

**HMR INSTITUTE OF TECHNOLOGY & MANAGEMENT**

**HAMIDPUR, DELHI – 110036**

**Affiliated to**

**GURU GOBIND SINGH INDRAPRASTHA UNIVERSITY**

**Sector - 16C Dwarka, Delhi - 110075, India 2008-12**

**Summer Training Project Report**

**on**

**MOODIFY**

Submitted in partial Fulfilment of the requirements for

The award of the degree of

**Bachelor of Technology**

**In**

**(COMPUTER SCIENCE ENGINEERING)**

**Guide: Submitted By:**

**Mr. Sachin Singh Arnav Choudhary**

**CSE -5E**

**00596502716**

**Utkarsh Dua**

**CSE- 5E**

**02696502716**

**Himanshu Saini**

**CSE- 5E**

**00996502716**

**Shivani Jadon**

**CSE- 5E**

**02396502716**

**DECLARATION**

I/We, student(s) of B.Tech (CSE 5th sem) hereby declare that the project entitled “MOODIFY(Suggestion of Songs on the basis of Facial Emotion Recognition)” which is submitted to Department of CSE, HMR Institute of Technology & Management, Hamidpur Delhi, affiliated to Guru Gobind Singh Indraprastha University, Dwarka(New Delhi) in partial fulfilment of requirement for the award of the degree of Bachelor of Technology in CSE, has not been previously formed the basis for the award of any degree, diploma or other similar title or recognition. The list of member(s) involved in the project is listed below: -

1. Arnav Choudhary (00596502716)
2. Himanshu Saini (00996502716)
3. Shivani Jadon (02396502716)
4. Utkarsh Dua (02696502716)

This is to certify that the above statement made by the candidate(s) is correct to the best of my knowledge.

|  |  |
| --- | --- |
| New Delhi | Mr. Sachin Singh |
| Date: July 13,2018 | Signature of Guide : |

**Mr. Ravinder Kumar**

**(Head of Department)**

**(Assistant Professor of Computer Science Department)**

**HMRITM Hamidpur, New Delhi-110036**

**Contents**

[**Role of Team Mates** 5](#_Toc519215883)

[**Abstract** 6](#_Toc519215884)

[**Algorithm Used** 7](#_Toc519215885)

[1. Convolutional Neural Network (CNN) 7](#_Toc519215886)

[**Libraries Used** 11](#_Toc519215887)

[1. NumPy: 11](#_Toc519215888)

[2. Keras.model.Sequential: 11](#_Toc519215889)

[3. Keras.Layers: 12](#_Toc519215890)

[4. Open CV 12](#_Toc519215891)

[5. OS: 13](#_Toc519215892)

[6. Subprocesses: 14](#_Toc519215893)

[7. Tkinter: 15](#_Toc519215894)

[**Function Used** 16](#_Toc519215895)

[1. keras.preprocessiong.ImageDataGenetor : 16](#_Toc519215896)

[2. keras.layers.Dense: 17](#_Toc519215897)

[5. Keras.layers.Dropout: 18](#_Toc519215898)

[4. Keras.layers.Flatten: 18](#_Toc519215899)

[5. Keras.layers.Conv2D : 19](#_Toc519215900)

[6. Keras.layers.Maxpooling: 19](#_Toc519215901)

[7. Face Detection with Haar Cascade: 20](#_Toc519215902)

[**Dataset** 22](#_Toc519215903)

[**CODE DESCRIPTION** 23](#_Toc519215904)

[**Variable Explorer** 29](#_Toc519215905)

[**IPython Console** 30](#_Toc519215906)

[**Code** 33](#_Toc519215907)

[**Summary** 52](#_Toc519215908)

# **Role of Team Mates**

Arnav Choudhary

(00596502716)

* Training the model for facial emotion recognition.
* Designing the algorithm for image segregation.

Himanshu Saini

(00996502716)

* Training the model for facial emotion recognition.
* Algorithm designing for music player.

Shivani Jadon

(02396502716)

* Graphical user interface designing.
* Testing the model.

Utkasrh Dua

(02696502716)

* Collection of data for model and music player.
* Preprocessing of the data and images.

# **Abstract**

Moodify’ is a song suggested which recommend the song to the user according to his mood. The major application of ‘Moodify’ is to suggest the music genres which are best fitted for the person’s mood at that time. The mood detection is done using image segregation. We have used Convolution Neural Network for image preprocessing and then detecting the mood. Our dataset model for now has been trained for three basic emotions that are happiness, sadness and excitement. A photo of user is clicked that shows the mood of the user. And then the result is generated determining if the person is happy or sad or excited. After mood detection the user will choose whether he wants to use the offline mode or online mode of music recommendation. The mode will then direct the user to the recommended music playlist best suited for his mood. Then the user will select the song from the suggested list and the music player will play the song. Hence the user will be able to reach the music which he wants to listen without having to go through a boring process of searching the music himself. ‘Moodify’ will do the job leaving the user to get carried away with the music.

# **Algorithm Used**

# Convolutional Neural Network (CNN)

In neural networks, Convolutional neural network is one of the main categories to do image recognition, image classification. Object detection, face detection etc. where CNN is widely used

CNN image classifications takes an input image, process it and classify it under certain categories (Eg. Dog, Cat, Tiger, Lion). Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d (h = Height, w = Width, d = Dimension). Eg. An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.

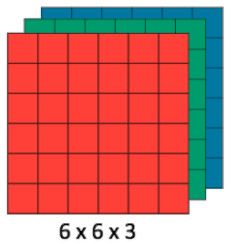


Figure: Array of RGB Matrix

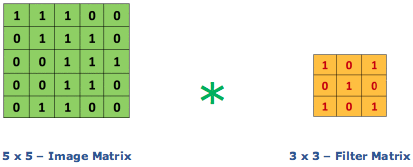
Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify an object with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies the objects based on values.



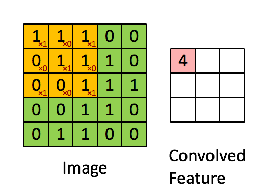
Figure: Neural Network with many convolutional layers

Convolution is the first layer to extract features from an input image. Convolution preserves the relationship between pixels by learning image features using small squares of input data. It is a mathematical operation that takes two inputs such as image matrix and a filter or kernel.

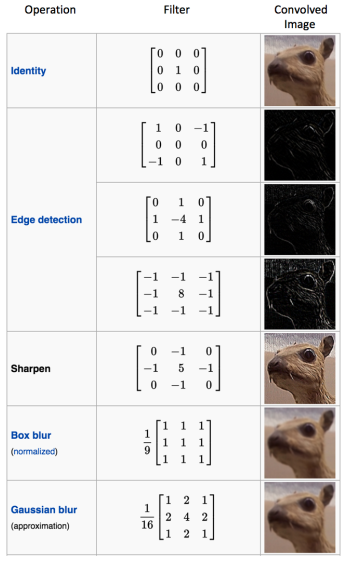
Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below:



Then the convolution of 5 x 5 image matrix multiplies with 3 x 3 filter matrix which is called **“Feature Map”** as output shown in below:



Convolution of an image with different filters can perform operations such as edge detection, blur and sharpen by applying filters. The below example shows various convolution image after applying different types of filters.



**Padding**

Sometimes filter does not fit perfectly fit the input image. We have two options:

* Pad the picture with zeros (zero-padding) so that it fits
* Drop the part of the image where the filter did not fit. This is called valid padding which keeps only valid part of the image.

**Non Linearity (ReLU)**

ReLU stands for Rectified Linear Unit for a non-linear operation. The output is ***ƒ(x) = max(0,x).***

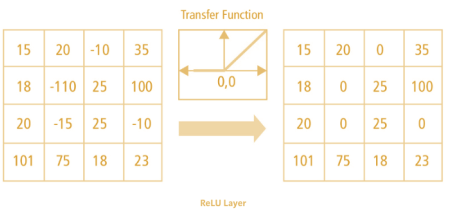


Figure: ReLu Operation

**Pooling Layer**

Pooling layers section would reduce the number of parameters when the images are too large. Spatial pooling also called subsampling or down sampling which reduces the dimensionality of each map but retains the important information. Spatial pooling can be of different types:

* Max Pooling
* Average Pooling
* Sum Pooling

Max pooling take the largest element from the rectified feature map. Taking the largest element could also take the average pooling. Sum of all elements in the feature map call as sum pooling.

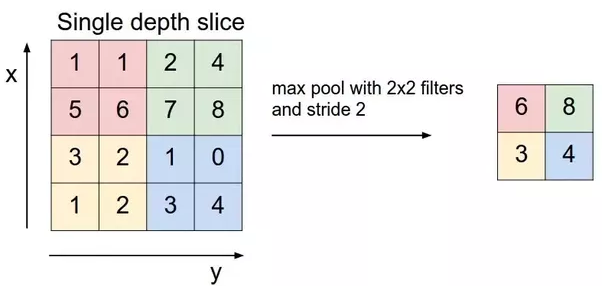


Figure: Max Pooling

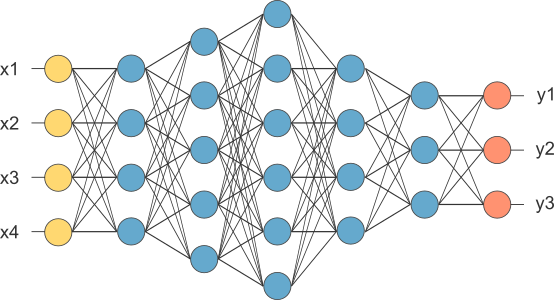
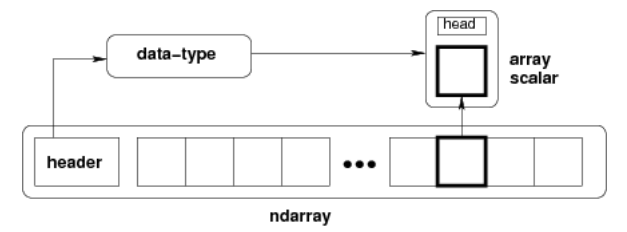


Figure: After pooling layer, flattened as FC layer

# **Libraries Used**

## NumPy:

NumPy provides an N-dimensional array type, the ndarray, which describes a collection of “items” of the same type. The items can be indexed using for example N integers. All ndarrays are homogenous: every item takes up the same size block of memory, and all blocks are interpreted in exactly the same way. How each item in the array is to be interpreted is specified by a separate data-type object, one of which is associated with every array. In addition to basic types (integers, floats, etc.), the data type objects can also represent data structures. An item extracted from an array, e.g., by indexing, is represented by a Python object whose type is one of the array scalar types built in Numpy. The array scalars allow easy manipulation of also more complicated arrangements of data.



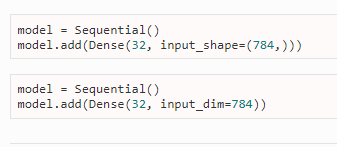
## Keras.model.Sequential:

The Sequential model is a linear stack of layers.

**Specifying the input shape :**

The model needs to know what input shape it should expect. For this reason, the first layer in a Sequential model (and only the first, because following layers can do automatic shape inference) needs to receive information about its input shape. There are several possible ways to do this:

* Pass an input\_shape argument to the first layer. This is a shape tuple (a tuple of integers or none entries, where none indicates that any positive integer may be expected). In input\_shape, the batch dimension is not included.
* Some 2D layers, such as Dense, support the specification of their input shape via the argument input\_dim, and some 3D temporal layers support the arguments input\_dim and input\_length.
* If you ever need to specify a fixed batch size for your inputs (this is useful for stateful recurrent networks), you can pass a batch\_size argument to a layer. If you pass both batch\_size=32 and input\_shape= (6, 8) to a layer, it will then expect every batch of inputs to have the batch shape (32, 6, 8).



**Compilation:**

Before training a model, you need to configure the learning process, which is done via the compile method. It receives three arguments:

* An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class. See: optimizers.
* A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as categorical\_crossentropy or mse), or it can be an objective function. See: losses.
* A list of metrics. For any classification problem you will want to set this to metrics= ['accuracy']. A metric could be the string identifier of an existing metric or a custom metric function.

## Keras.Layers:

All Keras layers have a number of methods in common:

* layer.get\_weights(): returns the weights of the layer as a list of Numpy arrays.
* layer.set\_weights(weights): sets the weights of the layer from a list of Numpy arrays (with the same shapes as the output of get\_weights).
* layer.get\_config(): returns a dictionary containing the configuration of the layer. The layer can be reinstantiated from its config via:



## Open CV

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSD-licensed library that includes several hundreds of computer vision algorithms. The document describes the so-called OpenCV 2.x API, which is essentially a C++ API, as opposite to the C-based OpenCV 1.x API. The latter is described in opencv1x.pdf.

OpenCV has a modular structure, which means that the package includes several shared or static libraries. The following modules are available:

* core - a compact module defining basic data structures, including the dense multi-dimensional array Mat and basic functions used by all other modules.
* imgproc - an image processing module that includes linear and non-linear image filtering, geometrical image transformations (resize, affine and perspective warping, generic table-based remapping), color space conversion, histograms, and so on.
* Video - a video analysis module that includes motion estimation, background subtraction, and object tracking algorithms.
* calib3d - basic multiple-view geometry algorithms, single and stereo camera calibration, and object pose estimation, stereo correspondence algorithms, and elements of 3D reconstruction.
* features2d - salient feature detectors, descriptors, and descriptor matchers.
* objdetect - detection of objects and instances of the predefined classes (for example, faces, eyes, mugs, people, cars, and so on).
* highgui - an easy-to-use interface to video capturing, image and video codecs, as well as simple UI capabilities.
* gpu - GPU-accelerated algorithms from different OpenCV modules.

... Some other helper modules, such as FLANN and Google test wrappers, Python bindings, and others.

The further chapters of the document describe functionality of each module. But first, make sure to get familiar with the common API concepts used thoroughly in the library.

It has C++, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

## OS:

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the fileinput module. For creating temporary files and directories see the tempfile module, and for high-level file and directory handling see the shutil module.

Notes on the availability of these functions:

* The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example, the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).
* Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

## Subprocesses:

The subprocesses module allows you to spawn new processes, connect to their input/output/error pipes, and obtain their return codes. This module intends to replace several older modules and functions:

The arguments shown above are merely the most common ones, described below in Frequently Used Arguments (hence the use of keyword-only notation in the abbreviated signature). The full function signature is largely the same as that of the Popen constructor - most of the arguments to this function are passed through to that interface. (timeout, input, check, and capture\_output are not.)

If capture\_output is true, stdout and stderr will be captured. When used, the internal Popen object is automatically created with stdout=PIPE and stderr=PIPE. The stdout and stderr arguments may not be used as well.

The timeout argument is passed to Popen.communicate(). If the timeout expires, the child process will be killed and waited for. The TimeoutExpired exception will be re-raised after the child process has terminated.

The input argument is passed to Popen.communicate() and thus to the subprocess’s stdin. If used it must be a byte sequence, or a string if encoding or errors is specified or text is true. When used, the internal Popen object is automatically created with stdin=PIPE, and the stdin argument may not be used as well.

If check is true, and the process exits with a non-zero exit code, a CalledProcessError exception will be raised. Attributes of that exception hold the arguments, the exit code, and stdout and stderr if they were captured.

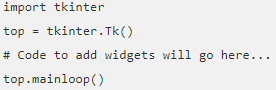
If encoding or errors are specified, or text is true, file objects for stdin, stdout and stderr are opened in text mode using the specified encoding and errors or the io.TextIOWrapper default. The universal\_newlines argument is equivalent to text and is provided for backwards compatibility. By default, file objects are opened in binary mode.

## Tkinter:

Tkinter is the standard GUI library for Python. Python when combined with Tkinter provides a fast and easy way to create GUI applications. Tkinter provides a powerful object-oriented interface to the Tk GUI toolkit.

Creating a GUI application using Tkinter is an easy task. All you need to do is perform the following steps −

* Import the Tkinter module.
* Create the GUI application main window.
* Add one or more of the above-mentioned widgets to the GUI application.
* Enter the main event loop to take action against each event triggered by the user.



# **Function Used**

## keras.preprocessiong.ImageDataGenetor :

keras.preprocessing.image.ImageDataGenerator(featurewise\_center=**False**, samplewise\_center=**False**, featurewise\_std\_normalization=**False**, samplewise\_std\_normalization=**False**, zca\_whitening=**False**, zca\_epsilon=1e-06, rotation\_range=0.0, width\_shift\_range=0.0, height\_shift\_range=0.0, brightness\_range=**None**, shear\_range=0.0, zoom\_range=0.0, channel\_shift\_range=0.0, fill\_mode='nearest', cval=0.0, horizontal\_flip=**False**, vertical\_flip=**False**, rescale=**None**, preprocessing\_function=**None**, data\_format=**None**, validation\_split=0.0)

Arguments

* featurewise\_center: Boolean. Set input mean to 0 over the dataset, feature-wise.
* samplewise\_center: Boolean. Set each sample mean to 0.
* featurewise\_std\_normalization: Boolean. Divide inputs by std of the dataset, feature-wise.
* samplewise\_std\_normalization: Boolean. Divide each input by its std.
* zca\_epsilon: epsilon for ZCA whitening. Default is 1e-6.
* zca\_whitening: Boolean. Apply ZCA whitening.
* rotation\_range: Int. Degree range for random rotations.
* width\_shift\_range: Float, 1-D array-like or int

float: fraction of total width, if < 1, or pixels if >= 1.

1-D array-like: random elements from the array.

int: integer number of pixels from interval (-width\_shift\_range, +width\_shift\_range)

With width\_shift\_range=2 possible values are integers [-1, 0, +1], same as with width\_shift\_range=[-1, 0, +1], while with width\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).

* height\_shift\_range: Float, 1-D array-like or int

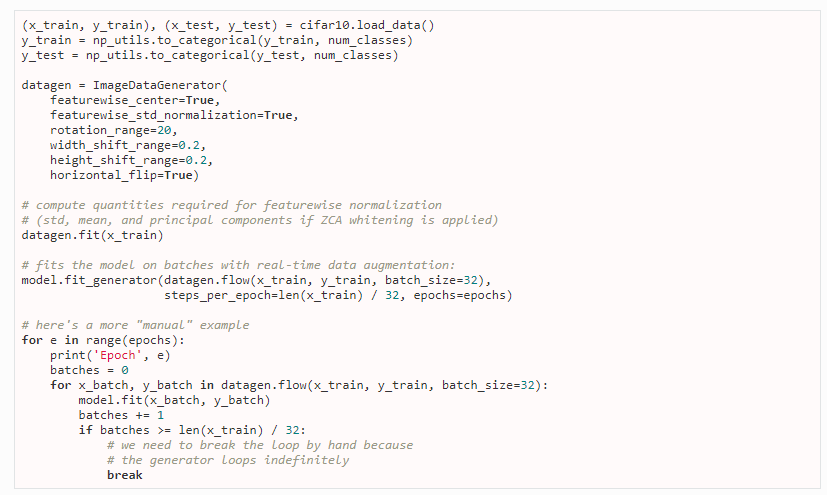
float: fraction of total height, if < 1, or pixels if >= 1.

1-D array-like: random elements from the array.

int: intger number of pixels from interval (-height\_shift\_range, +height\_shift\_range)

With height\_shift\_range=2 possible values are integers [-1, 0, +1], same as with height\_shift\_range=[-1, 0, +1], while with height\_shift\_range=1.0 possible values are floats in the interval [-1.0, +1.0).

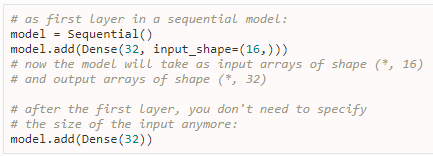
shear\_range: Float. Shear Intensity (Shear angle in counter-clockwise direction in degrees)



## keras.layers.Dense:

keras.layers.Dense(units, activation=**None**, use\_bias=**True**, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=**None**, bias\_regularizer=**None**, activity\_regularizer=**None**, kernel\_constraint=**None**, bias\_constraint=**None**)

Dense implements the operation: output = activation(dot(input, kernel) + bias) where activation is the element-wise activation function passed as the activation argument, kernel is a weights matrix created by the layer, and bias is a bias vector created by the layer (only applicable if use\_bias is True).



## Keras.layers.Dropout:

keras.layers.Dropout(rate, noise\_shape=**None**, seed=**None**)

Applies Dropout to the input.

Dropout consists in randomly setting a fraction rate of input units to 0 at each update during training time, which helps prevent overfitting

**Arguments**

* + rate: float between 0 and 1. Fraction of the input units to drop.
  + noise\_shape: 1D integer tensor representing the shape of the binary dropout mask that will be multiplied with the input. For instance, if your inputs have shape (batch\_size, timesteps, features) and you want the dropout mask to be the same for all timesteps, you can use noise\_shape=(batch\_size, 1, features).
  + seed: A Python integer to use as random seed.

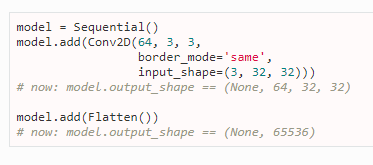
## Keras.layers.Flatten:

keras.layers.Flatten(data\_format=**None**)

Flattens the input. Does not affect the batch size.

**Arguments**

data\_format: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. The purpose of this argument is to preserve weight ordering when switching a model from one data format to another. channels\_last corresponds to inputs with shape (batch, ..., channels) while channels\_first corresponds to inputs with shape (batch, channels, ...). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".



## Keras.layers.Conv2D :

keras.layers.Conv2D(filters, kernel\_size, strides=(1, 1), padding='valid', data\_format=**None**, dilation\_rate=(1, 1), activation=**None**, use\_bias=**True**, kernel\_initializer='glorot\_uniform', bias\_initializer='zeros', kernel\_regularizer=**None**, bias\_regularizer=**None**, activity\_regularizer=**None**, kernel\_constraint=**None**, bias\_constraint=**None**)

2D convolution layer (e.g. spatial convolution over images).

This layer creates a convolution kernel that is convolved with the layer input to produce a tensor of outputs. If use\_bias is True, a bias vector is created and added to the outputs. Finally, if activation is not None, it is applied to the outputs as well.

When using this layer as the first layer in a model, provide the keyword argument input\_shape (tuple of integers, does not include the sample axis), e.g. input\_shape=(128, 128, 3) for 128x128 RGB pictures in data\_format="channels\_last".

## Keras.layers.Maxpooling:

keras.layers.MaxPooling3D(pool\_size=(2, 2, 2), strides=**None**, padding='valid', data\_format=**None**)

**Arguments:**

* + pool\_size: tuple of 3 integers, factors by which to downscale (dim1, dim2, dim3). (2, 2, 2) will halve the size of the 3D input in each dimension.
  + strides: tuple of 3 integers, or None. Strides values.
  + padding: One of "valid" or "same" (case-insensitive).
  + data\_format: A string, one of channels\_last (default) or channels\_first. The ordering of the dimensions in the inputs. channels\_last corresponds to inputs with shape (batch, spatial\_dim1, spatial\_dim2, spatial\_dim3, channels) while channels\_first corresponds to inputs with shape (batch, channels, spatial\_dim1, spatial\_dim2, spatial\_dim3). It defaults to the image\_data\_format value found in your Keras config file at ~/.keras/keras.json. If you never set it, then it will be "channels\_last".

## Face Detection with Haar Cascade:

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.



But among all these features we calculated, most of them are irrelevant. For example, consider the image below. The top row shows two good features. The first feature selected seems to focus on the property that the region of the eyes is often darker than the region of the nose and cheeks. The second feature selected relies on the property that the eyes are darker than the bridge of the nose.

**Haar-cascade Detection in OpenCV**

OpenCV comes with a trainer as well as detector. If you want to train your own classifier for any object like car, planes etc. you can use OpenCV to create one. Its full details are given here: [**Cascade Classifier Training**](https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html).

Here we will deal with detection. OpenCV already contains many pre-trained classifiers for face, eyes, smiles, etc. Those XML files are stored in the opencv/data/haarcascades/ folder. Let's create a face and eye detector with OpenCV.

First we need to load the required XML classifiers. Then load our input image (or video) in grayscale mode

import numpy as np

import cv2 as cv

face\_cascade = [cv.CascadeClassifier](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html)('haarcascade\_frontalface\_default.xml')

eye\_cascade = [cv.CascadeClassifier](https://docs.opencv.org/3.4/d1/de5/classcv_1_1CascadeClassifier.html)('haarcascade\_eye.xml')

img = [cv.imread](https://docs.opencv.org/3.4/d4/da8/group__imgcodecs.html#ga288b8b3da0892bd651fce07b3bbd3a56)('sachin.jpg')

gray = [cv.cvtColor](https://docs.opencv.org/3.4/d7/d1b/group__imgproc__misc.html#ga397ae87e1288a81d2363b61574eb8cab)(img, cv.COLOR\_BGR2GRAY)

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

for (x,y,w,h) in faces:

[cv.rectangle](https://docs.opencv.org/3.4/d6/d6e/group__imgproc__draw.html#ga346ac30b5c74e9b5137576c9ee9e0e8c)(img,(x,y),(x+w,y+h),(255,0,0),2)

roi\_gray = gray[y:y+h, x:x+w]

roi\_color = img[y:y+h, x:x+w]

eyes = eye\_cascade.detectMultiScale(roi\_gray)

for (ex,ey,ew,eh) in eyes:

[cv.rectangle](https://docs.opencv.org/3.4/d6/d6e/group__imgproc__draw.html#ga346ac30b5c74e9b5137576c9ee9e0e8c)(roi\_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

[cv.imshow](https://docs.opencv.org/3.4/df/d24/group__highgui__opengl.html#gaae7e90aa3415c68dba22a5ff2cefc25d)('img',img)

[cv.waitKey](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga5628525ad33f52eab17feebcfba38bd7)(0)

[cv.destroyAllWindows](https://docs.opencv.org/3.4/d7/dfc/group__highgui.html#ga6b7fc1c1a8960438156912027b38f481)()

Some other methods of OpenCV used are imread(Reading Image file),imwrite(writing a frame or image object),imshow(displaying frames of video),VideoCapture(Using the camera)

# **Dataset**

The dataset we have used “Cohn-Kanade”. The Cohn-Kanade AU-Coded Facial Expression Database is for research in automatic facial image analysis and synthesis and for perceptual studies. Cohn-Kanade is available in two versions and a third is in preparation.

This dataset is classified so we cannot provide the actual dataset but the link for you to download is :

<http://www.consortium.ri.cmu.edu/ckagree/index.cgi>

And to read more about dataset you can refer to:

<http://www.pitt.edu/~emotion/ck-spread.htm>

Feature Extraction and Selection :

1. Lips
2. Eyes
3. Forehead
4. Nose

These features are processed by CNN layers and then selected by the algorithm and then they are converted to NumPy array and then model is trained by that and the following three classifications are made.

# 



# **CODE DESCRIPTION**

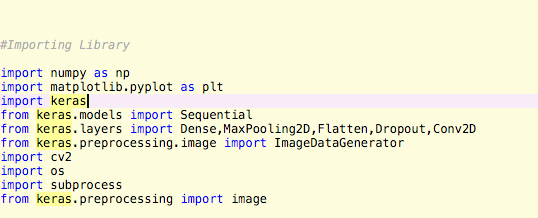


Figure : All libraries are imported in this .

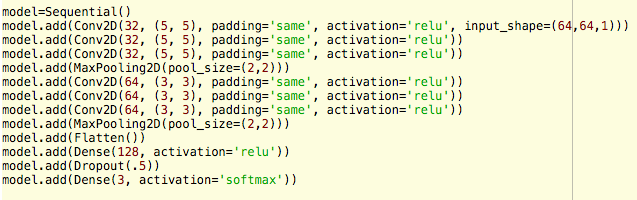


Figure : Model Initialization and building.

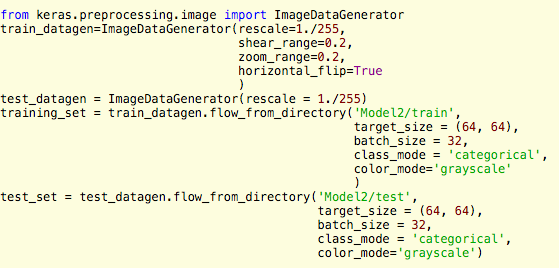


Figure : Training of test and testing.



Figure : Training our model

Figure : Model Buliding, Splitting of test and train set and training of model.

Screen%20Shot%202018-07-13%20at%203.08.59%20AM.png

Figure : Saving a model.

Screen%20Shot%202018-07-13%20at%2012.58.58%20AM.png

Figure : Loading a saved model.

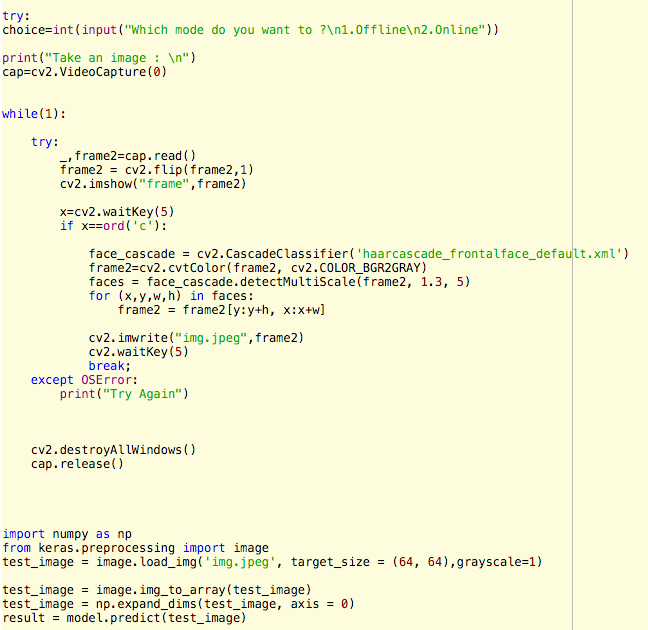


Figure : Saving image with opencv after cropping and loading it and then prediction

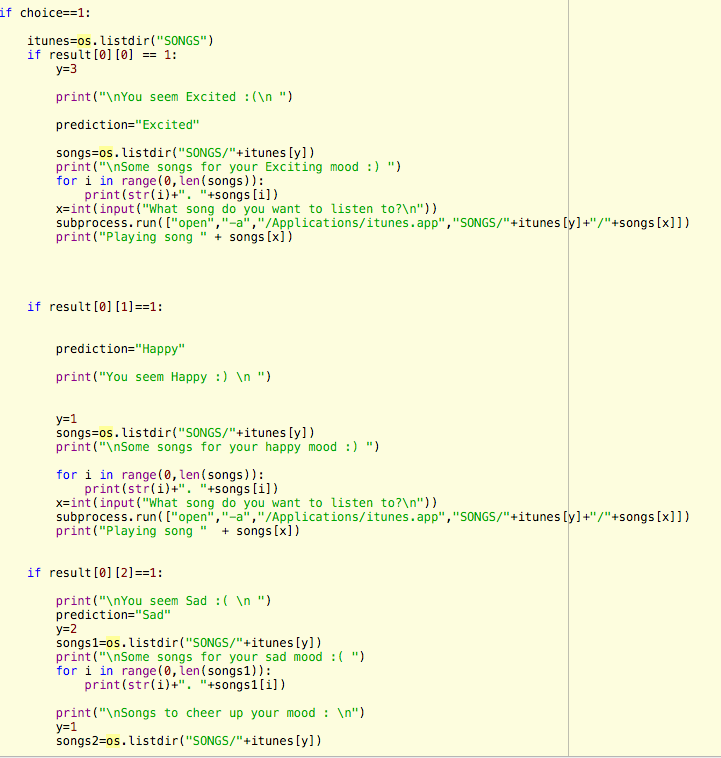


Figure: Suggesting songs in Offline mode

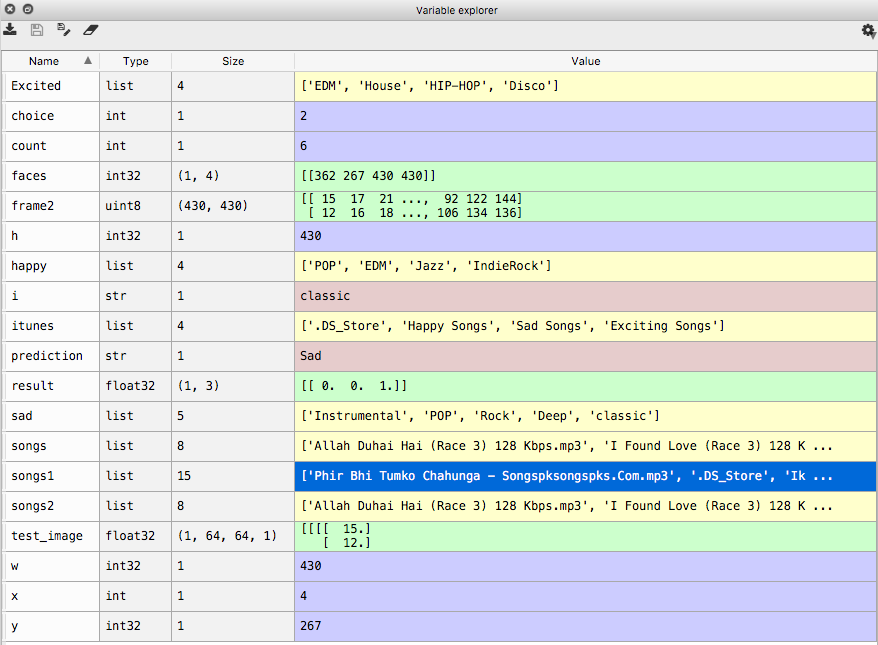


Figure: Suggesting songs online(Youtube)



Figure : Rest of GUI part

# **Variable Explorer**



# **IPython Console**

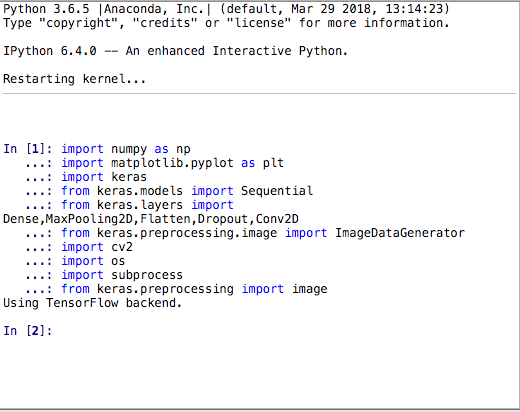


Figure: Importing Libraries

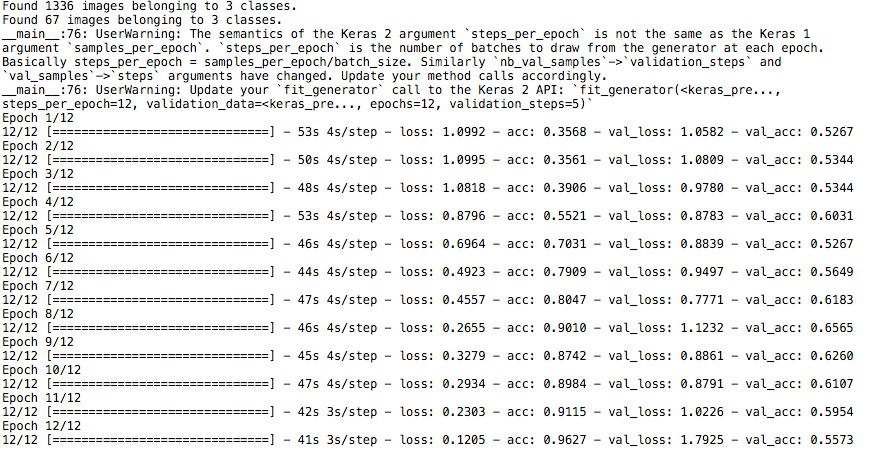


Figure: Model Training

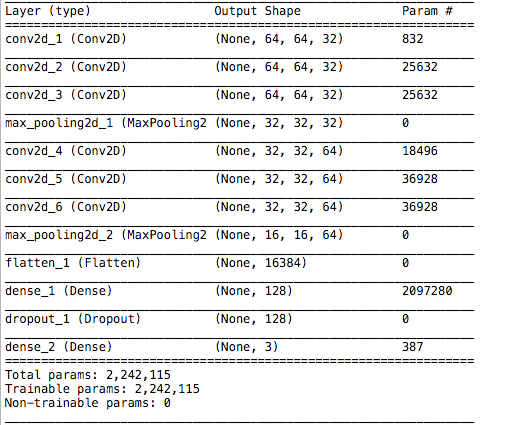


Figure: Model Summary

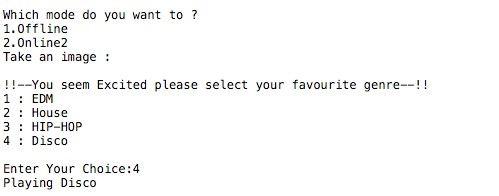


Figure: Online Mode

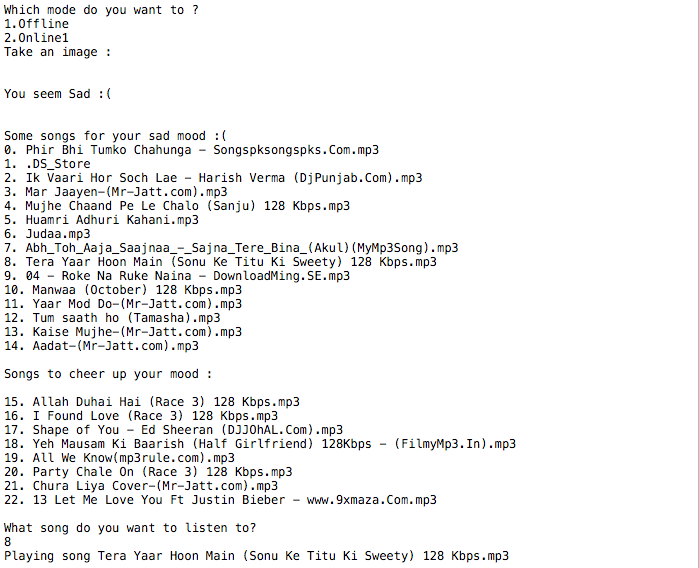


Figure: Offline Mode

# **GUI**

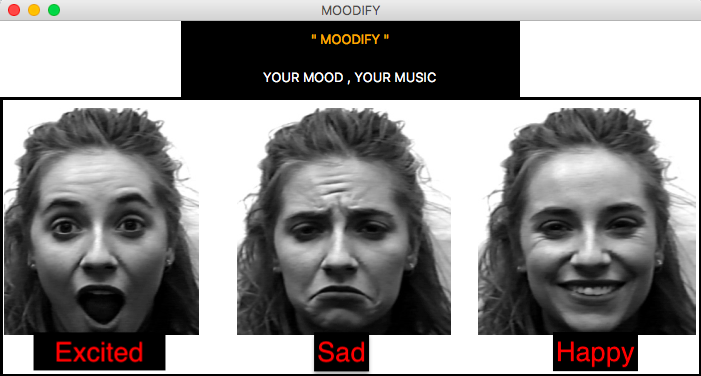


Figure : Splash Screen

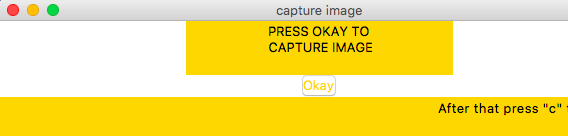


Figure: Main Screen

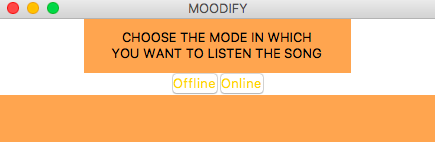


Figure : Selection screen

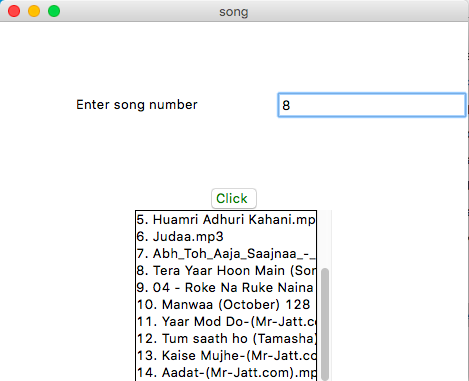


Figure: Display songs and then slelect, after that they will play

# **Code**

1. Code for running on ipython console with training :

#!/usr/bin/env python3

# -\*- coding: utf-8 -\*-

"""

Created on Wed Jul 11 09:13:29 2018

@author: arnavchoudhary

"""

#%%

#Importing Library

import numpy as np

import matplotlib.pyplot as plt

import keras

from keras.models import Sequential

from keras.layers import Dense,MaxPooling2D,Flatten,Dropout,Conv2D

from keras.preprocessing.image import ImageDataGenerator

import cv2

import os

import subprocess

from keras.preprocessing import image

#%%

model=Sequential()

model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input\_shape=(64,64,1)))

model.add(Conv2D(32, (5, 5), padding='same', activation='relu'))

model.add(Conv2D(32, (5, 5), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))

model.add(MaxPooling2D(pool\_size=(2,2)))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(.5))

model.add(Dense(3, activation='softmax'))

model.compile(loss='categorical\_crossentropy',

optimizer='adam',

metrics=['accuracy'])

from keras.preprocessing.image import ImageDataGenerator

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

shear\_range=0.2,

zoom\_range=0.2,

horizontal\_flip=True

)

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

training\_set = train\_datagen.flow\_from\_directory('Model2/train',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'categorical',

color\_mode='grayscale'

)

test\_set = test\_datagen.flow\_from\_directory('Model2/test',

target\_size = (64, 64),

batch\_size = 32,

class\_mode = 'categorical',

color\_mode='grayscale')

model.fit\_generator(training\_set,

steps\_per\_epoch = 12 ,

nb\_epoch = 12,

validation\_data = test\_set,

nb\_val\_samples = 5)

model.summary()

#%%

model = keras.models.load\_model("Final\_Model\_categories\_3.model")

#%%

#check

import cv2

import os

import subprocess

print("Take an image : \n")

cap=cv2.VideoCapture(0)

while(1):

try:

\_,frame2=cap.read()

frame2 = cv2.flip(frame2,1)

cv2.imshow("frame",frame2)

x=cv2.waitKey(5)

if x==ord('q')&0xFF:

break

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

frame2=cv2.cvtColor(frame2, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(frame2, 1.3, 5)

for (x,y,w,h) in faces:

frame2 = frame2[y:y+h, x:x+w]

cv2.imwrite("img.jpeg",frame2)

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('img.jpeg', target\_size = (64, 64),grayscale=1)

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = model.predict(test\_image)

if result[0][0]==1:

print("Excited")

if result[0][1]==1:

print("Happy")

if result[0][2]==1:

print("Sadness")

except OSError:

print("Try Again")

cap.release()

cv2.destroyAllWindows()

#%%

#SHOW CLASS INDICES

training\_set.class\_indices

#%%

#SAVE MODEL

model.save(Final\_Model\_categories\_3.model")

#%%

#PLAYING SONGS

import cv2

import os

try:

choice=int(input("Which mode do you want to ?\n1.Offline\n2.Online"))

print("Take an image : \n")

cap=cv2.VideoCapture(0)

while(1):

try:

\_,frame2=cap.read()

frame2 = cv2.flip(frame2,1)

cv2.imshow("frame",frame2)

x=cv2.waitKey(5)

if x==ord('c'):

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

frame2=cv2.cvtColor(frame2, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(frame2, 1.3, 5)

for (x,y,w,h) in faces:

frame2 = frame2[y:y+h, x:x+w]

cv2.imwrite("img.jpeg",frame2)

cv2.waitKey(5)

break;

except OSError:

print("Try Again")

cv2.destroyAllWindows()

cap.release()

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('img.jpeg', target\_size = (64, 64),grayscale=1)

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = model.predict(test\_image)

if choice==1:

itunes=os.listdir("SONGS")

if result[0][0] == 1:

y=3

print("\nYou seem Excited :(\n ")

prediction="Excited"

songs=os.listdir("SONGS/"+itunes[y])

print("\nSome songs for your Exciting mood :) ")

for i in range(0,len(songs)):

print(str(i)+". "+songs[i])

x=int(input("What song do you want to listen to?\n"))

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs[x]])

print("Playing song " + songs[x])

if result[0][1]==1:

prediction="Happy"

print("You seem Happy :) \n ")

y=1

songs=os.listdir("SONGS/"+itunes[y])

print("\nSome songs for your happy mood :) ")

for i in range(0,len(songs)):

print(str(i)+". "+songs[i])

x=int(input("What song do you want to listen to?\n"))

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs[x]])

print("Playing song " + songs[x])

if result[0][2]==1:

print("\nYou seem Sad :( \n ")

prediction="Sad"

y=2

songs1=os.listdir("SONGS/"+itunes[y])

print("\nSome songs for your sad mood :( ")

for i in range(0,len(songs1)):

print(str(i)+". "+songs1[i])

print("\nSongs to cheer up your mood : \n")

y=1

songs2=os.listdir("SONGS/"+itunes[y])

for i in range(0,len(songs2)):

print(str(i+len(songs1))+". "+songs2[i])

x=int(input("What song do you want to listen to?\n"))

if(x<len(songs1)):

y=2

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs1[x]])

print("Playing song " + songs1[x])

else:

y=1

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs2[x-len(songs1)]])

print("Playing song " + songs2[x-len(songs1)])

elif choice==2:

import webbrowser

happy = ['POP','EDM','Jazz','IndieRock']

sad = ['Instrumental','POP','Rock','Deep','classic']

Excited = ['EDM','House','HIP-HOP','Disco']

if result[0][1] ==1 :

count= 1

print('!!--You seem Happy please select your favourite genre--!!')

for i in happy:

print('{} : {}'.format(count,i))

count += 1

x = int(input('Enter Your Choice:'))

if x == 1:

webbrowser.open('https://www.youtube.com/watch?v=pgN-vvVVxMA&list=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start\_radio=1')

elif x== 2:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg\_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start\_radio=1')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=21LGv8Cf0us&list=PLMcThd22goGYit-NKu2O8b4YMtwSTK9b9&start\_radio=1')

elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=VQH8ZTgna3Q&list=PLVAJ90ZhCcL896CZDbuIz2HGKeVekfEee&start\_radio=1')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + happy[x-1])

elif result[0][2] == 1:

count= 1

print('!!--You Seem Sad please select your favourite genre--!!')

for i in sad:

print('{} : {}'.format(count,i))

count += 1

x = int(input('Enter Your Choice:'))

if x == 1:

webbrowser.open('https://www.youtube.com/watch?v=Pa\_\_NZaRXxs&list=PLIWSikhI2\_z2lNqsfjF4ahut28056cJtz')

elif x== 2:

webbrowser.open('https://www.youtube.com/watch?v=pgN-vvVVxMA&list=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start\_radio=1')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJzmzH7kkxjfdVoPPj&start\_radio=12')

elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=UAWcs5H-qgQ&list=PLzzwfO\_D01M4nNqJKR828zz6r2wGikC5a')

elif x==5:

webbrowser.open('https://www.youtube.com/watch?v=4Tr0otuiQuU&list=RDQMqk8OvYGJVWM&start\_radio=1')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + sad[x-1])

elif result[0][0] == 1:

count= 1

print('!!--You seem Excited please select your favourite genre--!!')

for i in Excited:

print('{} : {}'.format(count,i))

count += 1

x = int(input('Enter Your Choice:'))

if x == 1:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg\_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start\_radio=1')

elif x== 2:

webbrowser.open('https://www.youtube.com/watch?v=BDocp-VpCwY&list=PLhInz4M-OzRUsuBj8wF6383E7zm2dJfqZ&start\_radio=1')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=xTlNMmZKwpA&start\_radio=1&list=PLH6pfBXQXHEC2uDmDy5oi3tHW6X8kZ2Jo')

elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=kJQP7kiw5Fk&list=PL64E6BD94546734D8')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + Excited[x-1])

else:

webbrowser.open('https://www.youtube.com/watch?v=aJOTlE1K90k&list=PLw-VjHDlEOgvtnnnqWlTqByAtC7tXBg6D')

1. Program to run with GUI

# -\*- coding: utf-8 -\*-

"""

Created on Wed Jul 11 19:26:14 2018

@author: NEW

"""

import tkinter as tk

from tkinter import \*

import cv2

import os

import subprocess

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import keras

from keras.models import Sequential

from keras.layers import Dense,MaxPooling2D,Flatten,Dropout,Conv2D

model = keras.models.load\_model("Final\_Model\_categories\_3.model")

choice=1

import time

from PIL import ImageTk, Image

def callfunc(list1):

def makeWindow () :

win = Tk()

frame1 = Frame(win)

frame1.pack()

win.title('song')

Label(frame1, text='Enter song number', width = 30, height = 10).grid(row=0)

e1 = Entry(frame1)

e1.grid(row=0, column=1)

frame2 = Frame(win) # Row of buttons

frame2.pack()

b1 = Button(frame2,text=" Click ",bg = 'yellow',foreground = 'green',command = lambda : func(int(e1.get())))

b1.pack(side = LEFT)

frame3 = Frame(win) # select of names

frame3.pack()

scroll = Scrollbar(frame3, orient=VERTICAL)

mylist = Listbox(frame3, yscrollcommand = scroll.set )

for i in range(0,len(list1)):

mylist.insert(END,str(i) +". "+ list1[i])

scroll.config (command=mylist.yview)

scroll.pack(side=RIGHT, fill=Y)

mylist.pack(side=LEFT, fill=BOTH, expand=1)

return win

def func(ch):

global choice

wind.destroy()

choice=ch

global choice

wind = makeWindow()

wind.mainloop()

return choice

def splashscreen () :

window = Tk()

window.title('MOODIFY')

F1 = Frame(window)

F1.pack()

l1 = Label(F1, text='" MOODIFY "',height = 2, width = 37, font = 'coopergothicbold', bg = 'black', foreground ='orange')

l1.pack()

l2 = Label(F1, text = 'YOUR MOOD , YOUR MUSIC',height = 2, width = 37,font = 'cooperblack', bg = 'black', foreground = 'white')

l2.pack()

img = ImageTk.PhotoImage(Image.open("mood.png"))

l3 = Label(F1, image = img, bg ='black')

l3.img=img

l3.pack()

#l3.pack(side = 'bottom', fill= 'both', expand = 'yes')

return window

def CaptureImage():

print("Take an image : \n")

cap=cv2.VideoCapture(0)

while(1):

\_,frame2=cap.read()

frame2 = cv2.flip(frame2,1)

cv2.imshow("frame",frame2)

x=cv2.waitKey(5)

if x==ord('c'):

face\_cascade = cv2.CascadeClassifier('haarcascade\_frontalface\_default.xml')

frame2=cv2.cvtColor(frame2, cv2.COLOR\_BGR2GRAY)

faces = face\_cascade.detectMultiScale(frame2, 1.3, 5)

for (x,y,w,h) in faces:

frame2 = frame2[y:y+h, x:x+w]

cv2.imwrite("img.jpeg",frame2)

cv2.waitKey(5)

break

cv2.destroyAllWindows()

cap.release()

wind.destroy()

def startWindow() :

window = splashscreen()

window.update()

time.sleep(2)

window.destroy()

time.sleep(0.5)

wind = Tk()

wind.title('capture image')

f1 = Frame(wind)

f1.pack()

label1 = Label(f1, text='PRESS OKAY TO\n CAPTURE IMAGE ', width = 29, height = 3, font = 'algerian', bg = 'gold')

label1.pack()

f2 = Frame(wind)

f2.pack()

button = Button(f2, text='Okay', bg ='forest green',foreground = 'gold', font = 'algerian',command = CaptureImage)

#button = Button(command = f2.destroy)

#button.grid(row = 1, rowspan = 2, sticky = 'E')

button.pack()

f1 = Frame(wind)

f1.pack()

label2 = Label(f1, text='\t\t\t\t\t\t\n\t\t\t\t\t\t', bg = 'gold')

label2.pack()

return wind

wind = startWindow()

wind.mainloop()

#for offline mode

def offline():

wind.destroy()

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('img.jpeg', target\_size = (64, 64),grayscale=1)

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = model.predict(test\_image)

itunes=os.listdir("SONGS")

if result[0][0] == 1:

y=3

print("\nYou seem Excited :(\n ")

prediction="Excited"

songs=os.listdir("SONGS/"+itunes[y])

x=callfunc(songs)

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs[x]])

print("Playing song " + songs[x])

if result[0][1]==1:

prediction="Happy"

print("You seem Happy :) \n ")

y=1

songs=os.listdir("SONGS/"+itunes[y])

x=callfunc(songs)

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs[x]])

print("Playing song " + songs[x])

if result[0][2]==1:

print("\nYou seem Sad :( \n ")

prediction="Sad"

y=2

songs1=os.listdir("SONGS/"+itunes[y])

x=callfunc(songs1)

subprocess.run(["open","-a","/Applications/itunes.app","SONGS/"+itunes[y]+"/"+songs1[x]])

print("Playing song " + songs1[x])

#for online mode

def online():

wind.destroy()

import numpy as np

from keras.preprocessing import image

test\_image = image.load\_img('img.jpeg', target\_size = (64, 64),grayscale=1)

test\_image = image.img\_to\_array(test\_image)

test\_image = np.expand\_dims(test\_image, axis = 0)

result = model.predict(test\_image)

import webbrowser

happy = ['POP','EDM','Jazz','IndieRock']

sad = ['Instrumental','POP','Rock','Deep','classic']

Excited = ['EDM','House','HIP-HOP','Disco']

if result[0][1] ==1 :

print('!!--You seem Happy please select your favourite genre--!!')

prediction="Happy"

x=callfunc(happy)

if x == 0:

webbrowser.open('https://www.youtube.com/watch?v=pgN-vvVVxMA&list=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start\_radio=1')

elif x== 1:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg\_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start\_radio=1')

elif x==2:

webbrowser.open('https://www.youtube.com/watch?v=21LGv8Cf0us&list=PLMcThd22goGYit-NKu2O8b4YMtwSTK9b9&start\_radio=1')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=VQH8ZTgna3Q&list=PLVAJ90ZhCcL896CZDbuIz2HGKeVekfEee&start\_radio=1')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + happy[x])

elif result[0][2] == 1:

x=callfunc(sad)

print('!!--You Seem Sad please select your favourite genre--!!')

prediction="Sad"

if x == 0:

webbrowser.open('https://www.youtube.com/watch?v=Pa\_\_NZaRXxs&list=PLIWSikhI2\_z2lNqsfjF4ahut28056cJtz')

elif x== 1:

webbrowser.open('https://www.youtube.com/watch?v=pgN-vvVVxMA&list=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start\_radio=1')

elif x==2:

webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJzmzH7kkxjfdVoPPj&start\_radio=12')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=UAWcs5H-qgQ&list=PLzzwfO\_D01M4nNqJKR828zz6r2wGikC5a')

elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=4Tr0otuiQuU&list=RDQMqk8OvYGJVWM&start\_radio=1')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + sad[x])

elif result[0][0] == 1:

x=callfunc(Excited)

print('!!--You seem Excited please select your favourite genre--!!')

prediction="Exciting"

if x == 0:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg\_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start\_radio=1')

elif x== 1:

webbrowser.open('https://www.youtube.com/watch?v=BDocp-VpCwY&list=PLhInz4M-OzRUsuBj8wF6383E7zm2dJfqZ&start\_radio=1')

elif x==2:

webbrowser.open('https://www.youtube.com/watch?v=xTlNMmZKwpA&start\_radio=1&list=PLH6pfBXQXHEC2uDmDy5oi3tHW6X8kZ2Jo')

elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=kJQP7kiw5Fk&list=PL64E6BD94546734D8')

else:

webbrowser.open('https://www.youtube.com')

print("Playing " + Excited[x])

else:

webbrowser.open('https://www.youtube.com/watch?v=aJOTlE1K90k&list=PLw-VjHDlEOgvtnnnqWlTqByAtC7tXBg6D')

def SecondWindow () :

wind = Tk()

wind.title('MOODIFY')

Frame(bg = 'tan1',height = 15)

f1 = Frame(wind)

f1.grid()

Label(f1, text='CHOOSE THE MODE IN WHICH\n YOU WANT TO LISTEN THE SONG ', width = 29, height = 3, font = 'algerian', bg = 'tan1').grid(row=0, rowspan = 1)

f2 = Frame(wind)

f2.grid(row = 1, column = 0, columnspan = 2)

button1 = Button(f2, text='Offline', bg = 'White',foreground = 'Gold', relief = 'sunken', command = offline).grid(row = 1,column = 0,columnspan = 2 )

button2 = Button(f2, text='Online', bg = 'White',foreground = 'Gold', relief = 'sunken', command =online).grid(row = 1,column = 2,columnspan = 2)

f4 = Frame(wind)

f4.grid()

Label(f4, text='\t\t\t\t\t\t', bg = 'tan1', height =3).grid(row = 2)

return wind

wind = SecondWindow()

wind.mainloop()

# 

# **Summary**

We successfully build a model for Facial Emotion Recognition(FER) and trained it with an average accuracy over various test sets of over 75%. Then we successfully build a Desktop application to suggest songs on the basis of their facial expression and hence completed our project. This FER model can be widely used for various purposes such as home automation, social media, E-commerce, etc and we have the motivation to take this project to a next level.