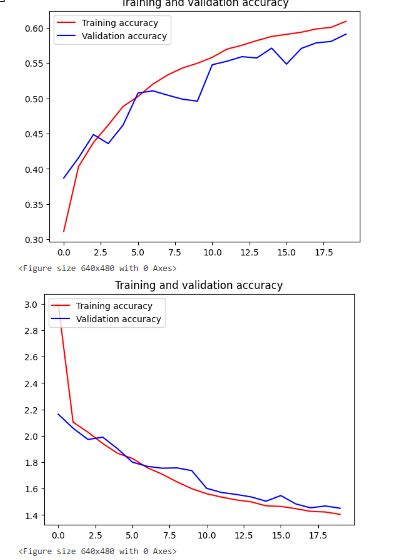
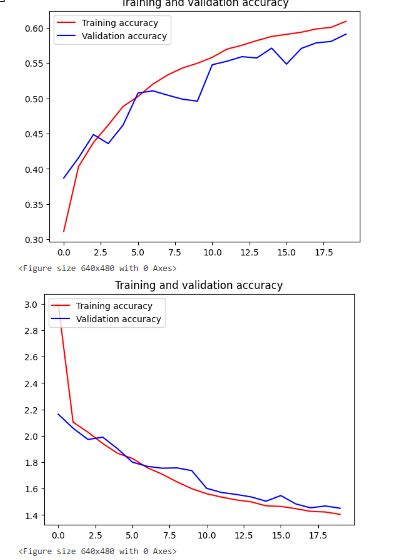


Fig 3 - Accuracy curve

Fig 4 – Learning curve

**CONCLUSIONS**

The Base model, i.e., the version 1 of the model for detecting the emotion is giving an accuracy of 50% for the real-world random images. On the other hand, the version 2 is giving an accuracy of 56% for the same dataset of the real-world images.

Therefore, the Model v2 which is fine tuned with the help of regularization to avoid overfit and getting closer to good fit is chosen for the deployment in the app.

# **FUTURE SCOPE**

1. Increasing the accuracy of the model up to more than 90%.
2. Making of customized data set which contains more than 7 expressions, i.e., Happy, Sad, Neutral, Disgust, Surprise, Fear and Angry.
3. Integrating it with the Spotify music app to play music online.
4. Playing an entire playlist in Spotify based on the certain expression.
5. Embedding with the IOT based system.

# **BIBLIOGRAPHY AND REFERENCE (APA Style)**

* Dhruba59. (2020). Music recommendation based on facial emotion recognition [GitHub repository]. https://github.com/Dhruba59/Music-recommendation-based-on-facial-emotion-recognition
* GeeksforGeeks. (n.d.). How to play a Spotify audio with Python? https://www.geeksforgeeks.org/how-to-play-a-spotify-audio-with-python/
* Oheix, J. (2019). Face expression recognition with deep learning [Code]. Kaggle. https://www.kaggle.com/code/jonathanoheix/face-expression-recognition-with-deep-learning
* Prajapati, N. (2020). Facial expression detection using machine learning in Python. Analytics Vidhya. https://medium.com/analytics-vidhya/facial-expression-detection-using-machine-learning-in-python-c6a188ac765f
* GitHub. (n.d.). Facial expression recognition repositories in Python. https://github.com/topics/facial-expression-recognition?l=python