

DAYANANDA SAGAR COLLEGE OF ENGINEERING

(An Autonomous Institute affiliated to VTU, Belagavi, Approved by AICTE & ISO 9001:2008 Certified)

Accredited by National Assessment & Accreditation Council (NAAC) with 'A' grade, Shavige

Malleshwara Hills, Kumaraswamy Layout, Bengaluru-560078.



Project Report On

“IMAGE MUSICALIZATION- USING IMAGE EMOTION ANALYSIS”

Submitted By

SOORYA VINOD- 1DS16CS108

SWATHI UTTARKAR-1DS16CS113

UMME KULSUM-1DS16CS116

MANSI R SETTY-1DS16CS129

[Eighth Semester B.E (CSE)]

Under the guidance of
Prof. PRASAD A M
Assistant Professor
Dept. of CSE
DSCE, Bangalore

**Department of Computer Science and Engineering
Dayananda Sagar College of Engineering
Bangalore-78**

**VISVESVARAYA TECHNOLOGICAL
UNIVERSITY**

DAYANANDA SAGAR COLLEGE OF ENGINEERING

Shavige Malleshwara Hills, Kumaraswamy Layout, Bangalore - 560078

Department of Computer Science & Engineering



CERTIFICATE

This is to certify that the project entitled **IMAGE MUSICALIZATION- USING IMAGE EMOTION ANALYSIS** is a bonafide work carried out by **Soorya Vinod [1DS16CS108]**, **Swathi Uttarkar [1DS16CS113]**, **Umme Kulsum [1DS16CS116]** and **Mansi R Setty [1DS16CS129]** in partial fulfilment of 8th semester, Bachelor of Engineering in Computer Science and Engineering under Visvesvaraya Technological University, Belgaum during the year 2019-20.

Prof. Prasad A M
(Internal Guide) Assistant
Professor, Department of
CSE, DSCE

Signature:.....

Dr. Ramesh Babu Vice
Principal & Head of
Department, CSE, DSCE

Signature:.....

Dr. C P S Prakash
Principal, DSCE

Signature:.....

Name of the Examiners:

1.....

2.....

Signature with date:

.....

.....

Abstract

Nowadays, everyone becomes an “artist” with the widespread use of digital cameras, capturing every aspect of their life by images to express their emotions and to share with their friends. When demonstrating images, accompanying appropriate music can make pictures vivid and bring people better feelings. Many people have frequently merged images into videos attached related to music, using professional softwares such as Premiere, Ulead Video Studio, and Movie Maker.

The objective of this project is to determine the emotion of any non-facial input image and map that image to a relevant music based on the classified emotion of the image and music of similar emotion. We also use a combined dataset of non-facial images for training that contains around 400-450 images for each class.

The proposed system uses Convolution Neural Network (CNN) model to achieve the objective. CNN is a branch of Deep Neural Networks, specifically useful for image processing. Input to the system is an image from the gallery and the resulting output is a music and the detected emotion of the image.

Acknowledgement

We are immensely satisfied with the successful completion of this project on Image Musicalization using Image Emotion Analysis.

We would like to take this opportunity to express our gratitude to **Dr. C P S Prakash**, Principal of DSCE, for making the necessary facilities of the institution available for our utilization.

We are also very grateful to our respected Vice Principal and the HOD of Computer Science & Engineering, DSCE, **Dr. Ramesh Babu D R**, for his support and encouragement in the department.

We are immensely grateful to our project guide, **Prof. Prasad A M**, Assistant professor, CSE, DSCE for his valuable guidance and support throughout. We are extremely thankful for all the encouragement and guidance that he has showered upon us during every stage of the project.

We would then like to thank our project coordinator **Dr. Vindhya P Malagi** Associate Professor, CSE, DSCE for the valuable guidance and support.

We are also thankful to all the faculty and staff members of our department for their kind cooperation and help.

Lastly, we would like to express our hearty appreciation towards our classmates and our family for providing us with constant moral support and encouragement.

SOORYA VINOD- [1DS16CS108]
SWATHI UTTARKAR-[1DS16CS113]
UMME KULSUM-[1DS16CS116]
MANSI R SETTY-[1DS16CS129]

List of Figures

4.3 Use Case Diagram.....	10
5.1 System Design.....	11
5.2 Block Diagram of The Proposed Solution.....	12
5.3 Data Flow Diagram.....	13
5.4 SequenceDiagram.....	13
5.5 Filters in CNN.....	15
5.6 CNN Structure.....	15
6.1 Directory Structure.....	16
6.2 Image Dataset Folders.....	17
6.3 Music Dataset Folders.....	18
7.1 Testing Snapshot: Anger.....	21
7.2 Testing Snapshot: Joy.....	21
7.3 Testing Snapshot: Surprise.....	22
7.4 Testing Snapshot: Sadness.....	22
8.1 GUI: demo.....	23
8.2 GUI: Image Upload.....	24
8.3 GUI: Predict emotion.....	25
8.4 GUI: histogram.....	25
8.5 Output: Play Music-Surprise.....	25
8.6 Output: Play Music-Fear.....	26
8.7 Output: Play Music-Joy.....	27
8.8 Output: Play music- Angry + sad.....	27

CONTENTS

Introduction	1
1.1 Overview	1
1.2 Organization of the Project Report	1
Problem Statement and Proposed solution	3
2.1 Problem statement	3
2.2 Existing Systems	4
2.3 Proposed Solution	4
Literature Survey	5
Requirements and Specifications	9
4.1 Software Requirements	9
4.2 Hardware Requirements	9
4.3 System Requirements	10
System Design and Architecture	11
5.1 System Overview	11
5.2 System Architecture	12
5.2.1 System block diagram	12
5.2.2 Data flow Diagram	12
5.2.3 Sequence Diagram	13
5.3 Convolutional Neural Networks	14
Implementation	16
6.1 Implementation Overview	16
6.2 Datasets	17
6.2.1 Image dataset	17
6.2.2 Music dataset	18
6.3 Implementation Details	18
Testing and Validation	20
7.1 Accuracy Table	20
7.2 Testing	20
Results and Output	23
8.1 GUI	23
8.2 Outputs	26
Conclusion and Future Enhancements	28
9.1 Conclusion	28
9.2 Future Enhancements	29
References	30
Appendix: Code Snippets	31

Chapter 1

INTRODUCTION

1.1 Overview

Human face-based emotion detection and analysis have evolved over the years but there has been very little work on abstract image emotions. By abstract image emotions we mean emotions of non-facial pictures.

In an image with human faces, we look for the facial features such as the curve of the face, eyes, orientation and others to decide the emotion of that human, which can then be generalized to be the emotion associated with that picture. But when it comes to recognizing emotions out of non-facial images, we have to look at different aspects such as color, texture, contrast, tone, harmony, gradation etc. Whenever we feel happy, we look for brighter images with lots of colors and vibrance whereas when we feel sad, we generally look for images which correspond to less vibrance and are dull.

These are just the immediate responses we give as to what we look at in an abstract image to know its emotions, but there are a lot of other aspects that we unknowingly consider before declaring the emotion of an image. So, this project aims in the direction of identifying the emotions that a person is experiencing, playing a music that falls under that emotion category and could help to create a mood swing model as a next generation application.

1.2 Organization of the Project Report

The project report is organized as follows: In Chapter (2), we discuss the problem statement and our solution to the problem. The Chapter that follows i.e. chapter (3) consists of the details on the literature survey of the papers referred to get to know the problem statement and the proposed solution. This followed by Chapter (4) we specify the system requirements for the project. In Chapter (5), we present the System Overview in the form of Data flow diagram and sequence diagram. The next chapter, chapter (6) gives the details about the implementation of the proposed system. Chapter 7 deals with the testing of the system and their results. In Chapter (8), we discuss

the results and the overall output of the system. The chapter (9) concludes the paper along with mention of the Future Enhancements. This is followed by a reference section that details the references made during the development of the system. The other supporting information and code snippets are gathered in the Appendix.

Chapter 2

PROBLEM STATEMENT AND SOLUTION

2.1 Problem Statement

There is a huge set of image data being generated throughout the world by each passing day and most of it is accumulated without any use. Big data analysis helps in utilizing numeric data more than image data.

Another less explored area in image processing and analysis is the emotion analysis of an abstract image. Emotion detection and analysis based on human facial features have evolved over the years but there has been very little work on abstract or non-facial image emotions. Analysis of abstract image emotions could find an application in places like art galleries where emotions behind abstract art masterpieces could be found.

Playing appropriate music when watching images can make the images more realistic and bring people into their intrinsic world. When demonstrating images, accompanying them with appropriate music can make pictures vivid and bring people better feelings. People over recent years have been frequently merging images into videos attached to appropriate music, using various professional softwares. We hence investigate the problem of automatic image musicalization based on emotions.

So, our goal with this project is to develop an application that can identify the emotion from abstract images and play corresponding music with the help of image processing models using Convolutional Neural Network.

This attempt of Image Musicalization could help visually impaired persons by providing them a means to analyze images by hearing them. It could also help hearing impaired persons by acting as a means to feel the music by looking at relevant/similar images.

2.2 Existing Solutions

The previous works on image emotion analysis focuses mainly on elements-of-art based low-level visual features, which are vulnerable to the arrangements of elements. These solutions extract the emotions of an image by acquiring low-level features of an image that are colors, lines and shapes. These are important but not enough to make a better prediction of emotions.

There are some solutions that research on using the principles-of-art features which are emphasis, balance, harmony, movement, gradation and variety. These solutions provide novel algorithms without appropriate baselines. Also, most of these proposed works make use of Machine Learning Classification algorithms for modelling. Usage of Convolution Neural Networks for Emotion detection in images and then using it to generate music is a novel idea.

2.3 Proposed Solution

The solution we propose for the above given problem is called Image Musicalization which is an application basically for entertainment purposes.

The goal is to develop an application that can identify abstract images emotion and play corresponding music with the help of image processing models using Convolutional Neural Network. We initially load all the images from the datasets in the database and train the Convolutional Neural network over each image. The datasets provided for training the network shall be dynamic in nature and can anytime be increased with more images. The model is iterated over various epochs to achieve better accuracy. A test image is provided by the user using a GUI with buttons and the image is tested over the trained network to give a satisfactory result.

The obtained result would be a music of respective emotion being played and the emotion detected from the image along with the percentages of individual emotions it contains. The image can also be classified as a combination of two emotions, if found ambiguous.

Chapter 3

LITERATURE SURVEY

[1] Representing Pictures with Sound

Authors: Edward m. Schaefer

In this paper series of sound audio sounds are used to represent the image. Initially the images are divided into 4X4 arrays of sound elements. For each sound element assign intensity and an audio sound. The audio sound is assigned to the position of sound element. From the sound element's part of the image intensity is computed. The audio for the sound element is the audio sound for the sound element being played at that computed intensity. On combining 16 audios generated for sound elements the audio for image is obtained. One finding says that mostly images generate a lot of noise which is hard for a person to find small differences in between audios when more noise is present. This can be brought down by constraining the amount of noise generated is work against the change between frames instead of individual frames. This leads to a less noisy result.

[2] On Shape and Computability of Emotions

Authors: Xin Lu, Poonam Suryanarayan, Reginald B. Adams, Jr. Jia Li, Michelle G. Newman, James Z. Wang

In this paper, it is mentioned and explored how the roundness and complexity of shapes are the two important factors to understand an image's emotions. According to this paper, valence, arousal and dominance are the three dimensions that affect any image and these three dimensions have been defined accordingly. SVM regression with RBF kernel is used to model the valence- arousal values and it is found from the MSE values that visual shapes provide use in understanding valence instead of color, texture and composition. Line segments, angles, orientation, continuous lines, curves are the shape parameters used in this paper to capture emotion. The paper also stresses on why and how the IAPS dataset is important to use in this process.

[3] Predicting Discrete Probability Distribution of Image Emotions

Authors: Sicheng Zhao, Hongxun Yao, Xiaolei Jiang, Xiaoshuai Sun

In this paper, focus is on predicting the probability distribution of categorical image emotions and the experiments that are carried out, demonstrate the superiority of categorizing images using principles-of-art over state-of-the-art approaches. There are two emotion representation models- categorical emotion states (CES) and Dimensional Emotion spaces (DES) of which prediction in CES form is highlighted here. They have introduced three simple baselines and then the main algorithm to predict image emotions. Global weighting, K-nearest Neighbor Weighting and SoftMax Regression are the three baseline models and Shared Sparse Learning is the algorithm used which is optimized by Iteratively Reweighted Least Squares. This paper introduces us to the two emotion representation models and the principles-of-art approach which shall be used by us for the development of our project and hence this paper was considered relevant.

[4] Exploring Principles of Art-Based Emotion Features for Image Emotion Recognition

Authors: Sicheng Zhao, Yue Gao, Xiaolei Jiang, Hongxun Yao, Tat-Seng Chua, Xiaoshuai Sun

Most of the previous work on image emotion analysis had used “elements of art based low level visual features” (EAEF). But EAEF has weak links to emotions i.e., EAEF are vulnerable and are also not invariant to different arrangements of elements. This results in poor performance for emotion recognition for the given image. As low-level features include color, space, value, texture, line, space and form, these low-level features are not capable of representing high level emotions. Therefore, we prefer PAEF (principles of art-based emotion features). They include the parameters like emphasis, balance, harmony, movement, gradation and variety. They are more semantic than EAEF and that symmetry and variety are human understandable when compared to texture and line. PAEF effectiveness is evaluated by two ways namely Affective image classification and Emotion score prediction. Affective image classification depends on emotion categories such that different combinations of principles express different emotions. In Affective image classification balance and harmony positive emotions often whereas variety and emphasis classify all 8 categories of emotions. Therefore, the experimental results on Affective image classification have demonstrated that the performance of PAEF is superior.

[5] Emotion based Image Musicalization

Authors: Sicheng Zhao, Hongxun Yao, Fanglin Wang, Xiaolei Jiang, Wwei Zhang

When watching images, playing appropriate music can make the images vivid and bring people into their world of intrinsic. They suggest musicalizing objects based on their emotions in this article. Most previous works on the study of object emotion primarily used low-level visual features focused on elements of art that are sensitive to element arrangements. Here we propose to extract graphical features to identify object emotions, motivated by the idea of principles-of-art. In order to enrich the expressive capacity, emotion modeling is implemented with a dimensional perspective. IAPS data collection studies reveal the superiority of the proposed method relative to state-of-the-art emotion regression approaches. To musicalize these images, the music in the MST dataset with approximate emotions to the known picture emotions is selected. They use the technique of regression and the process of extraction to estimate the music's emotions. The findings of the user study show the effectiveness and success of the process of object Musicalization. The images used in this paper have disordered emotions and hence they use the “Locally consistent, globally choppy” principle to readjust the showing order of the images. The algorithm goes like, first decide the first image to be shown or randomly select one, then the music with the most estimated emotions to be played with the emotions of the image. Since music is sequential, they select the images with the most approximate emotions with the display image to be displayed until the music's over or the similarity of emotions is greater than the threshold or users stop the process. Next, they randomly select an unshown image to be shown and circulate until the unshown picture collection is empty or users avoid the algorithm of Musicalization.

[6] Building Emotional Machines: Recognizing Image Emotions through Deep Neural Networks

Authors: Hye-Rin Kim, Yeong-Seok Kim, In-Kwon Lee, Seon Joo Kim

In this paper, the authors focus on two high level features of any input image, object and background of the picture, and explain that the semantic information of images is a good signal to use for predicting emotion. Object is one of the important parts for defining an image, and experiments here show that there is a huge correlation among the object and the emotion of images.

the combination of different levels of features, feed forward deep neural network based on emotions is built that gives an output with emotion values of an input image.

[7] Building a Large-Scale Dataset for Image Emotion Recognition

Authors: Quanzeng You, Jiebo Luo, Hailin Jin and Jianchao Yang

In this paper, the authors introduce a new data set, which is 30 times as large as the today's largest publicly available image emotion data set. They collect a huge amount of weakly emotion labeled images and then manually label these images one by one to obtain a relatively strongly labeled data set of images, which in turn makes using CNN for visual emotion analysis possible. This data set is very much useful for research on visual emotion analysis

Chapter 4

SYSTEM REQUIREMENTS AND SPECIFICATION

4.1 Hardware Requirements

- System : Intel i3 and above.
- RAM : 4 GB or more.
- GPU : NVIDIA 660m and above. (optional)
- Any system with above configuration or higher level

4.2 Software Requirements

- Operating system : Windows XP ,7 ,8 or 10
- Coding Language : Python
- Software : Anaconda
- IDE : Spyder Notebook

4.3 System Requirements

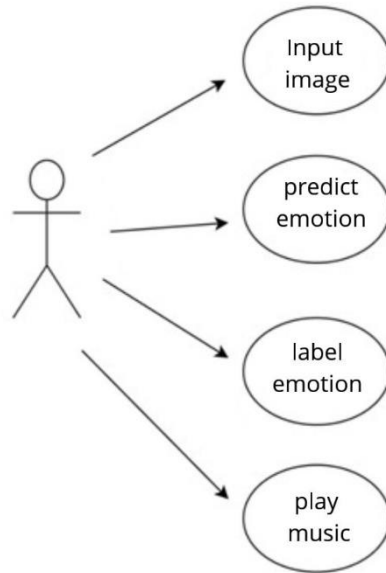


Fig.4.1

The system shall be able to:

- Accept an image input in the form of a JPEG or PNG.
- Predict the percentage of each emotion for the given image.
- Label the image with the emotion having the highest percentage.
- Play a music of the same emotion as the image is predicted to be.

Chapter 5

SYSTEM DESIGN AND ARCHITECTURE

5.1 System Overview

The system “design” is defined as the process of applying various requirements and permits its physical realization. Various design features are followed to develop the system; the design specification describes the features of the system, the opponent or elements of the system and their appearance to the end-users.

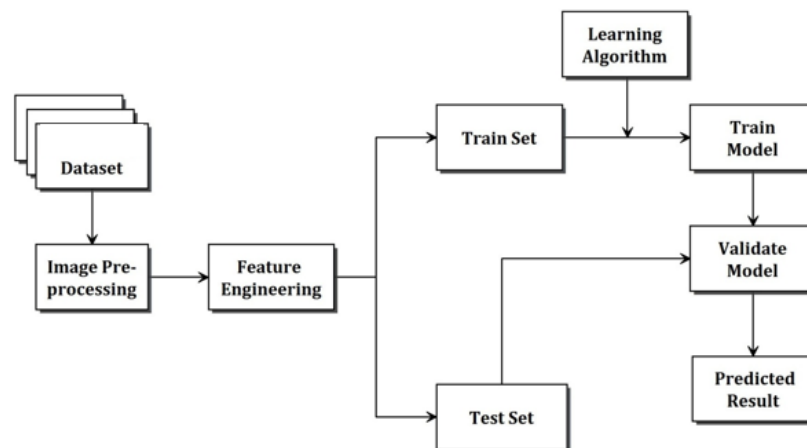


fig.5.1

The system overview as shown in figure 5.1 has the following modules working together:

- The Image Dataset that is fed for the network.
- Image preprocessing module where image is converted into array.
- The CNN models.
- Training the model with existing databases.
- Testing the model with real time inputs.
- UI where the resulting music is played.

5.2 Software Architecture

5.2.1 System Block Diagram

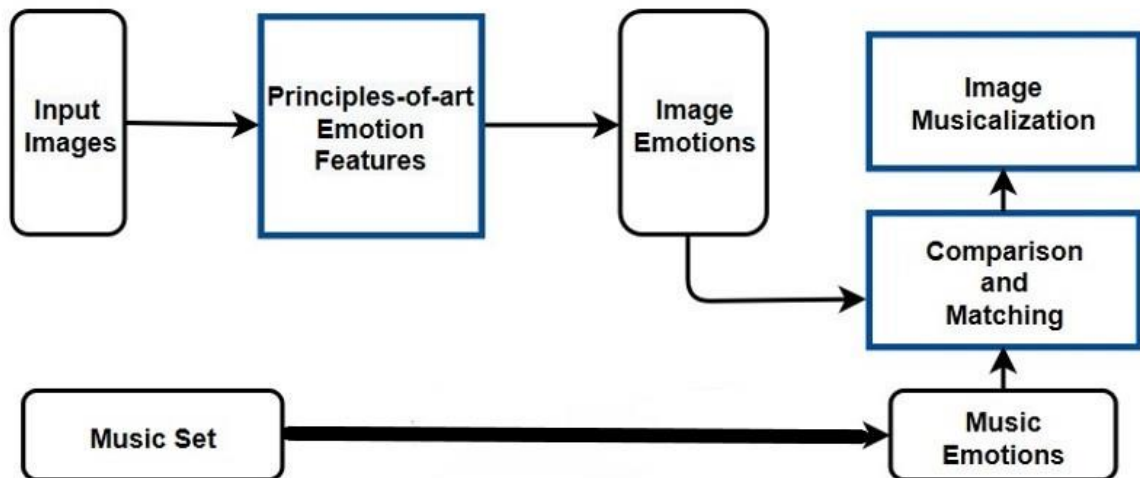


fig.5.2

The system block diagram shown in Figure 5.2 is explained as follows:

- User inputs image using system UI.
- Features of image are detected and emotions are labelled.
- These emotions are compared with the music dataset.
- Chosen music is produced as the output.

5.2.2 Data Flow Diagram

Data flowchart or Data Flow Diagram is a graphical representation of the "flow" of information in a system. This is a preliminary step that is used to create a summary of the system which may later be elaborated. The figure 5.3 represents DFD for our system.

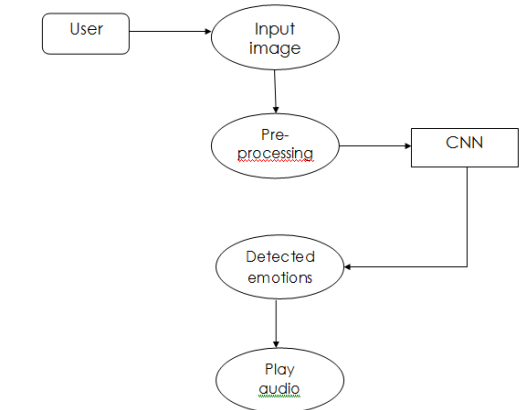


fig.5.3

5.2.3 Sequence Diagram

Sequence diagram is an interaction diagram that shows how processes operate with each other and in what order. A sequence diagram displays the object interactions that are arranged with time sequence, hence the name sequence diagram. A sequence diagram depicts objects and the classes involved within the scenario. Therefore, sequence of the messages being exchanged between the objects needed to hold out functionality of scenario.

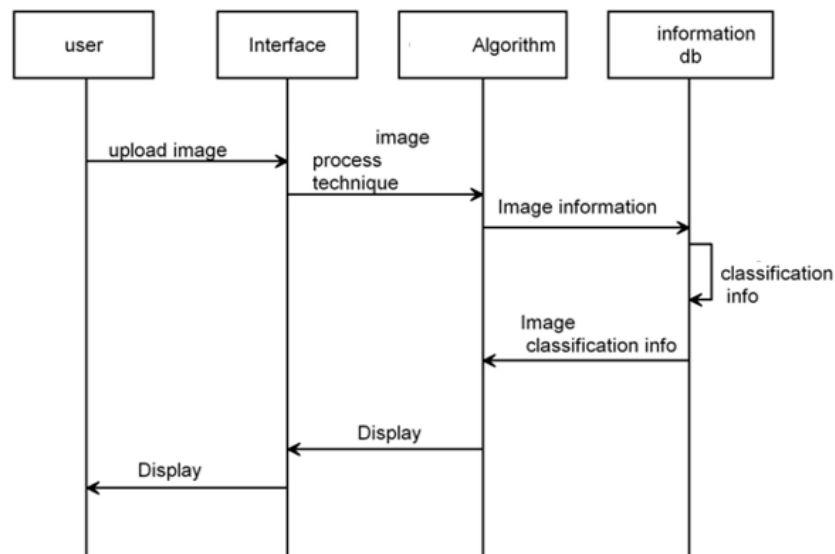


fig.5.4

Sequence diagrams at times are also called as event scenarios or as event diagrams. A sequence diagram is displayed with vertical and horizontal lines. The figure 5.4 depicts a sequence diagram for the scenario of a random input image by a user. The objects involved in this sequence diagram are user, interface, algorithm/trained network and the database used for training the network. When a user uploads an image, the GUI directs the image data to the trained model. Here, the image is preprocessed and classified into an emotion label by comparing the input image features to the classification information from the datasets and displays the detected emotion accordingly. The model also directs the UI to play a particular song from the database of music.

5.3 Convolutional Neural Network

CNN refers to the special architecture of the ANN (artificial neural networks). It makes use of some features of the visual cortex. Image classification is the most popular use among other uses of this architecture.

In a Convolutional Neural Network, the image is passed through a series of convolutional, nonlinear, pooling layers and fully connected layers, and then generates the output.

The Convolution layer is always the first. The image (matrix with pixel values) is entered into it. Imagine that the reading of the input matrix begins at the top left of the image. Next software will select the lower or smaller matrix there, that is termed as filter ("neuron" or "core"). The task of the filter is to multiply its values with the 'original pixel' values. All these multiplications are summed up (Figure 5.5). One number is obtained in the end. Now that the filter has read the picture only in the 'upper left corner', it moves further right by 1 unit performing a similar operation by moving further. After the filter is passed across all positions, the matrix is obtained, but lower or smaller than that of the input matrix. The network consists of several convolution layers mixed with nonlinear and pooling layers.

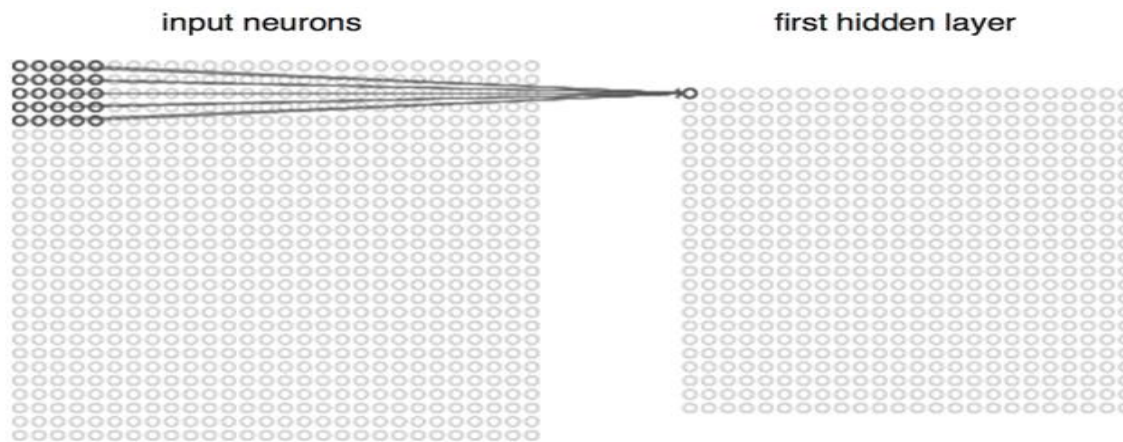


fig.5.5

The layer called nonlinear layer is added after each convolution operation. It has a function called activation function, which defines 'nonlinear property'. Without this nonlinear property a network would not be sufficiently intense and the network will not be able to model the response variable (as a class label).

The pooling layer follows the nonlinear layer. It works with the width and the height of the picture and then performs a down sampling operation on width and height of image. As a result of this the image volume is decreased. This indicates that if some of the features for example "boundaries" has been already identified within the previous convolution operation, then the detailed picture will no longer be needed for the further processing, and it is compressed to less detailed images.

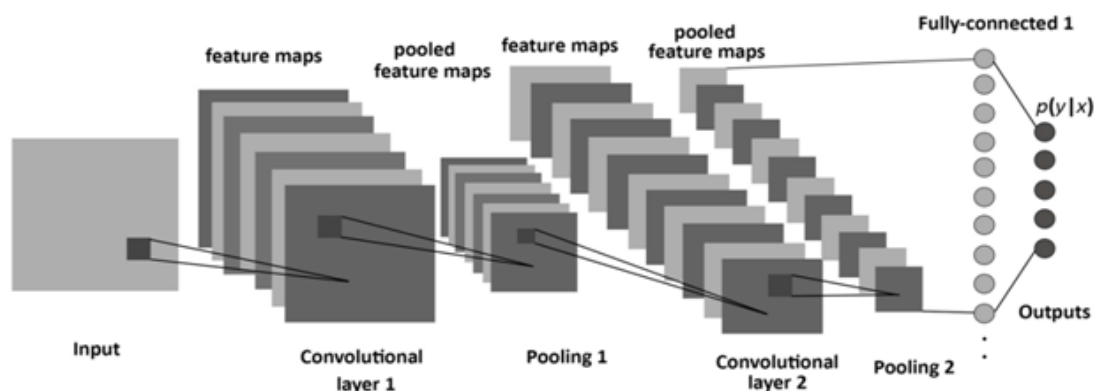


fig.5.6: Post completion of a series of convolutions, the 'nonlinear' and the 'pooling layers', it is necessary to attach fully connected layers. This fully connected layer will take the output information from the convolutional networks. Now attaching a FC layer (fully connected) to the end of the network results in an "N-dimensional vector". Here N is the number of classes where the model selects the desired class.

Chapter 6

IMPLEMENTATION

6.1 Implementation overview

Implementation is that stage of the project where its theoretical design is being converted into a working system. The crucial stage in achieving the main system that will work efficiently and effectively. The major goal of the implementation stage is to interpret the design of the system into the code using the suitable programming language.

6.1.1 Organization of files

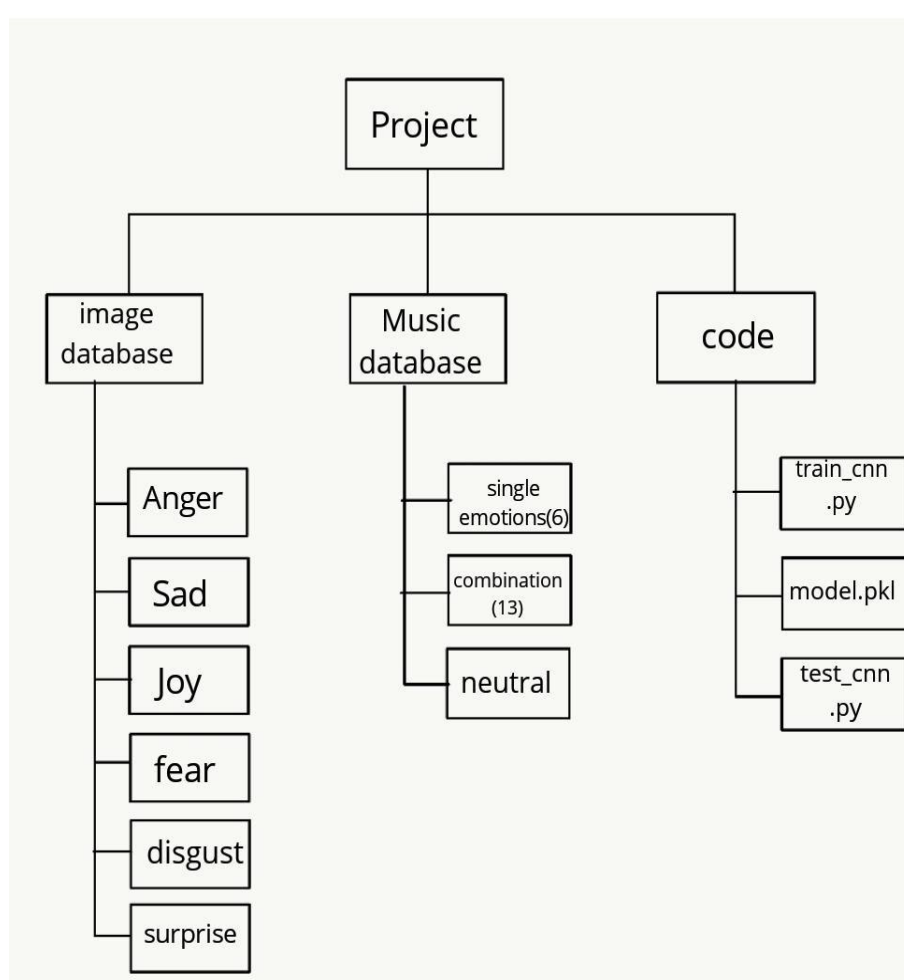


Fig. 6.1

6.2 Datasets:

The project requires two datasets i.e., image files and music files. The image dataset is used to train the CNN model to accuracy. The music dataset is used to play the audio files once the model predicts the emotion label for the input image.

6.2.1 Image dataset:

The image dataset we use here contains a total of 2400 images and is a combination of two different datasets namely emotion6 and abstract. The emotion6 dataset is collected by the Advanced Multimedia Processing Laboratory at Cornell University [8] from Google and Flickr to study dataset bias in visual emotion recognition. It contains around 1500 images classified into six emotion categories: anger, disgust, fear, joy, sadness, and surprise. Each of these folders approximately contain 330 images.

Additionally, we added 70 more images in each of the emotion categories from the abstract paintings dataset as recommended in [4], [5] and [7]. The abstract paintings database contains abstract art paintings and are a great source of data required for our project.







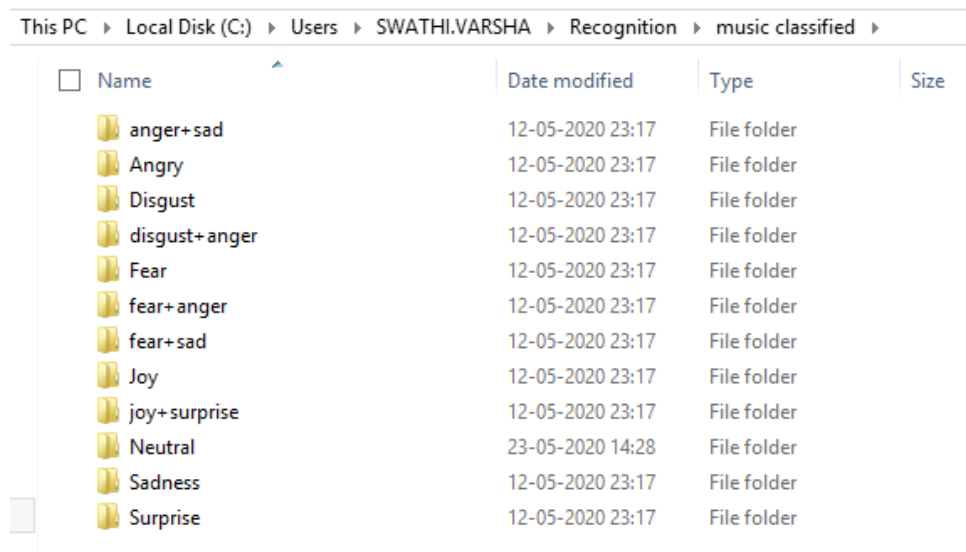
This PC ▸ Local Disk (C:) ▸ Users ▸ SWATHI.VARSHA ▸ Recognition ▸ DATABASE ▸		
<input type="checkbox"/> Name	Date modified	Type
 anger	12-05-2020 23:08	File folder
 disgust	12-05-2020 23:08	File folder
 fear	12-05-2020 23:08	File folder
 joy	12-05-2020 23:08	File folder
 sadness	12-05-2020 23:08	File folder
 surprise	12-05-2020 23:08	File folder

Fig.6.2

6.2.2 Music dataset

The music dataset was segregated manually with the help of peer students to detect the emotions and classify in accordance with the image database. The source for the accurate segregation of music files into different emotion labels is mentioned in [9], which were then peer validated. The music dataset also has six emotions similar to the image dataset, anger, disgust, fear, joy, sadness, surprise and an additional neutral folder. It is further divided by using the combinations of the emotions.



Name	Date modified	Type	Size
anger+sad	12-05-2020 23:17	File folder	
Angry	12-05-2020 23:17	File folder	
Disgust	12-05-2020 23:17	File folder	
disgust+anger	12-05-2020 23:17	File folder	
Fear	12-05-2020 23:17	File folder	
fear+anger	12-05-2020 23:17	File folder	
fear+sad	12-05-2020 23:17	File folder	
Joy	12-05-2020 23:17	File folder	
joy+surprise	12-05-2020 23:17	File folder	
Neutral	23-05-2020 14:28	File folder	
Sadness	12-05-2020 23:17	File folder	
Surprise	12-05-2020 23:17	File folder	

fig.6.3

6.3 Implementation details

Here, we are implementing using the python programming language. Using Spyder notebook as our ide. The GUI used is also coded using python.

First to train the model, we are using a convolution neural network (CNN) model, which is a deep learning algorithm that can take in an input image, assigns importance to various aspects/objects of the image to be able to differentiate one from the other. CNN models help us distinguish the emotions in the images using the colors as mentioned in 5.3.

The model is trained using the image datasets as mentioned previously. The model first extracts

the images from the dataset and the emotions of these images are found using CNN based on the different colors of these images. To increase accuracy, we have trained it multiple times by giving different batch and epoch sizes. The training code is given in Appendix.

The CNN model is trained using various libraries, the major one being keras API. Keras is an open source python library designed for fast experimentation of deep Neural Networks. It is used over a TensorFlow environment. Another important module imported is the OpenCV library which is essentially needed for any image processing manipulations. Pickle is used to load and dump the model created. To render the GUI Tkinter is used.

The first step in training involves loading the database and dividing it into batch sizes. Batch size specifies the number of images loaded at a time.

With the database loaded, the next step is preprocessing the image. The image is converted into an array of integer values. Preprocessing also involves rotation of the image in all angles before sending it to the network.

The feed forward neural network consists of an input layer, various hidden layers and an output layer. Here, we use 2D convolutional layers, a ReLU layer and end with a SoftMax layer. These layers are repeatedly iterated within the network until maximum accuracy is achieved. A classification report is generated with accuracy, loss, valence-loss and is printed at the end of every epoch.

Chapter 7

TESTING AND VALIDATION

The model was trained by giving the image dataset as the input. The accuracy of the model varied as we changed the batch and the epoch sizes. The best accuracy of 82.22% was acquired when the batch size was 30 and the epoch size was 25. Once the model is trained then it can be tested by giving an image path to check the emotions of images.

7.1 Accuracy table:

trial	epoch	batch size	image size	accuracy
1	15	10	256 x 256	82.19
2	20	10	256 x 256	82.00
3	20	10	256 x 256	80.27
4	25	20	256 x 256	82.04
5	25	30	256 x 256	82.22

Table 1

7.2 Testing

Once the model is trained rigorously for various combinations of epochs and batch size, it is now ready to be tested on a test image file. So, before we implement the UI, we test the network over various inputs from the dataset. The test code loads the model that was created and trained, and classifies the given image into an emotion class. The extracted emotions are represented in the form of a histogram. We can see the emotion that is labelled to the input image.

Then, a music file is selected at random using the random function from the classified emotion's folder in the music database. This music file gets played as the output. The following figure 10 shows how an image from the 'anger' folder is given as input by the

admin manually. The histogram shows the percentage of each emotion detected in that image and we can see that the ‘anger’ label has the highest percentage.

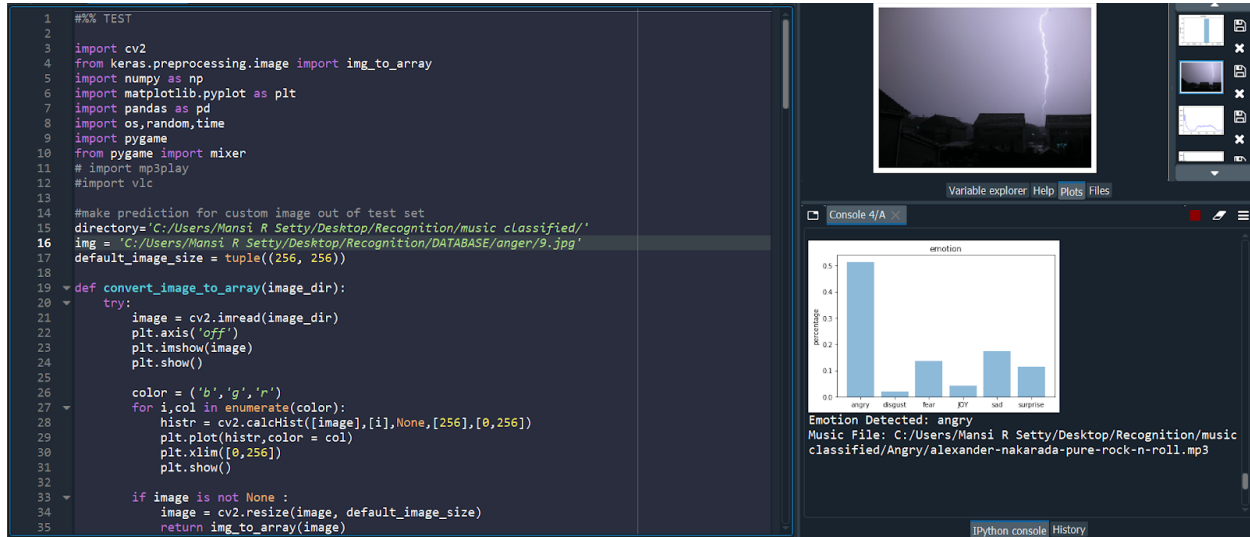


fig. 7.1: Anger

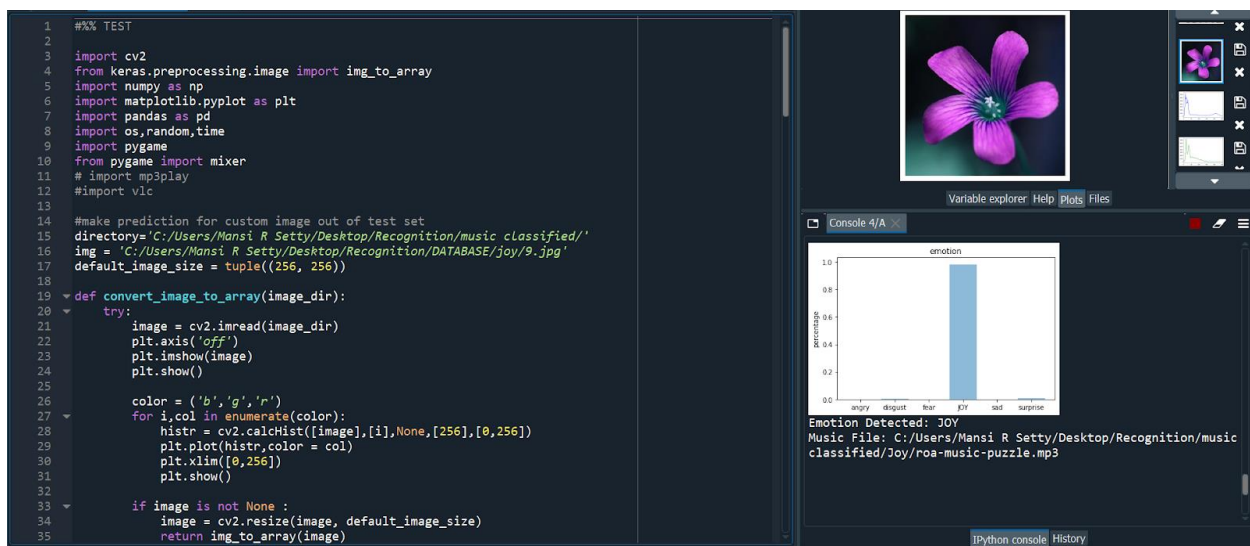


fig. 7.2: Joy

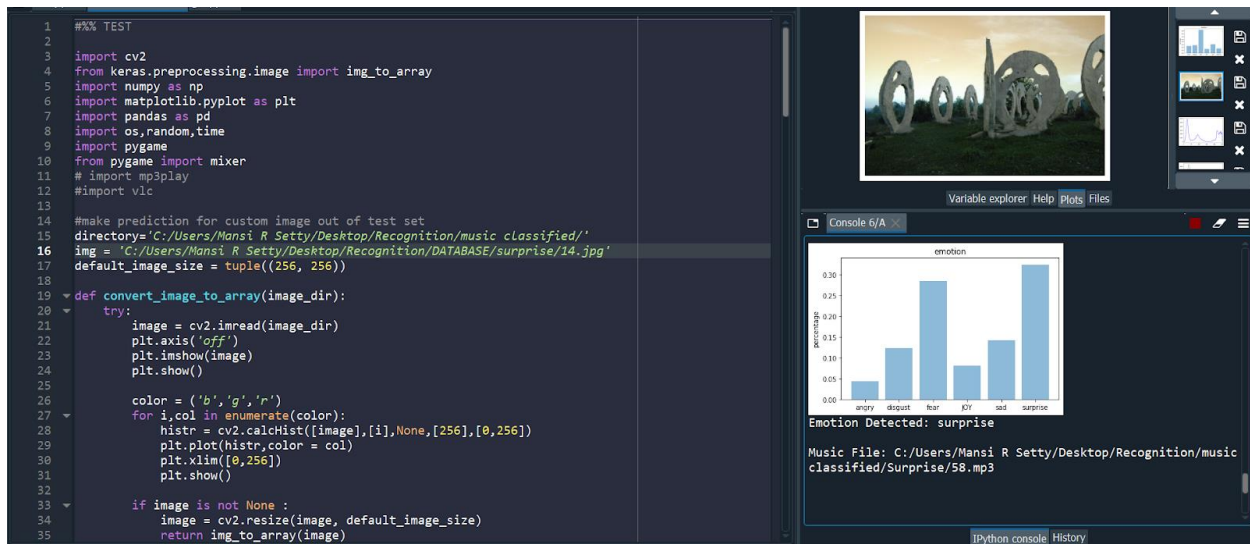


fig. 7.3: Surprise



fig. 7.4: Sadness

Chapter 8

RESULTS AND OUTPUT

8.1 GUI

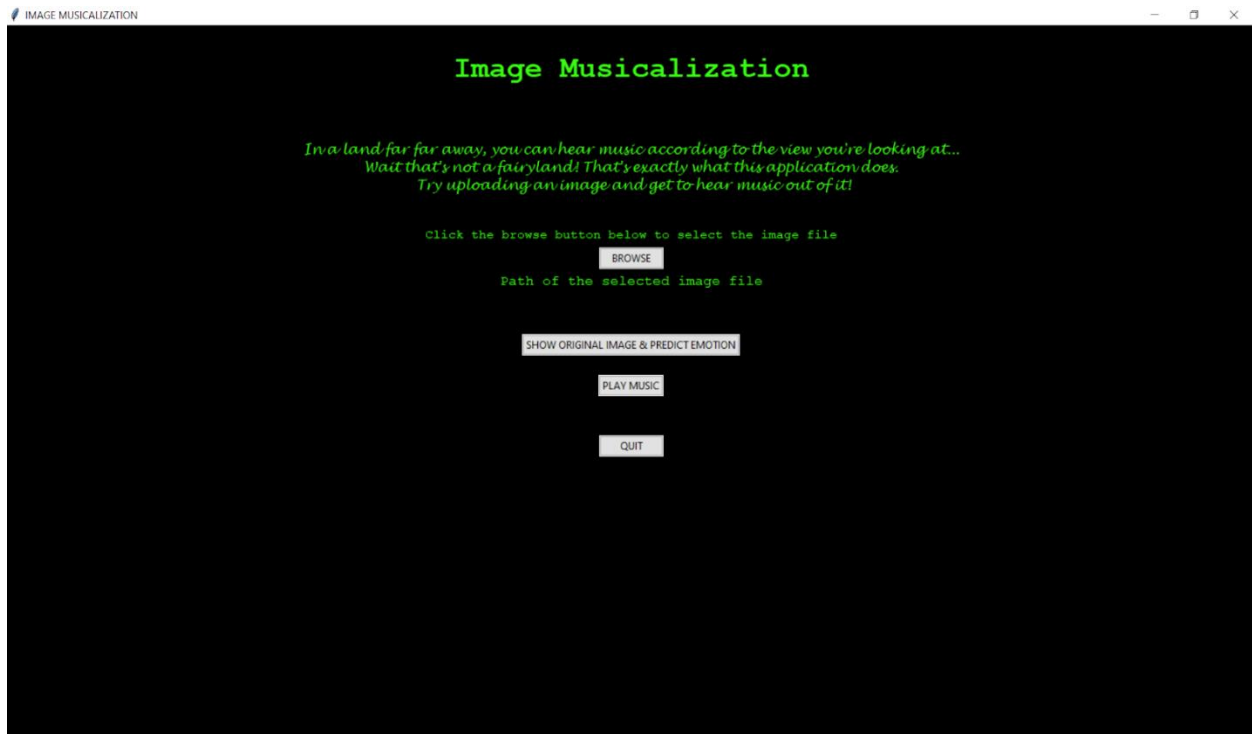


fig.8.1: This is the first view of the GUI for the image musicalization application.

The functions of the buttons:

1. Browse: it lets the user browse and choose an image from the PC.
2. Show original image & prediction: this shows the image that was uploaded after it has extracted the important part of the image to get optimal results. It also shows the histogram which shows the emotions of the image.
3. Play music: this plays the music in the GUI.
4. Quit: this is just to quit the application and exit.

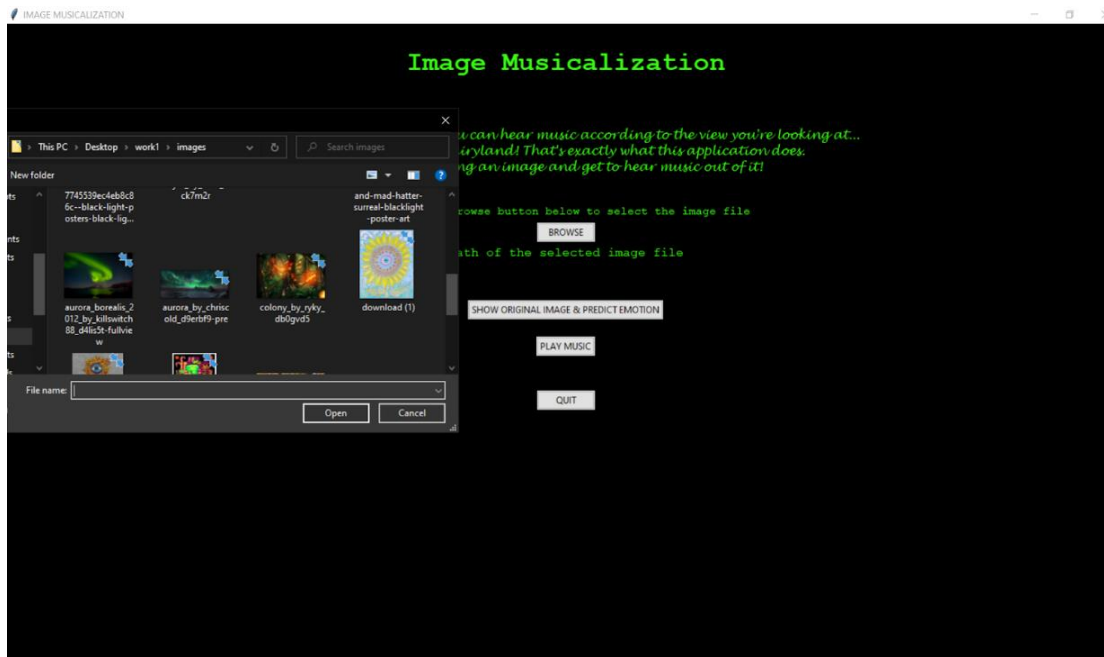


fig.8.2: Here, we can see the browsing of the image from any of the computers directories the user wishes to choose from.

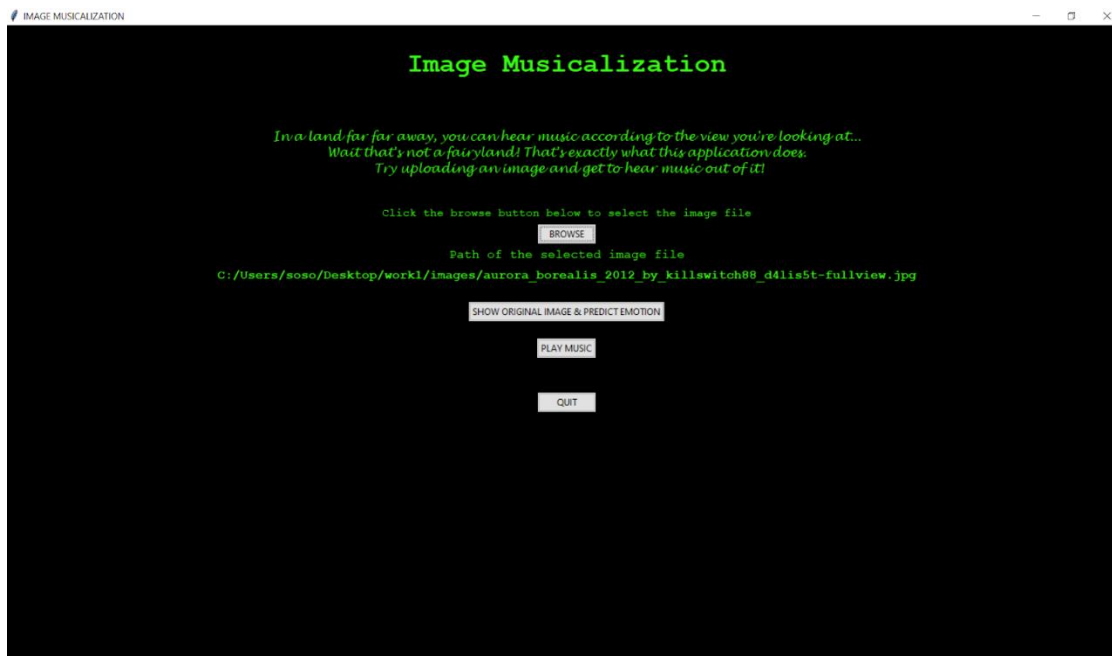


fig.8.3: Here, we can see the path of the image which was chosen by the user.

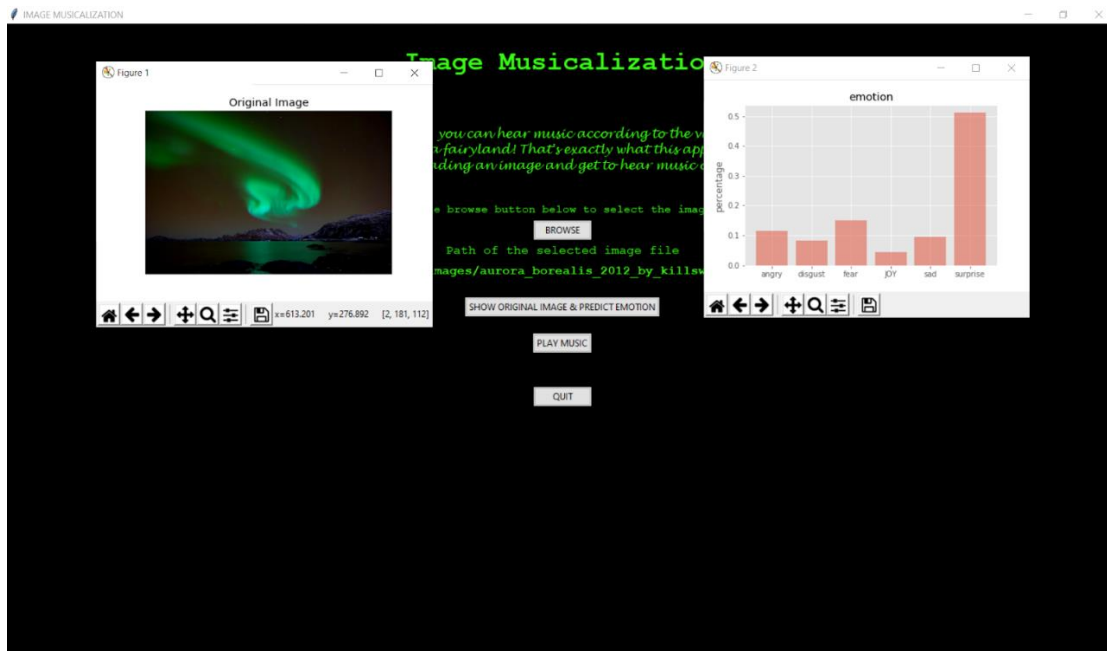


fig.8.4: Output of the image with the histogram representing the emotion of the image.

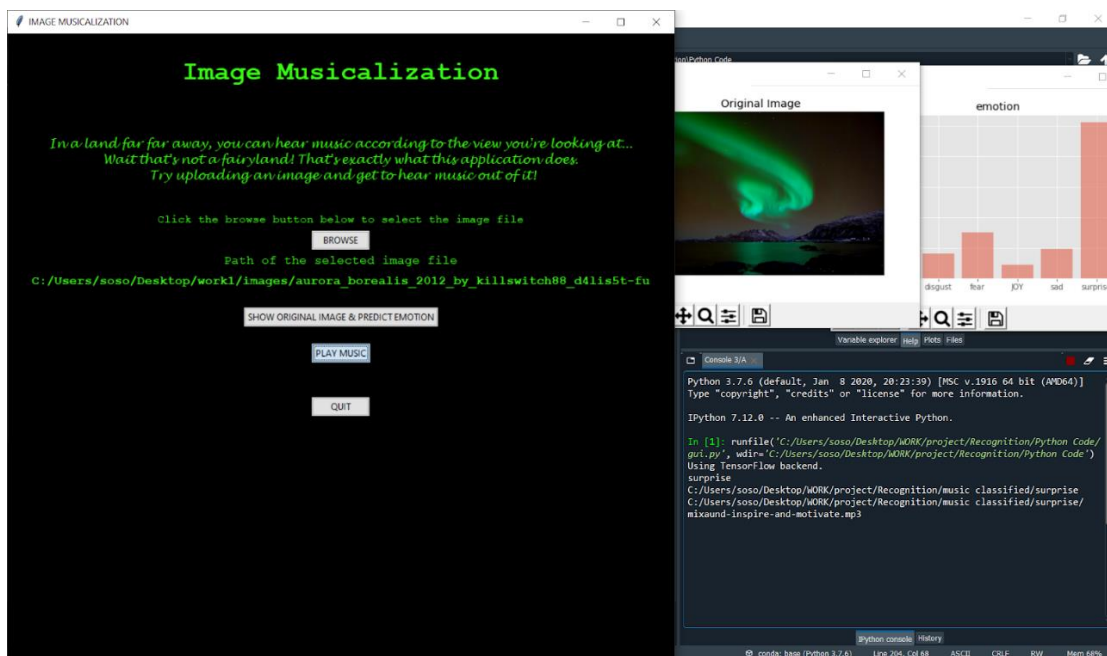


fig.8.5: Once the “play music” button is clicked the music is played. It is played in the GUI; the selected music being played can be viewed in the console on the Spyder IDE. The music is selected based on the image and its emotion which was detected by the program. The current emotion detected from this image is surprise.

8.2 Outputs

Below are the outputs for various user input images using the GUI. (Note that the music playing in all these scenarios is also displayed as the filename with path on the console)

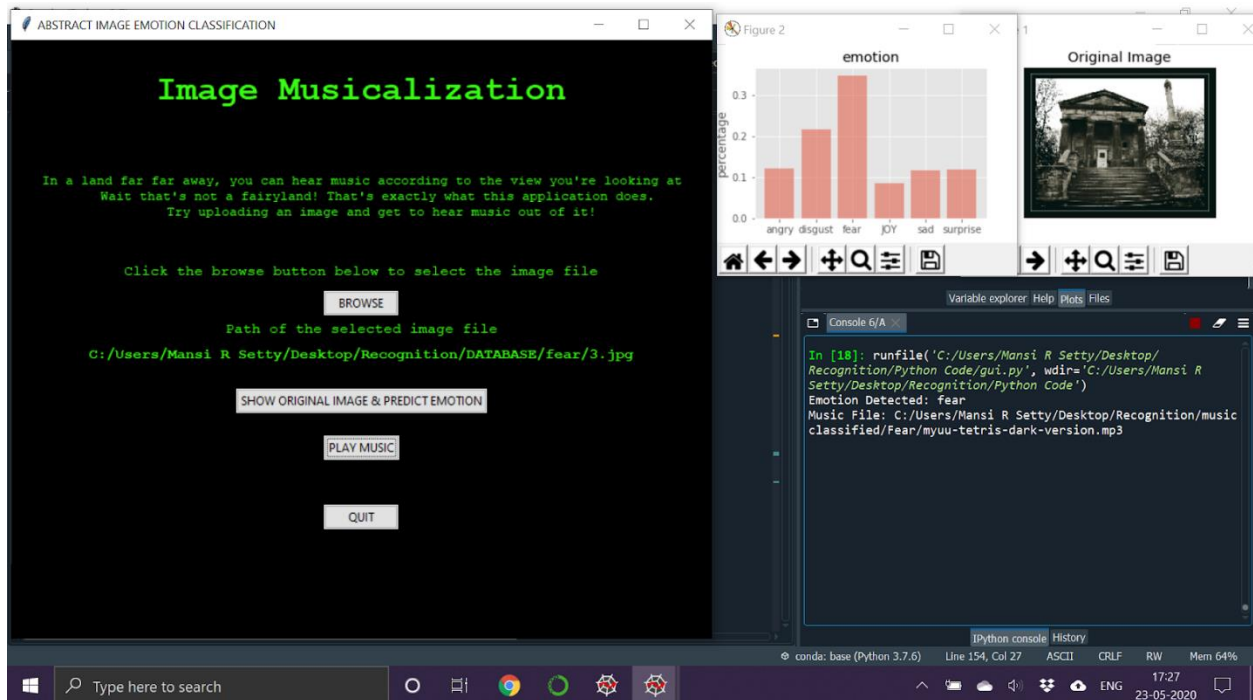


fig.8.6: Fear

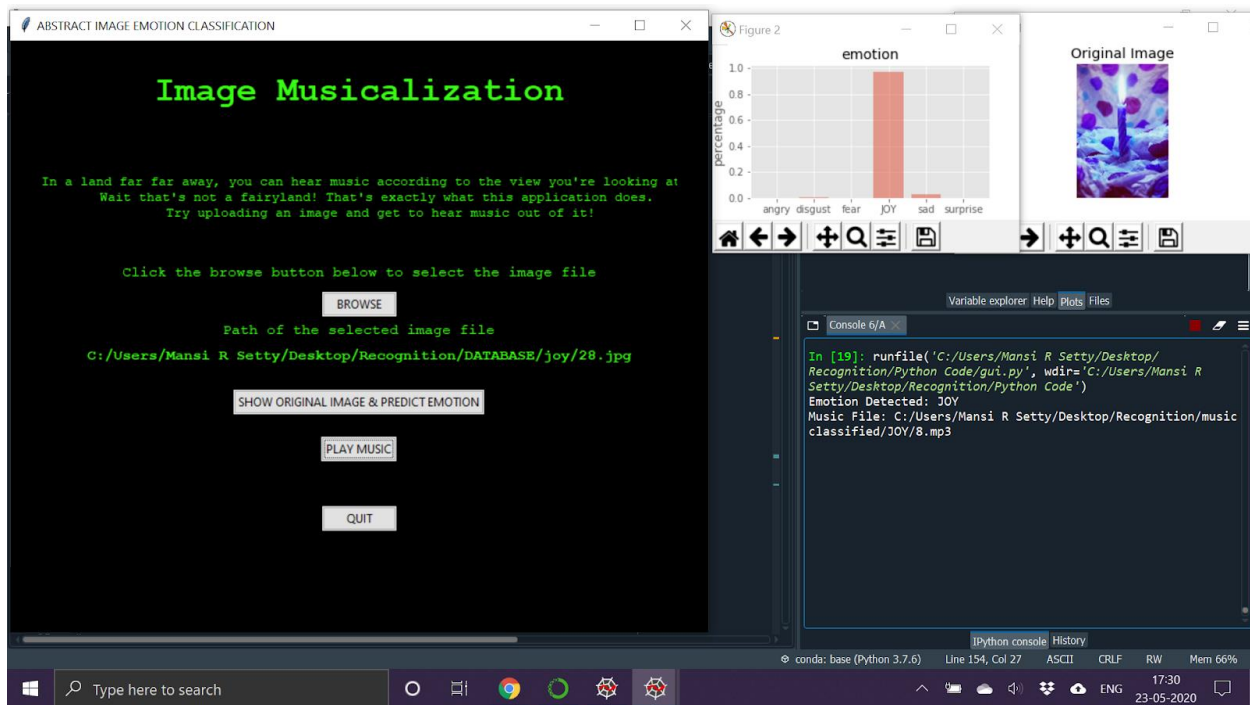


fig.8.7: Joy

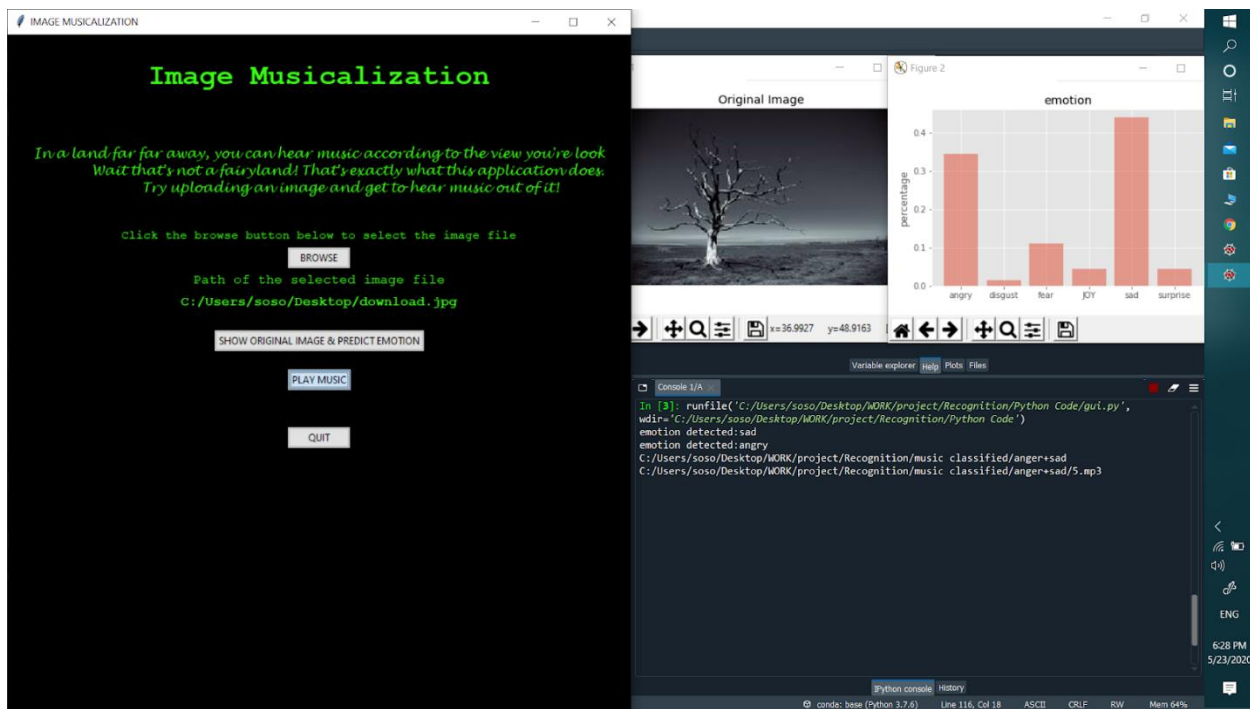


Fig.8.8: this is the combination of angry + sad

Chapter 9

CONCLUSION AND FUTURE ENHANCEMENTS

9.1 Conclusion

In the proposed system image musicalization, detection of the emotions of an image and playing the appropriate music has been implemented by resolving the major issues.

The objectives achieved of the project are explained in the following:

- Collect the defined datasets for image and music which specifically suits the proposed system.
- Earlier solutions had used SVR (support vector regression) to implement the system but in the proposed system we use Convolution Neural Network (CNN).
- We use CNN because it is precisely used to deal with image recognition and the classification.
- In the proposed system we are not just limited to facial images but we do use abstract images as well and detect the emotions from them.
- The proposed system is trained with different epoch and different number of batch sizes which gives the accuracy without overfitting.
- The proposed system is able to detect the emotions out of image and play the appropriate music.
- The results shown display the prediction of emotion label and playing music for predicted emotion. The system was able to detect the multiple emotion labels as well.

9.2 Future enhancements

There are a number of functionalities and features which can be integrated in addition to our proposed system. Such features and additions are listed as future enhancements below.

1. Here, we're only mapping images to music, this can also be done the other way by mapping music also to the images and can be used for a better application.
2. In the current system, we are limited to giving user input images from the gallery. So as a future enhancement we can have the images to be taken live through a camera.
3. Music is being selected from the music database locally available on the desktop. Therefore, we can have a wide variety of music being played when instead of mapping music from the database, we try to play the music using 'Gaana', 'Spotify' or any music application with an online access.
4. Currently we have a desktop python-based GUI. This can be later modified and we can have it running on our cell phones through a smartphone application.
5. Since the current system runs based on local IDE, we cannot have it running on all the systems unless we have an application with a database system. Therefore, we can have the current system in the form of a simple website.

REFERENCES

1. "Representing pictures with sound" Edward m. Schaefer, IEEE 2014
2. "On shape and the computability of emotions" by Xin Lu, Poonam Suryanarayan, Reginald B. Adams, Jr. Jia Li, Michelle G. Newman, James Z. Wang
3. "Predicting discrete probability distribution of image emotions" Sicheng Zhao, Hongxun Yao, Xiaolei Jiang, Xiaoshuai Sun IEEE 2015.
4. "Exploring principles-of-art features for Image Emotion Recognition" by Sicheng Zhao, Yue Gao, Xiaolei Jiang, Hongxun Yao, Tat-Seng Chua, Xiaoshuai Sun (Proceedings of the 22nd ACM International Conference on Multimedia 2014)
5. "Emotion based image musicalization" by Sicheng Zhao, Hongxun Yao, Fanglin Wang, Xiaolei Jiang, Wwei Zhang (International Conference on Multimedia and Expo Workshops (icmew) IEEE 2014)
6. "Building Emotional Machines: Recognizing Image Emotions through Deep Neural Networks" by Hye-Rin Kim, Yeong-Seok Kim, Seon Joo Kim, In-Kwon Lee (2017)
7. "Building a Large-Scale Dataset for Image Emotion Recognition: The Fine Print and The Benchmark" by Quanzeng You, Jiebo Luo, Hailin Jin and Jianchao Yang (2016)
8. "<http://chenlab.ece.cornell.edu/downloads.html>"
9. "<https://www.free-stock-music.com/>"

APPENDIX: CODE SNIPPETS

Train_cnn.py

```
import numpy as np
import pickle
import cv2
from os import listdir
from sklearn.preprocessing import LabelBinarizer
from keras.models import Sequential
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator
from keras.optimizers import Adam
from keras.preprocessing import image
from keras.preprocessing.image import img_to_array
from sklearn.preprocessing import MultiLabelBinarizer
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

directory_root = 'C:/Users/Mansi R Setty/Desktop/Recognition/DATABASE'
#%% Set required parameters
CNN_type=1
if CNN_type==1:
```

```
EPOCHS = 25

INIT_LR = 1e-3

BS = 20

default_image_size = tuple((256, 256))

image_size = 0

width=256

height=256

depth=3

loss_type="binary_crossentropy"

save_model=True

### Set function to convert to gray in case required as We do not
need colour features for Emotion

def convert_image_to_array(image_dir):

    try:

        image = cv2.imread(image_dir)

        if image is not None :

            image = cv2.resize(image, default_image_size)

            return img_to_array(image)

        else :

            return np.array([])

    except Exception as e:

        print(f"Error : {e}")

        return None

### For CNN we need to append image and image label

image_list, label_list = [], []

print("[INFO] Loading images ...")

###Check root directory to remove any files which does not have image
```

```
properties

root_dir = listdir(directory_root)

for directory in root_dir :

    # remove .DS_Store from list

    if directory == ".DS_Store" :

        root_dir.remove(directory)

#%% We are considering all image to have property of PNG but as we
run matlab code all images

for emotion_folder in root_dir :

    emotion_folder_list =
listdir(f"{directory_root}/{emotion_folder}")

    for image in emotion_folder_list[:]:

        image_directory = f"{directory_root}/{emotion_folder}/{image}"

        if image_directory.endswith(".JPG") == True or
image_directory.endswith(".jpg") == True:

            image_list.append(convert_image_to_array(image_directory))

            label_list.append(emotion_folder)

    print("[INFO] Image loading completed")

# Get Size of Processed Image

#%% Find how many image list

image_size = len(image_list)

#%% Convert image to labels and save it

label_binarizer = LabelBinarizer()

image_labels = label_binarizer.fit_transform(label_list)

pickle.dump(label_binarizer,open('label_transform.pkl', 'wb'))

n_classes = len(label_binarizer.classes_)

print(label_binarizer.classes_)
```

```
### Image list normalized

np_image_list = np.array(image_list, dtype=np.float16) / 225.0

### Split to train and test

print("[INFO] Splitting data to train, test")

x_train, x_test, y_train, y_test = train_test_split(np_image_list,
image_labels, test_size=0.2, random_state = 42)

### Generate image with rotation

aug = ImageDataGenerator(

    rotation_range=25, width_shift_range=0.1,

    height_shift_range=0.1, shear_range=0.2,

    zoom_range=0.2, horizontal_flip=True,

    fill_mode="nearest")

### Staart With CNN

if CNN_type==1:

    model = Sequential()

    inputShape = (height, width, depth)

    chanDim = -1

    if K.image_data_format() == "channels_first":

        inputShape = (depth, height, width)

        chanDim = 1

    model.add(Conv2D(32, (3, 3),

padding="same", input_shape=inputShape))

    model.add(Activation("relu")) #rectified linear unit

    model.add(BatchNormalization(axis=chanDim))

    model.add(MaxPooling2D(pool_size=(3, 3)))

    model.add(Dropout(0.25))
```



```
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
print(model.summary())
opt = Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS)
model.compile(loss=loss_type, optimizer=opt, metrics=["accuracy"])
```

```
# Train network

print("[INFO] training network...")

history = model.fit_generator(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, verbose=1)

print('MYHISTORY',history.history)

acc = history.history['accuracy']
val_acc = history.history['val_accuracy']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)

#Train and validation accuracy
plt.plot(epochs, acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()
plt.figure()

#Train and validation loss
plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()
```

```
### Model Accuracy

print("[INFO] Calculating model accuracy")

scores = model.evaluate(x_test, y_test)

print(f"Test Accuracy: {scores[1]*100}")


# Save model using Pickle to disk

if save_model:

    print("[INFO] Saving model...")

    pickle.dump(model, open('cnn_model.pkl', 'wb'))
```

Test_cnn.py

```
### TEST

import cv2

from keras.preprocessing.image import img_to_array

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import os, random, time

import pygame

from pygame import mixer

#make prediction for custom image out of test set

directory='C:/Users/Mansi R Setty/Desktop/Recognition/music
classified/'

img = 'C:/Users/Mansi R
Setty/Desktop/Recognition/DATABASE/sadness/4.jpg'

default_image_size = tuple ((256, 256))

def convert_image_to_array(image_dir):
```

```
try:

    image = cv2.imread(image_dir)

    plt.axis('off')

    plt.imshow(image)

    plt.show()

    color = ('b','g','r')

    for i,col in enumerate(color):

        histr = cv2.calcHist([image], [i], None,[256], [0,256])

        plt.plot(histr,color = col)

        plt.xlim([0,256])

        plt.show()

    if image is not None:

        image = cv2.resize(image, default_image_size)

        return img_to_array(image)

    else:

        return np.array([])

except Exception as e:

    print (f"Error: {e}")

    return None

def emotion_analysis(emotions):

    objects = ('angry', 'disgust', 'fear', 'JOY', 'sad', 'surprise')

    y_pos = np.arange(len(objects))

    plt.bar(y_pos, emotions, align='center', alpha=0.5)

    plt.xticks(y_pos, objects)

    plt.ylabel('percentage')

    plt.title('emotion')
```

```
plt.show()

default_image_size = tuple ((256, 256))

x=convert_image_to_array(img)

x = np.expand_dims(x, axis = 0)

x /= 255

import pickle

file_object = open ('cnn_model.pkl', 'rb')

file_object1= open ('label_transform.pkl', 'rb')

label_binarizer=pickle.load(file_object1) #not used in this model

model=pickle.load(file_object)

custom = model.predict(x)

emotion_analysis(custom [0])

percentage=pd.DataFrame(np.round([custom[0]*100])).T

pos=pd.DataFrame(percentage[0].nlargest(2))

pos[pos[0] >40]

Categories = pd.DataFrame(['angry', 'disgust', 'fear', 'JOY', 'sad',
'surprise'])

req_val=pos[pos[0] >40]

if len(req_val)>1:

    combination=2

elif len(req_val)==1 :

    combination=1

else:

    combination=0

    print("Neutral")

    folder =directory+'Neutral'
```

```
print(folder)

while (combination==2):

    out=pd.Series(req_val.index.astype(int))

    decision=Categories[0][out[0]]

    decision1=Categories[0][out[1]]

    print ("Emotion Detected: "+Categories[0][out[0]])

    print (Categories[0][out[1]])

    if ((decision or decision1)=="fear" and (decision or
decision1)=="angry"):

        folder =directory+'fear+anger'

    elif ((decision or decision1)=='sad' and (decision or
decision1)=='fear'):

        folder =directory+'fear+sad'

    elif ((decision or decision1)=='angry' and (decision or
decision1)=='sad'):

        folder =directory+'anger+sad'

    elif ((decision or decision1)=='disgust' and (decision or
decision1)=='angry'):

        folder =directory+'disgust+anger'

    elif ((decision or decision1)=='JOY' and (decision or
decision1)=='surprise'):

        folder =directory+'joy+surprise'

    combination=0 #to exit the loop

while (combination==1):

    out=pd.Series(req_val.index.astype(int))

    decision=Categories[0][out[0]]

    print ("Emotion Detected: "+Categories[0][out[0]])

    if decision=='fear':

        folder =directory+'Fear'
```

```
elif decision=='disgust':
    folder =directory+'Disgust'
elif decision=='angry':
    folder =directory+'Angry'
elif decision=='JOY':
    folder =directory+'Joy'
elif decision=='sad':
    folder =directory+'Sad'
elif decision=='surprise':
    folder =directory+'Surprise'
else:
    folder =directory+'Neutral'
combination=0

# Music Part
mediapath = folder
path=mediapath
files=os.listdir(path)
d=random.choice(files)
final_path= path+'/'+d
# To trim music to desired length
print("\nMusic File: "+final_path)
mixer.init()
mixer.music.load(final_path)
mixer.music.play()
time.sleep(20)
pygame.mixer.music.stop()
```

Gui.py:

```
## Tkinter GUI

root = tk.Tk()

tk.Tk.wm_title(root, "ABSTRACT IMAGE EMOTION CLASSIFICATION")

root.configure(background='#000000')

label = ttk.Label(root, text="Image Musicalization", font=("Courier",
26, "bold"), background="#000000", foreground="#39FF14")

label.pack(side="top", pady=30, padx=50)

desc = '''In a land far far away, you can hear music according to the
view you're looking at...

Wait that's not a fairyland! That's exactly what this application
does.

Try uploading an image and get to hear music out of it!'''

label = ttk.Label(root, justify=tk.CENTER, text=desc, font=("Lucida
handwriting", 12),background="#000000", foreground="#39FF14")

label.pack(side="top", pady=30, padx=30)

label = ttk.Label(root, justify=tk.CENTER,

                    text="Click the browse button below to select the
image file", font= ("Courier", 12), background="#000000",
foreground="#39FF14")

label.pack(side="top", pady=10, padx=30)

button1 = ttk.Button(root, text="BROWSE", command=lambda:
browsefunc())

button1.pack()

label = ttk.Label(root, justify=tk.CENTER, text="Path of the selected
image file", font=("Courier", 11),background="#000000",
foreground="#39FF14")

label.pack(side="top", pady=3, padx=30)

pathlabel = ttk.Label(root, font=("Courier", 12,
"bold"),background="#000000", foreground="#39FF14")
```



```
pathlabel.pack(side="top", pady=3, padx=30)

label = ttk.Label(root, justify=tk.CENTER, text="",
background="#000000", foreground="#39FF14")

label.pack(side="top", pady=1, padx=30)

button1 = ttk.Button(root, text="SHOW ORIGINAL IMAGE & PREDICT
EMOTION", command=lambda: show_lr(global_filename))

button1.pack()

label = ttk.Label(root, justify=tk.CENTER, text="",
background="#000000", foreground="#39FF14")

label.pack(side="top", pady=2, padx=30)

button2 = ttk.Button(root, text="PLAY MUSIC", command=lambda:
show_sr(custom))

button2.pack()

label = ttk.Label(root, justify=tk.CENTER, text="",
background="#000000", foreground="#39FF14")

label.pack(side="top", pady=2, padx=30)

label = ttk.Label(root, justify=tk.CENTER, text="",
background="#000000", foreground="#39FF14")

label.pack(side="top", pady=2, padx=30)

button3 = ttk.Button(root, text="QUIT", command=lambda: show_exit())

button3.pack()

label = ttk.Label(root, justify=tk.CENTER, text="",
background="#000000", foreground="#39FF14")

label.pack(side="top", pady=5, padx=30)

if __name__ == "__main__":
    root.mainloop()
```