Contents

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

1. 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.

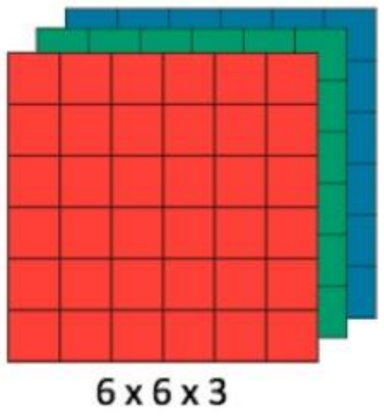


Figure: Array of RGB Matrix

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.

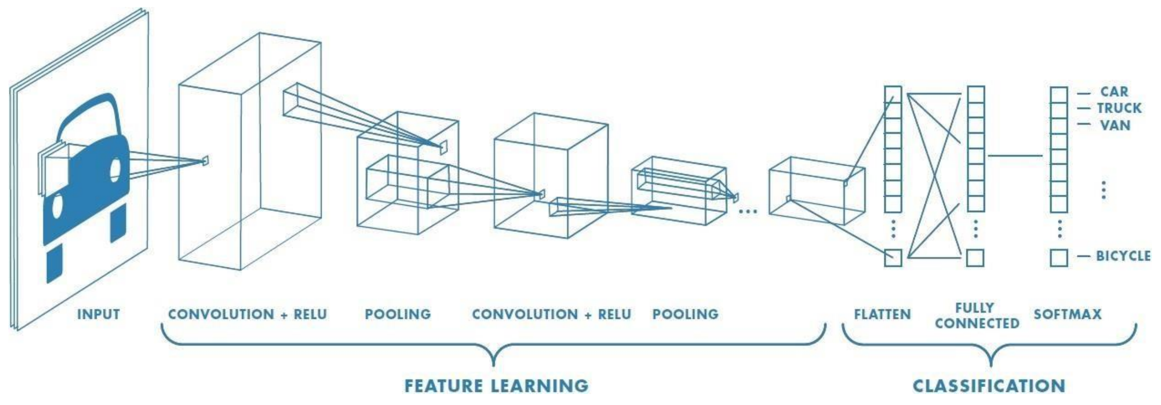
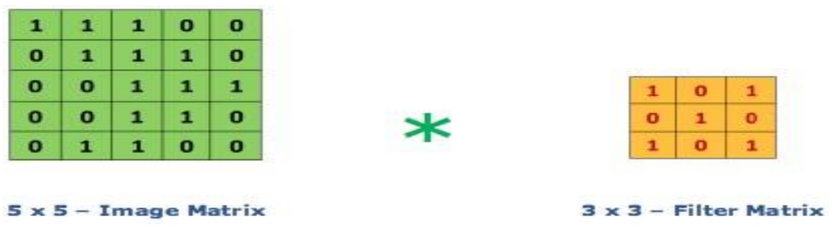


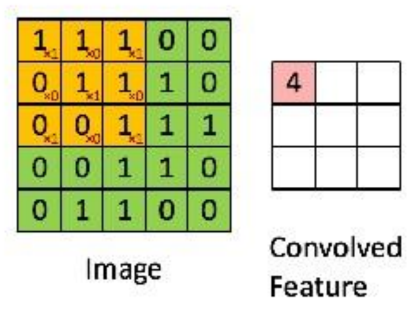
Figure: Neural Network with many convolutional layers

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.

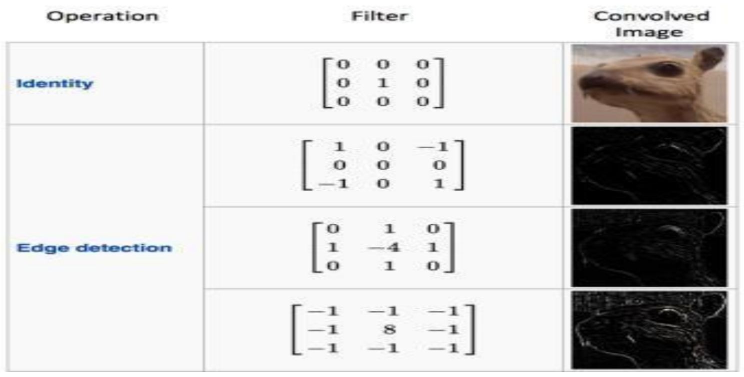
Consider a 5 x 5 whose image pixel values are 0, 1 and filter matrix 3 x 3 as shown in below:



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.



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

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 ***f(x) = max(0,x).***

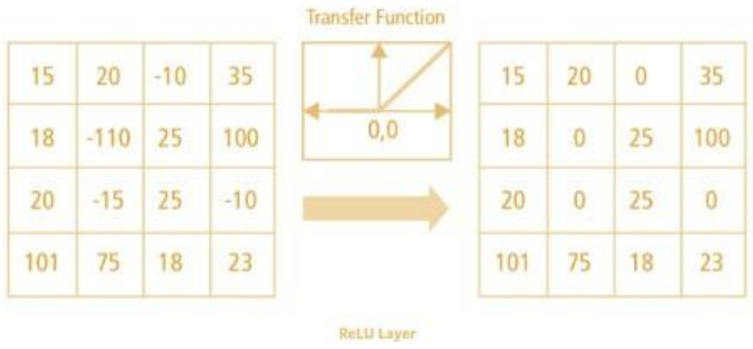


Figure: ReLu Operation

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.

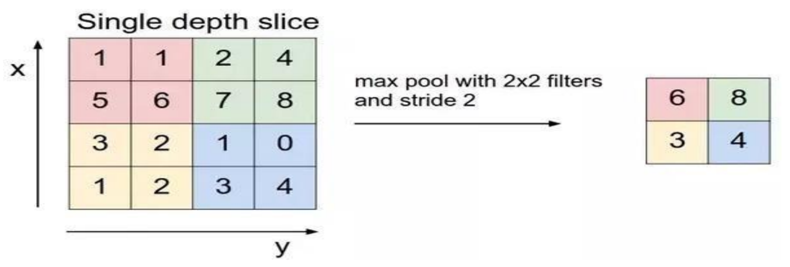


Figure: Max Pooling

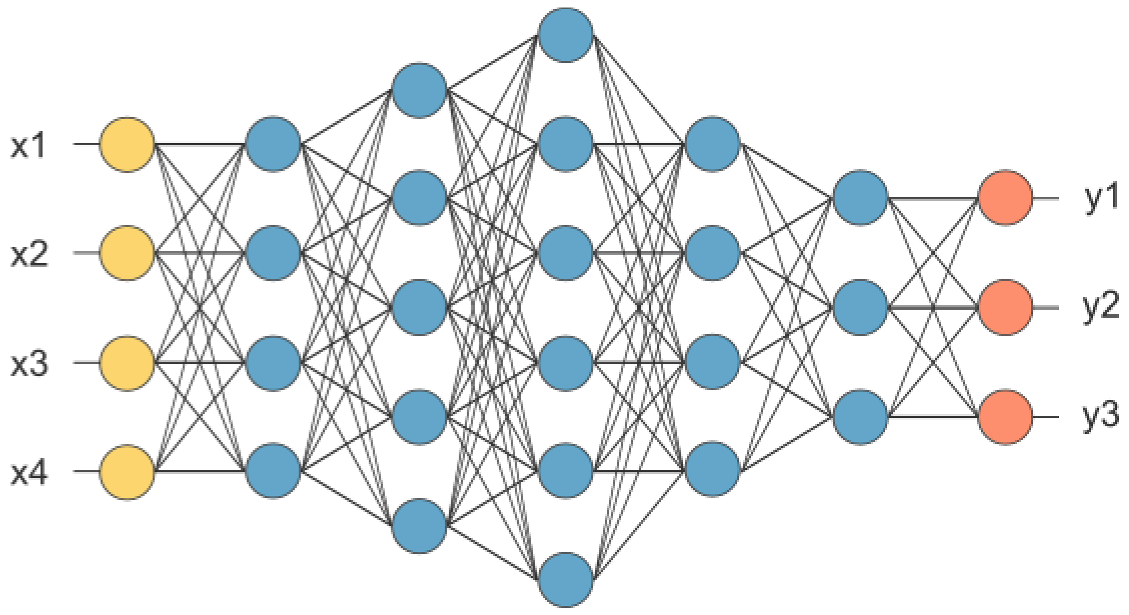
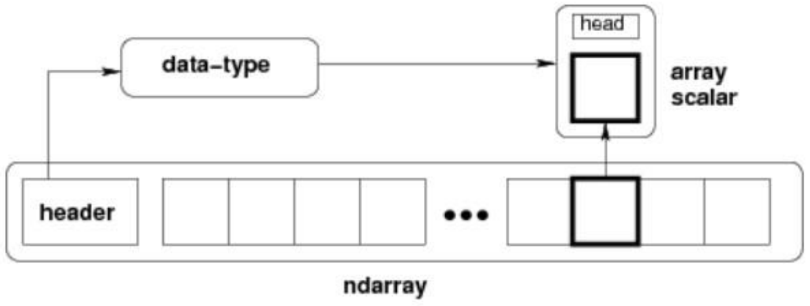


Figure: After pooling layer, flattened as FC layer

Libraries Used

1. 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.



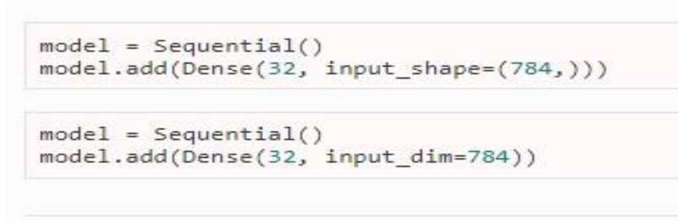
2. Keras.model.Sequential:

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).



```
model = Sequential()
model.add(Dense(32, input_shape=(784,)))
```

```
model = Sequential()
model.add(Dense(32, input_dim=784))
```

Compilation:

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

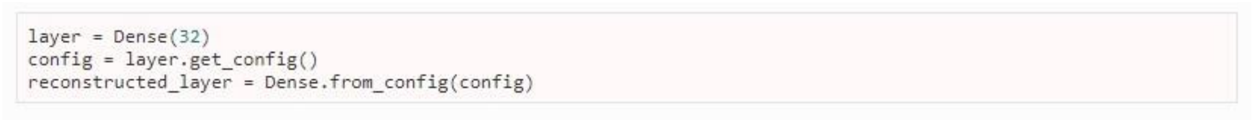
- An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the Optimizer class. See: optimizers.
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (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.

3. 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:



```
layer = Dense(32)
config = layer.get_config()
reconstructed_layer = Dense.from_config(config)
```

4. Open CV

OpenCV (Open Source Computer Vision Library: http://opencv.org) is an open-source BSDlicensed 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 Cbased 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 multidimensional 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.

5. 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.

6. 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.

7. 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.

```
import tkinter
top = tkinter.Tk()
# Code to add widgets will go here...
top.mainloop()
```

Function Used

1. 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

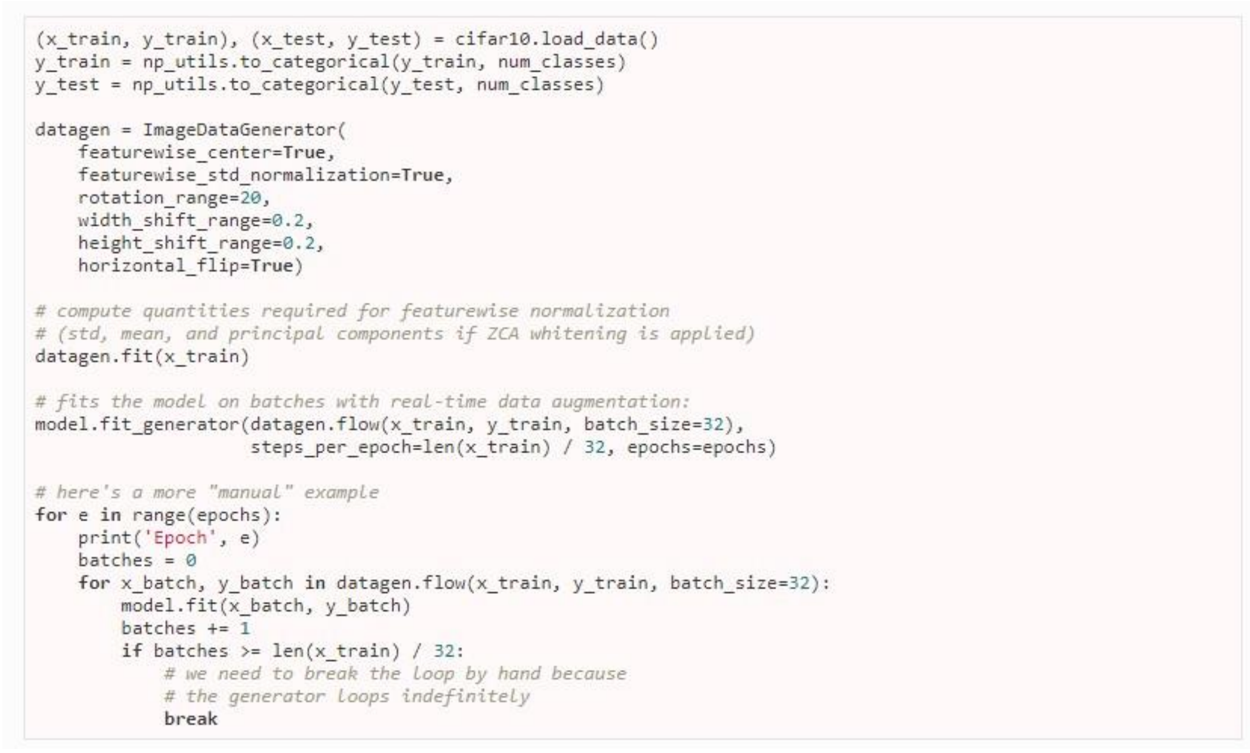
  interval (-width_shift_range, +width_shift_range)

  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)



```
(x_train, y_train), (x_test, y_test) = cifar10.load_data()
y_train = np_utils.to_categorical(y_train, num_classes)
y_test = np_utils.to_categorical(y_test, num_classes)

datagen = ImageDataGenerator(
    featurewise_center=True,
    featurewise_std_normalization=True,
    rotation_range=20,
    width_shift_range=0.2,
    height_shift_range=0.2,
    horizontal_flip=True)

# compute quantities required for featurewise normalization
# (std, mean, and principal components if ZCA whitening is applied)
datagen.fit(x_train)

# fits the model on batches with real-time data augmentation:
model.fit_generator(datagen.flow(x_train, y_train, batch_size=32),
                    steps_per_epoch=len(x_train) / 32, epochs=epochs)

# here's a more "manual" example
for e in range(epochs):
    print('Epoch', e)
    batches = 0
    for x_batch, y_batch in datagen.flow(x_train, y_train, batch_size=32):
        model.fit(x_batch, y_batch)
        batches += 1
        if batches >= len(x_train) / 32:
            # we need to break the loop by hand because
            # the generator loops indefinitely
            break
```

2. 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).

```
# as first layer in a sequential model:
model = Sequential()
model.add(Dense(32, input_shape=(16,)))
# now the model will take as input arrays of shape (*, 16)
# and output arrays of shape (*, 32)

# after the first layer, you don't need to specify
# the size of the input anymore:
model.add(Dense(32))
```

1. 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.

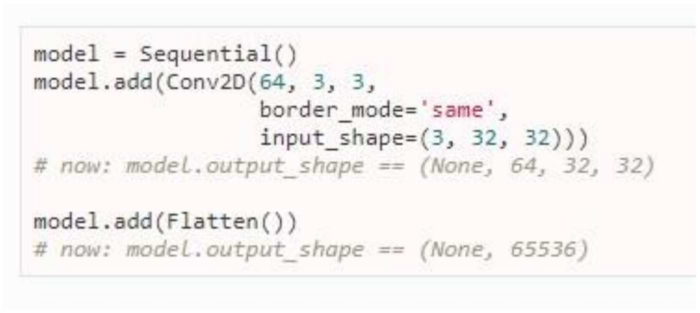
4. Keras.layers.Flatten:

```
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".



```
model = Sequential()
model.add(Conv2D(64, 3, 3,
                 border_mode='same',
                 input_shape=(3, 32, 32)))
# now: model.output_shape == (None, 64, 32, 32)

model.add(Flatten())
# now: model.output_shape == (None, 65536)
```

5. 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".

6. Keras.layers.Maxpooling:

```
 keras.layers.MaxPooling3D(pool_size=(2, 2, 2), strides=None,
padding='valid',

data_format=None)
```

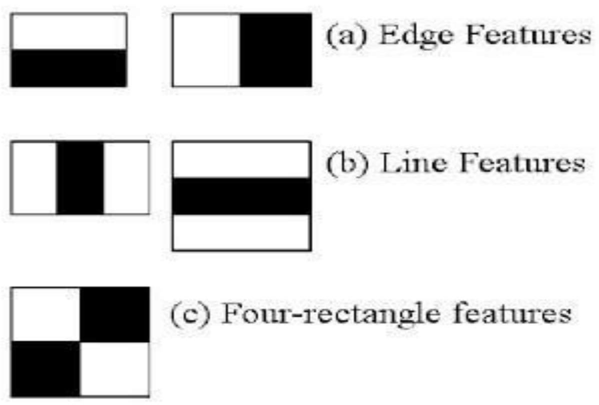
Arguments:

- 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".

7. 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**.

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('haarcascade_frontalface_default.xml') eye_cascade =
cv.CascadeClassifier('haarcascade_eye.xml') img = cv.imread('sachin.jpg')
gray = cv.cvtColor(img, cv.COLOR_BGR2GRAY) faces =
face_cascade.detectMultiScale(gray, 1.3, 5) for (x,y,w,h) in faces:
cv.rectangle(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(roi_color,(ex,ey),(ex+ew,ey+eh),(0,255,0),2)

cv.imshow('img',img) cv.waitKey(0) cv.destroyAllWindows()

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

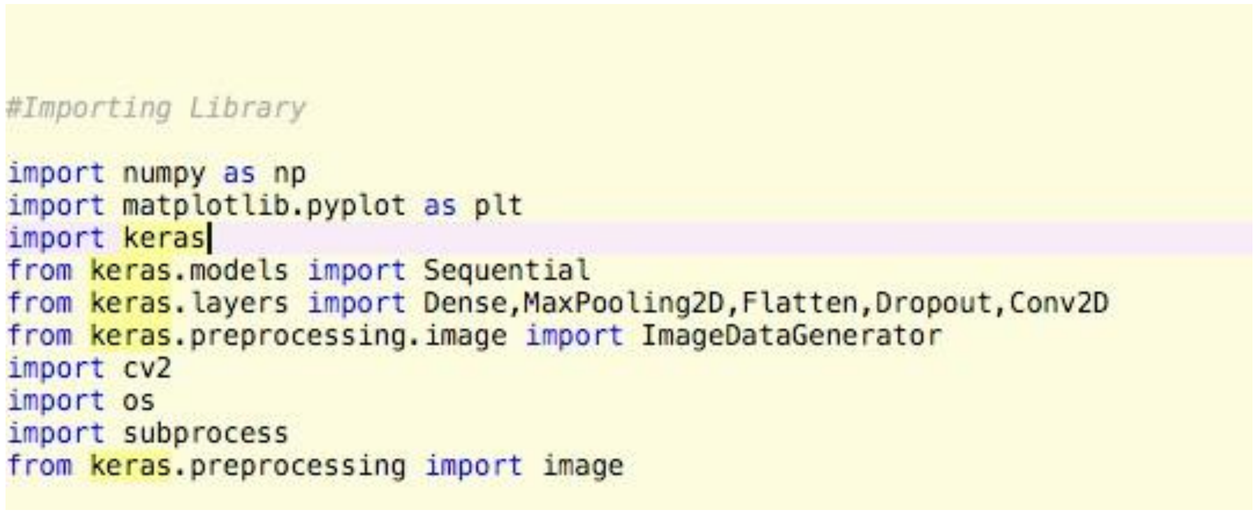
And to read more about dataset you can refer to: http://www.pitt.edu/~emotion/ck-spread.htm

Feature Extraction and Selection :

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

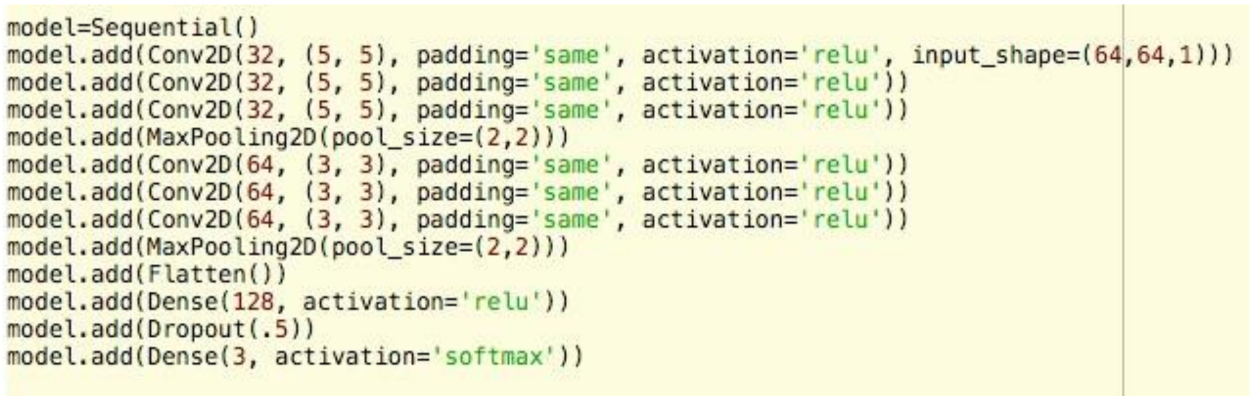
These features are processed by CNN layers and then selected by the algorithm and then they are converted to NumPy array and then model is trained by that and the following three classifications are made.

CODE DESCRIPTION



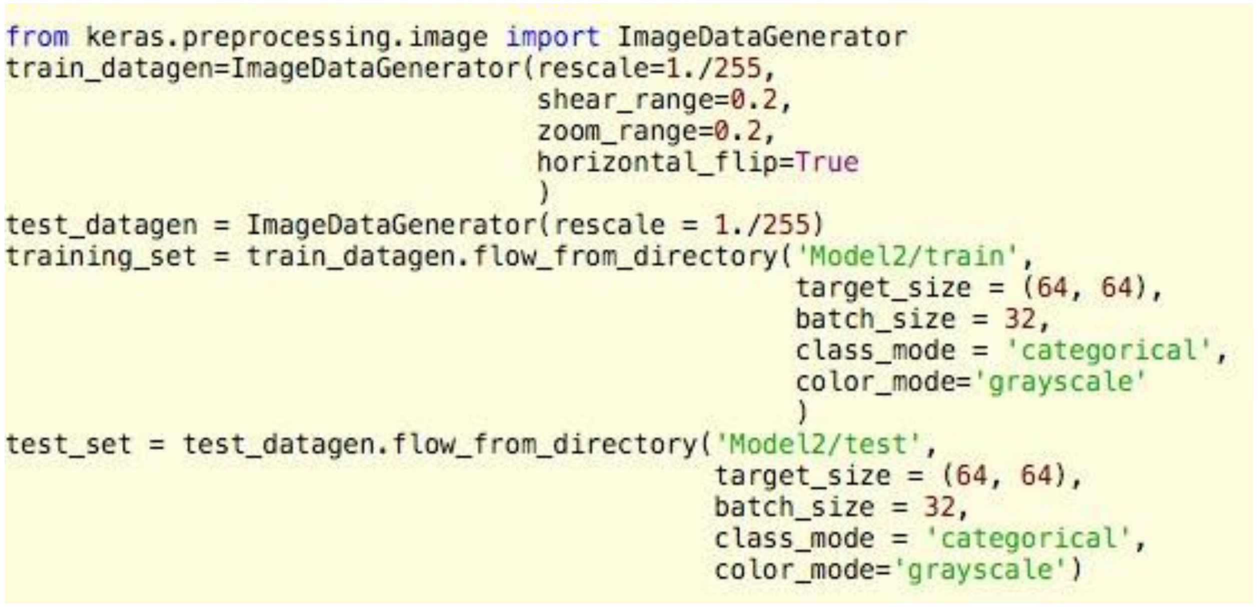
```
#Importing Library

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
```

Figure : All libraries are imported in this .



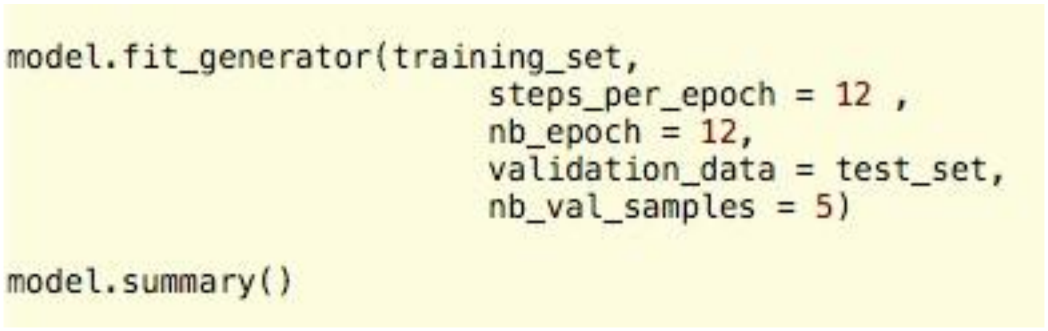
```
model=Sequential()
model.add(Conv2D(32, (5, 5), padding='same', activation='relu', input_shape=(64,64,1)))
model.add(Conv2D(32, (5, 5), padding='same', activation='relu'))
model.add(Conv2D(32, (5, 5), padding='same', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Conv2D(64, (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'))
```

Figure : Model Initialization and building.



```
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')
```

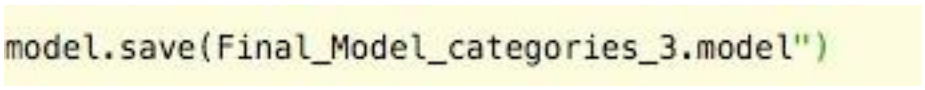
Figure : Training of test and testing.



```
model.fit_generator(training_set,
                         steps_per_epoch = 12 ,
                         nb_epoch = 12,
                         validation_data = test_set,
                         nb_val_samples = 5)

model.summary()
```

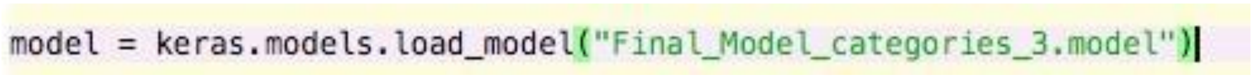
Figure : Training our model

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



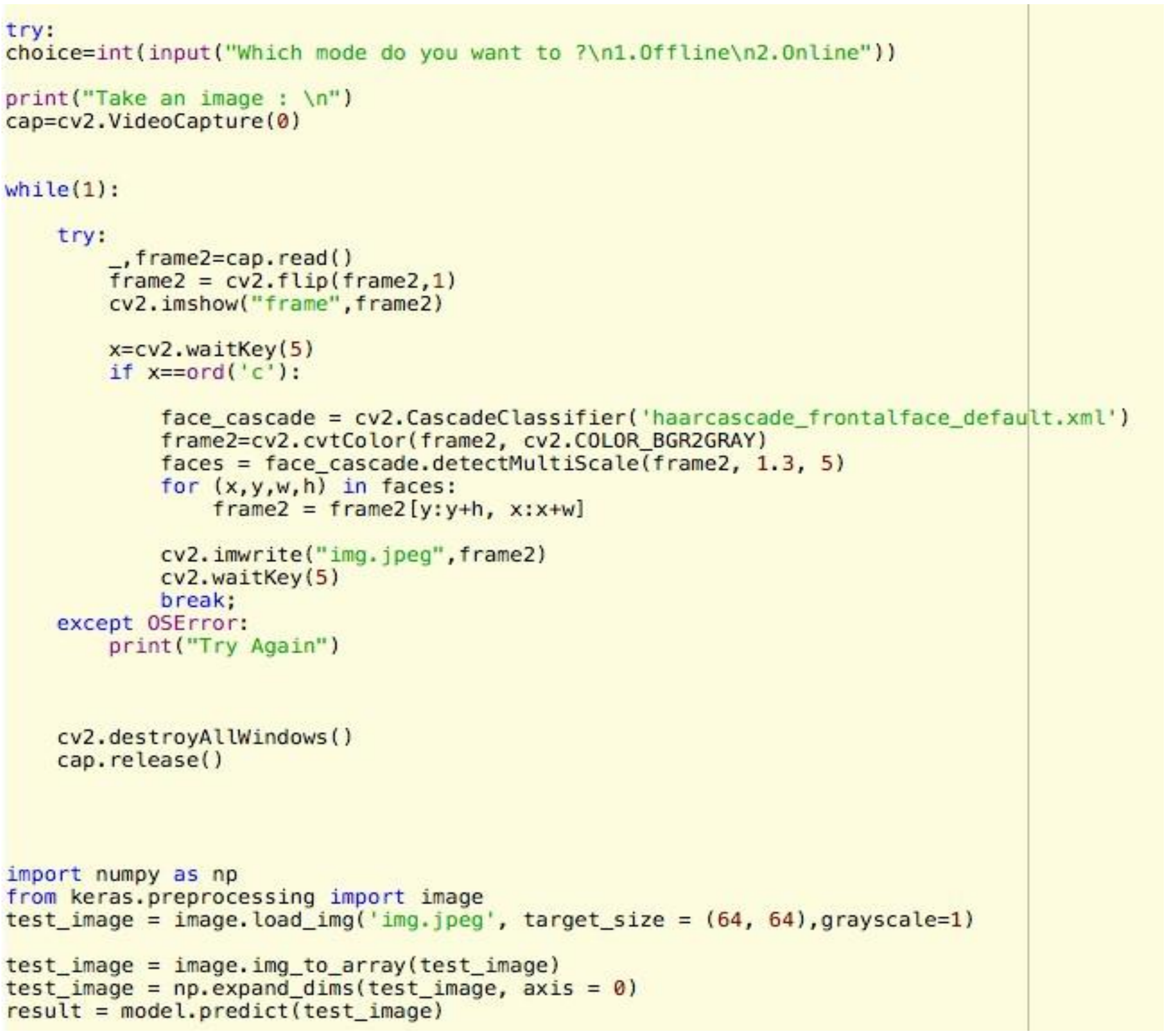
```
model.save(Final_Model_categories_3.model")
```

Figure : Saving a model.



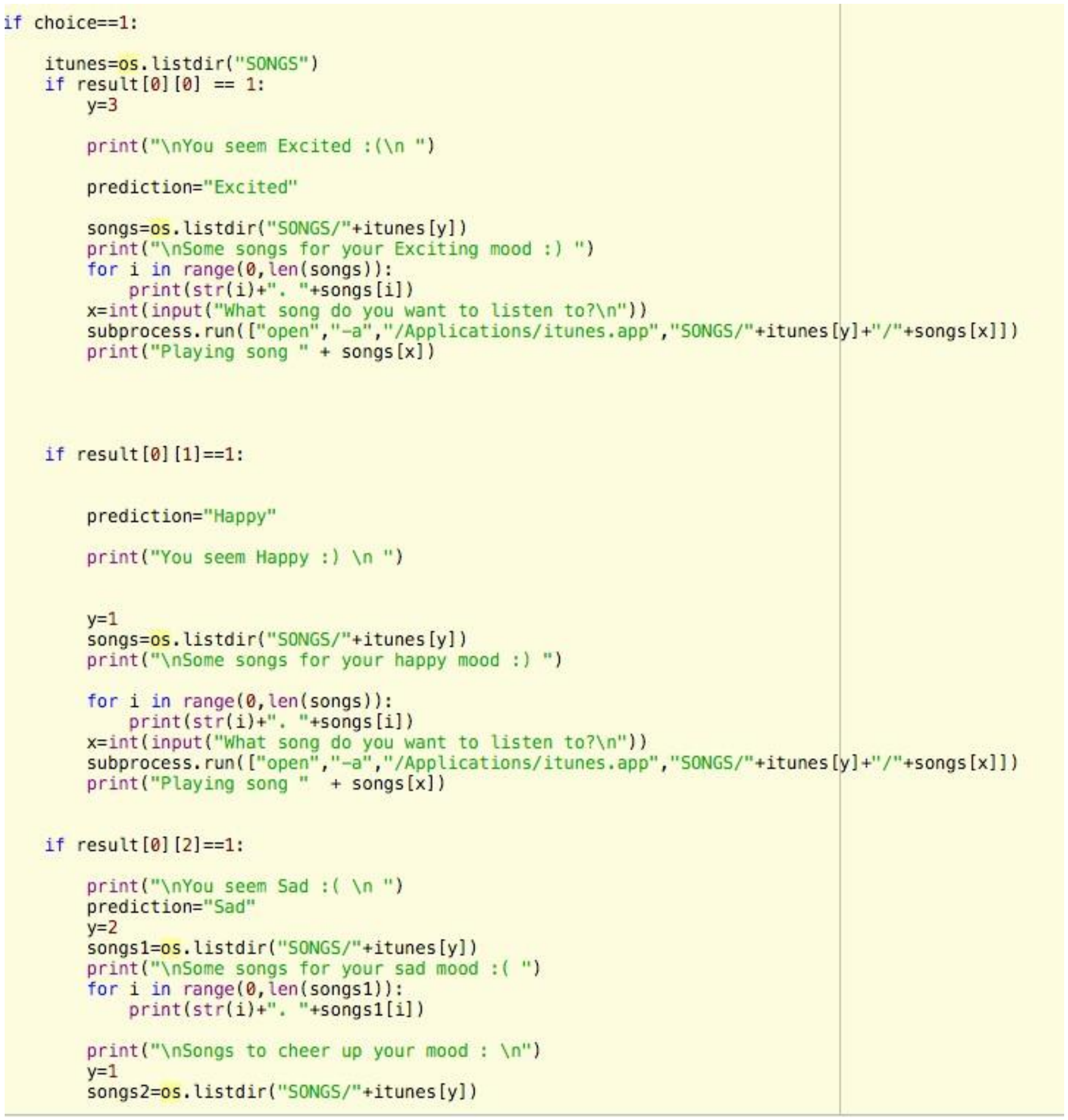
```
model = keras.models.load_model("Final_Model_categories_3.model")
```

Figure : Loading a saved model.



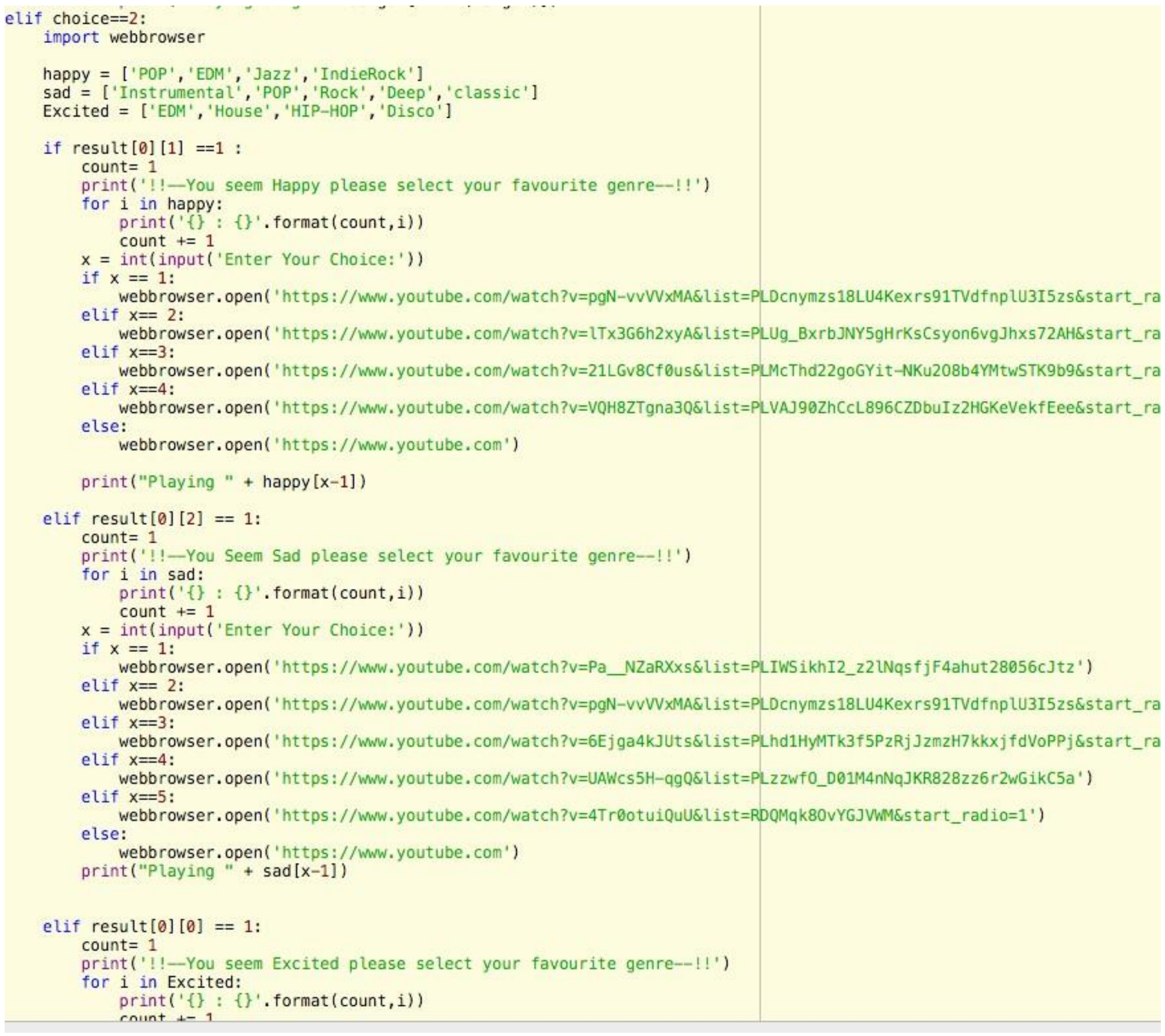
```
try:
choice=int(input("Which mode do you want to ?\n1.Offline\n2.Online"))

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)
```

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



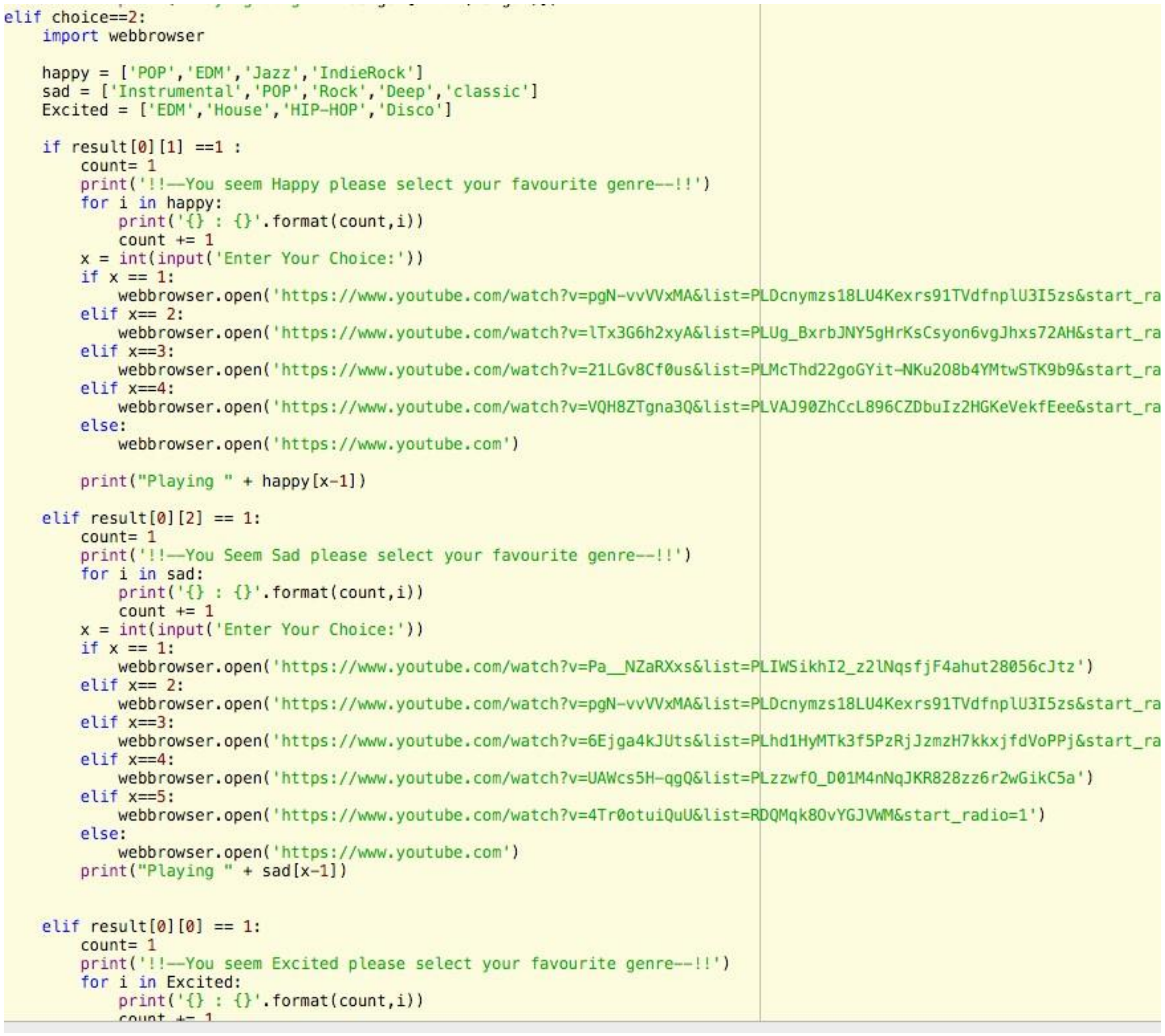
```
if choice==1:

    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])
```

Figure: Suggesting songs in Offline mode



```python
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_ra
        elif x== 2:
            webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start_ra
        elif x==3:
            webbrowser.open('https://www.youtube.com/watch?v=21LGv8Cf0us&list=PLMcThd22goGYit-NKu2O8b4YMtwSTK9b9&start_ra
        elif x==4:
            webbrowser.open('https://www.youtube.com/watch?v=VQH8ZTgna3Q&list=PLVAJ90ZhCcL896CZDbuIz2HGKeVekfEee&start_ra
        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_ra
        elif x==3:
            webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJzmzH7kkxjfdVoPPj&start_ra
        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
```

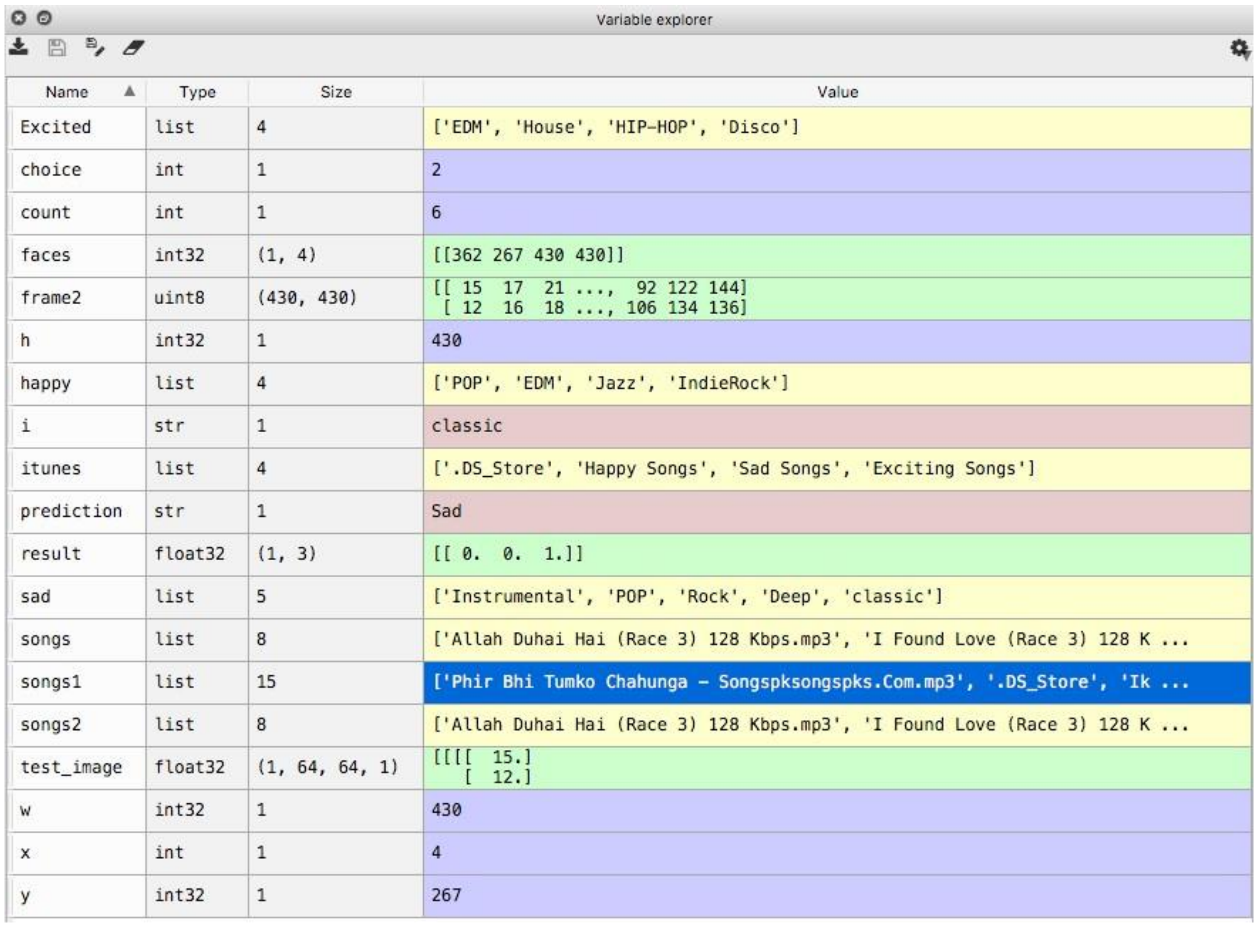
Figure: Suggesting songs online(Youtube)



```
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_ra
        elif x== 2:
            webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg_BxrbJNY5gHrKsCsyon6vgJhxs72AH&start_ra
        elif x==3:
            webbrowser.open('https://www.youtube.com/watch?v=21LGv8Cf0us&list=PLMcThd22goGYit-NKu2O8b4YMtwSTK9b9&start_ra
        elif x==4:
            webbrowser.open('https://www.youtube.com/watch?v=VQH8ZTgna3Q&list=PLVAJ90ZhCcL896CZDbuIz2HGKeVekfEee&start_ra
        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_ra
        elif x==3:
            webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJzmzH7kkxjfdVoPPj&start_ra
        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))
```

Figure : Rest of GUI part

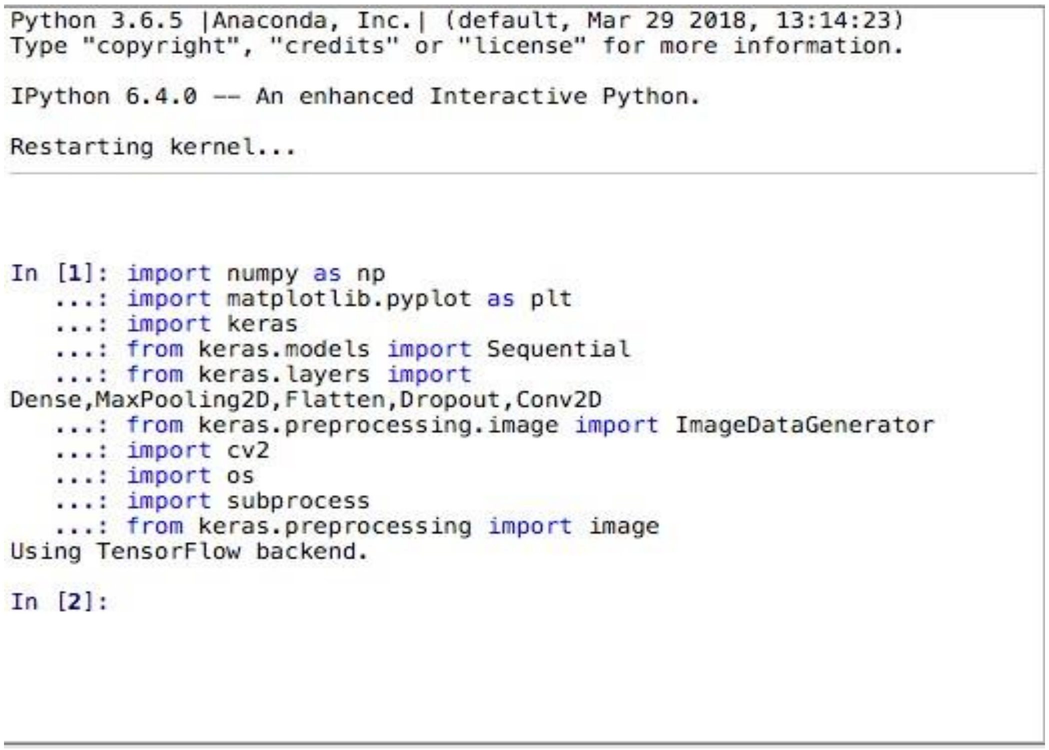
Variable Explorer



Variable explorer

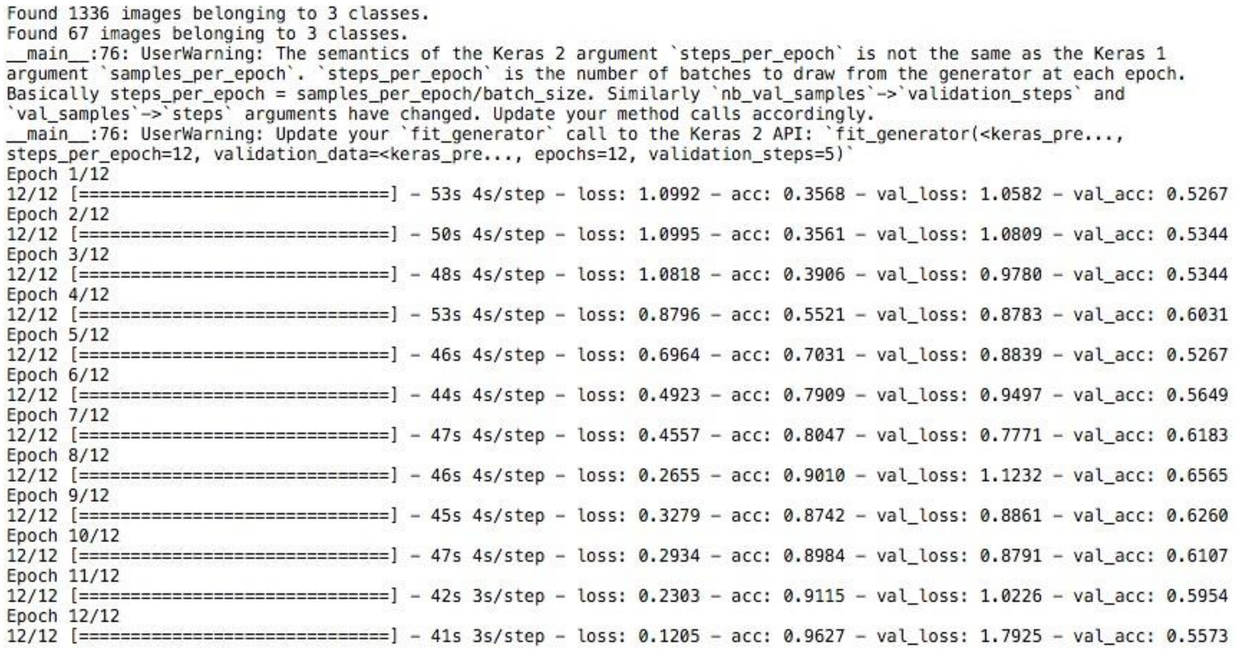
Name	Type	Size	Value
Excited	list	4	['EDM', 'House', 'HIP-HOP', 'Disco']
choice	int	1	2
count	int	1	6
faces	int32	(1, 4)	[[362 267 430 430]]
frame2	uint8	(430, 430)	[[15 17 21 ..., 92 122 144] [12 16 18 ..., 106 134 136]
h	int32	1	430
happy	list	4	['POP', 'EDM', 'Jazz', 'IndieRock']
i	str	1	classic
itunes	list	4	['.DS_Store', 'Happy Songs', 'Sad Songs', 'Exciting Songs']
prediction	str	1	Sad
result	float32	(1, 3)	[[0. 0. 1.]]
sad	list	5	['Instrumental', 'POP', 'Rock', 'Deep', 'classic']
songs	list	8	['Allah Duhai Hai (Race 3) 128 Kbps.mp3', 'I Found Love (Race 3) 128 K ...
songs1	list	15	['Phir Bhi Tumko Chahunga - Songspksongspks.Com.mp3', '.DS_Store', 'Ik ...
songs2	list	8	['Allah Duhai Hai (Race 3) 128 Kbps.mp3', 'I Found Love (Race 3) 128 K ...
test_image	float32	(1, 64, 64, 1)	[[[[15.] [12.]
w	int32	1	430
x	int	1	4
y	int32	1	267

IPython Console



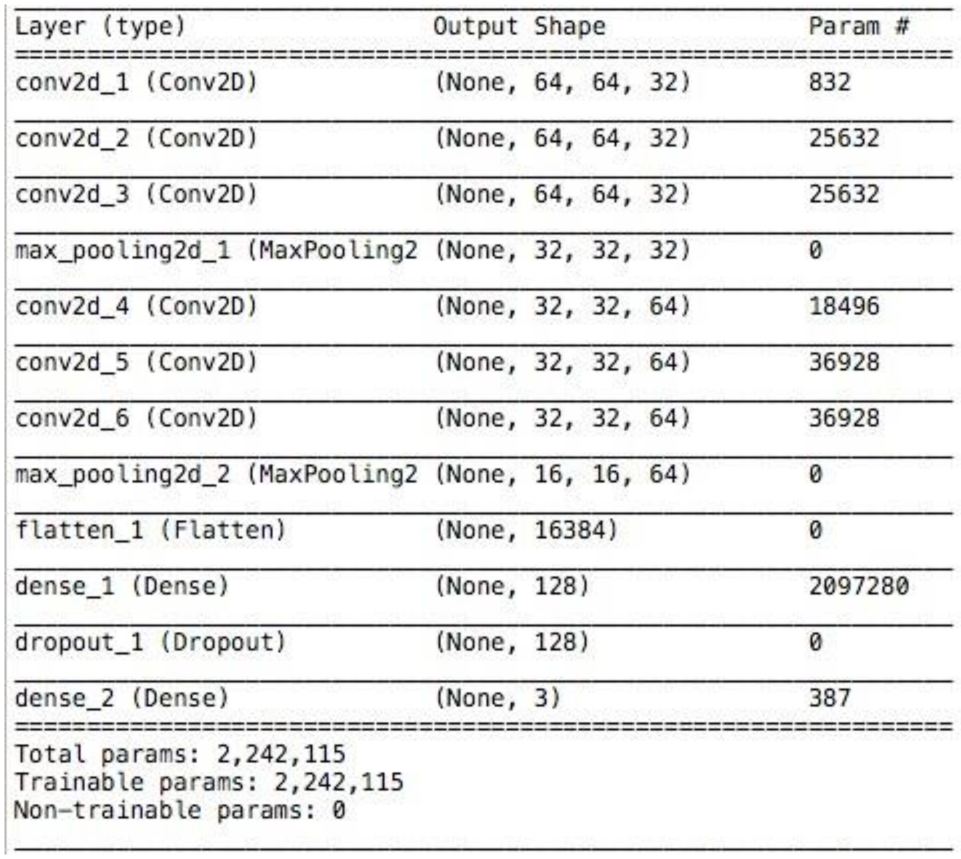
```
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:14:23)
Type "copyright", "credits" or "license" for more information.

IPython 6.4.0 -- An enhanced Interactive Python.

Restarting kernel...


In [1]: 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
Using TensorFlow backend.

In [2]:
```

Figure: Importing Libraries



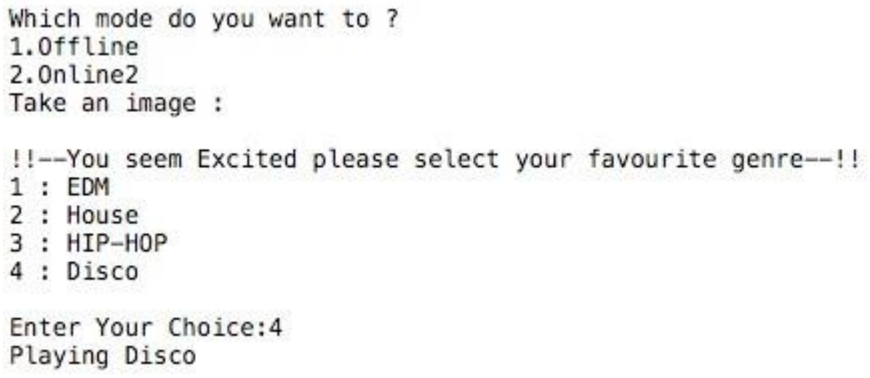
```
Found 1336 images belonging to 3 classes.
Found 67 images belonging to 3 classes.
__main__:76: UserWarning: The semantics of the Keras 2 argument `steps_per_epoch` is not the same as the Keras 1
argument `samples_per_epoch`. `steps_per_epoch` is the number of batches to draw from the generator at each epoch.
Basically steps_per_epoch = samples_per_epoch/batch_size. Similarly `nb_val_samples`->`validation_steps` and
`val_samples`->`steps` arguments have changed. Update your method calls accordingly.
__main__:76: UserWarning: Update your `fit_generator` call to the Keras 2 API: `fit_generator(<keras_pre...,
steps_per_epoch=12, validation_data=<keras_pre..., epochs=12, validation_steps=5)`
Epoch 1/12
12/12 [==============================] - 53s 4s/step - loss: 1.0992 - acc: 0.3568 - val_loss: 1.0582 - val_acc: 0.5267
Epoch 2/12
12/12 [==============================] - 50s 4s/step - loss: 1.0995 - acc: 0.3561 - val_loss: 1.0809 - val_acc: 0.5344
Epoch 3/12
12/12 [==============================] - 48s 4s/step - loss: 1.0818 - acc: 0.3906 - val_loss: 0.9780 - val_acc: 0.5344
Epoch 4/12
12/12 [==============================] - 53s 4s/step - loss: 0.8796 - acc: 0.5521 - val_loss: 0.8783 - val_acc: 0.6031
Epoch 5/12
12/12 [==============================] - 46s 4s/step - loss: 0.6964 - acc: 0.7031 - val_loss: 0.8839 - val_acc: 0.5267
Epoch 6/12
12/12 [==============================] - 44s 4s/step - loss: 0.4923 - acc: 0.7909 - val_loss: 0.9497 - val_acc: 0.5649
Epoch 7/12
12/12 [==============================] - 47s 4s/step - loss: 0.4557 - acc: 0.8047 - val_loss: 0.7771 - val_acc: 0.6183
Epoch 8/12
12/12 [==============================] - 46s 4s/step - loss: 0.2655 - acc: 0.9010 - val_loss: 1.1232 - val_acc: 0.6565
Epoch 9/12
12/12 [==============================] - 45s 4s/step - loss: 0.3279 - acc: 0.8742 - val_loss: 0.8861 - val_acc: 0.6260
Epoch 10/12
12/12 [==============================] - 47s 4s/step - loss: 0.2934 - acc: 0.8984 - val_loss: 0.8791 - val_acc: 0.6107
Epoch 11/12
12/12 [==============================] - 42s 3s/step - loss: 0.2303 - acc: 0.9115 - val_loss: 1.0226 - val_acc: 0.5954
Epoch 12/12
12/12 [==============================] - 41s 3s/step - loss: 0.1205 - acc: 0.9627 - val_loss: 1.7925 - val_acc: 0.5573
```

Figure: Model Training



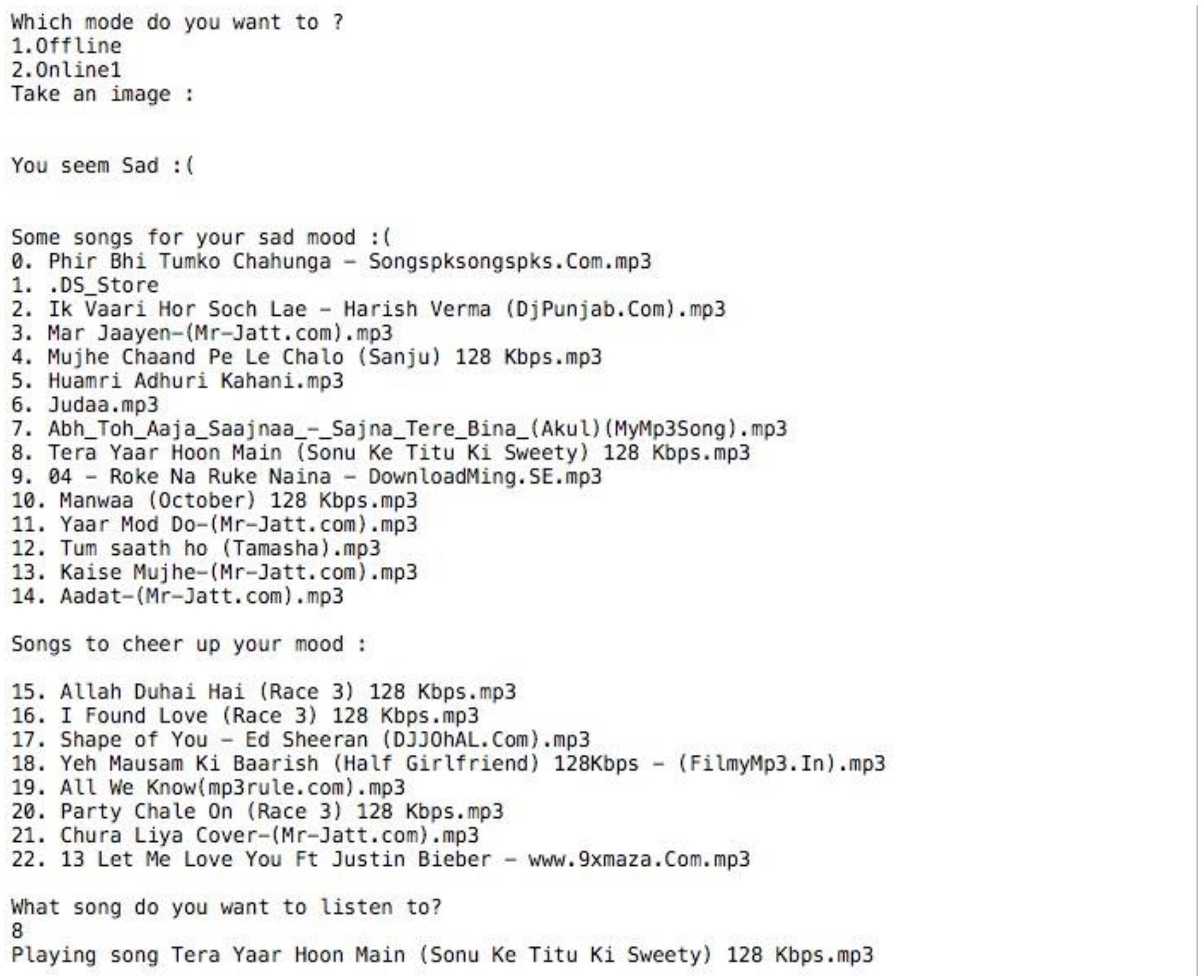
```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 64, 64, 32)        832
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 32)        25632
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 32)        25632
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 32, 32, 64)        18496
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 16384)             0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               2097280
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 3)                 387
=================================================================
Total params: 2,242,115
Trainable params: 2,242,115
Non-trainable params: 0
_________________________________________________________________
```

Figure: Model Summary



```
Which mode do you want to ?
1.Offline
2.Online2
Take an image :

!!--You seem Excited please select your favourite genre--!!
1 : EDM
2 : House
3 : HIP-HOP
4 : Disco

Enter Your Choice:4
Playing Disco
```

Figure: Online Mode



```
Which mode do you want to ?
1.Offline
2.Online1
Take an image :


You seem Sad :(


Some songs for your sad mood :(
0. Phir Bhi Tumko Chahunga - Songspksongspks.Com.mp3
1. .DS_Store
2. Ik Vaari Hor Soch Lae - Harish Verma (DjPunjab.Com).mp3
3. Mar Jaayen-(Mr-Jatt.com).mp3
4. Mujhe Chaand Pe Le Chalo (Sanju) 128 Kbps.mp3
5. Huamri Adhuri Kahani.mp3
6. Judaa.mp3
7. Abh_Toh_Aaja_Saajnaa_-_Sajna_Tere_Bina_(Akul)(MyMp3Song).mp3
8. Tera Yaar Hoon Main (Sonu Ke Titu Ki Sweety) 128 Kbps.mp3
9. 04 - Roke Na Ruke Naina - DownloadMing.SE.mp3
10. Manwaa (October) 128 Kbps.mp3
11. Yaar Mod Do-(Mr-Jatt.com).mp3
12. Tum saath ho (Tamasha).mp3
13. Kaise Mujhe-(Mr-Jatt.com).mp3
14. Aadat-(Mr-Jatt.com).mp3

Songs to cheer up your mood :

15. Allah Duhai Hai (Race 3) 128 Kbps.mp3
16. I Found Love (Race 3) 128 Kbps.mp3
17. Shape of You - Ed Sheeran (DJJOhAL.Com).mp3
18. Yeh Mausam Ki Baarish (Half Girlfriend) 128Kbps - (FilmyMp3.In).mp3
19. All We Know(mp3rule.com).mp3
20. Party Chale On (Race 3) 128 Kbps.mp3
21. Chura Liya Cover-(Mr-Jatt.com).mp3
22. 13 Let Me Love You Ft Justin Bieber - www.9xmaza.Com.mp3

What song do you want to listen to?
8
Playing song Tera Yaar Hoon Main (Sonu Ke Titu Ki Sweety) 128 Kbps.mp3
```

Figure: Offline Mode

GUI

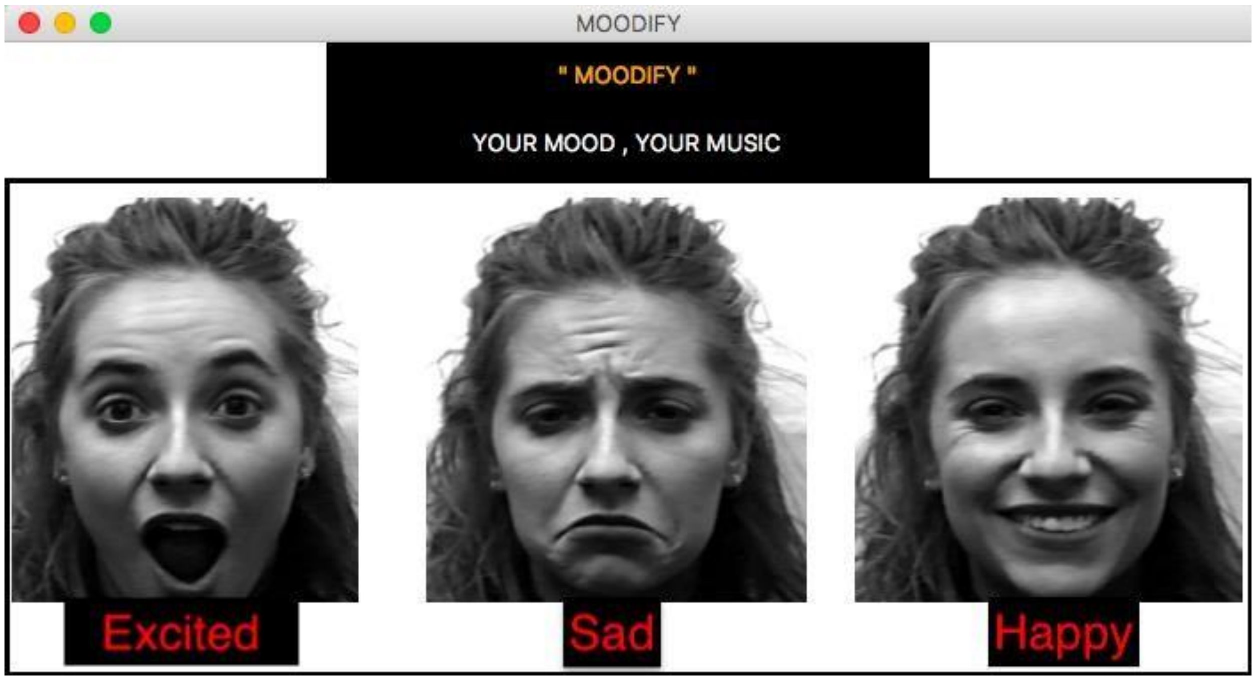
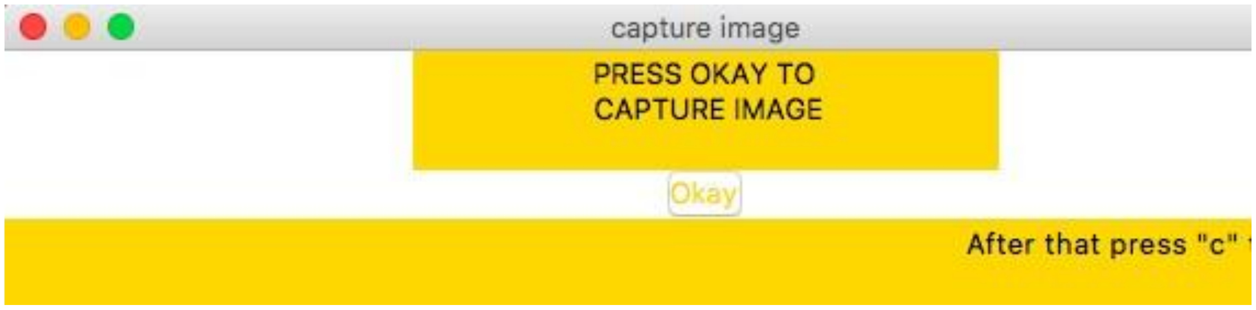
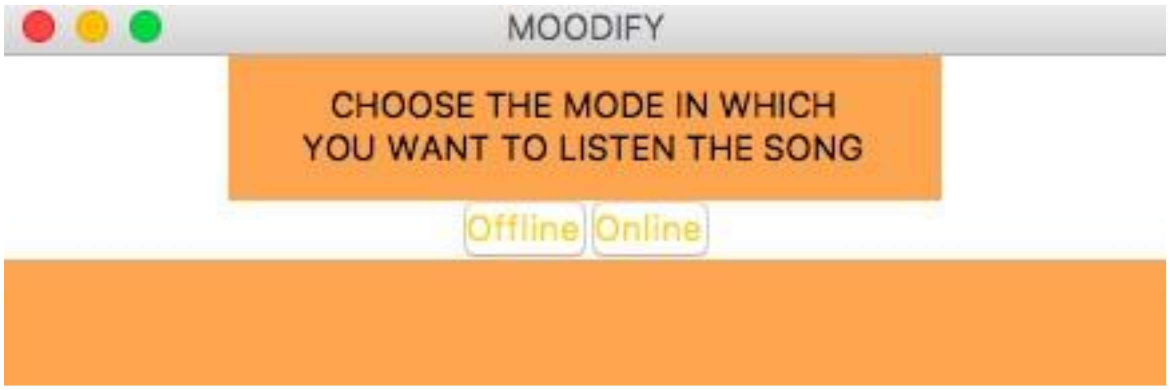


Figure : Splash Screen



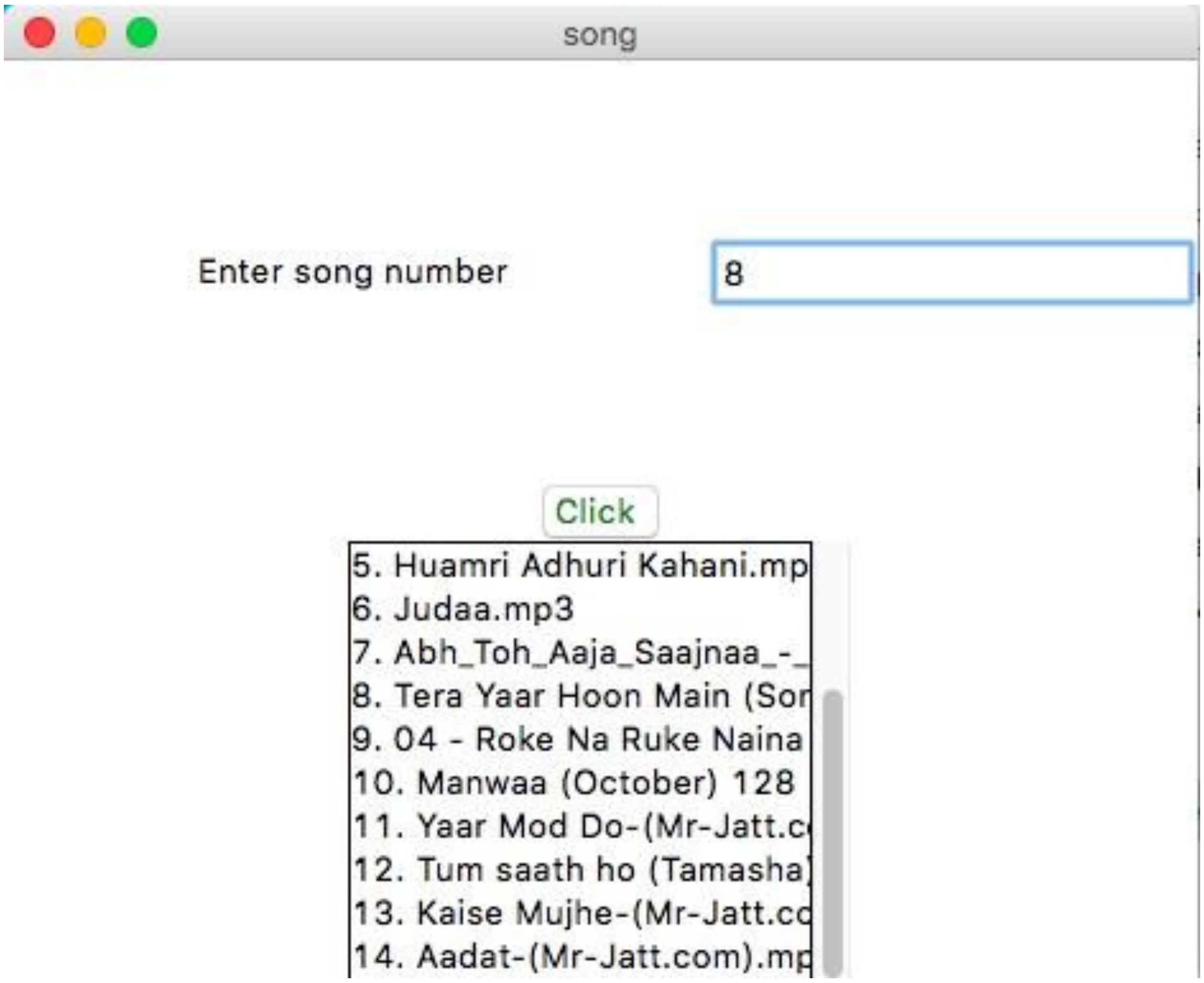
After that press "c"

Figure: Main Screen


MOODIFY

CHOOSE THE MODE IN WHICH
YOU WANT TO LISTEN THE SONG

Figure : Selection screen


song

Enter song number: 8

```
5. Huamri Adhuri Kahani.mp
6. Judaa.mp3
7. Abh_Toh_Aaja_Saajnaa_-_
8. Tera Yaar Hoon Main (Son
9. 04 - Roke Na Ruke Naina
10. Manwaa (October) 128
11. Yaar Mod Do-(Mr-Jatt.c
12. Tum saath ho (Tamasha)
13. Kaise Mujhe-(Mr-Jatt.co
14. Aadat-(Mr-Jatt.com).mp
```

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
"""

#%%



#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=pgNvvVVxMA&list
=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start_radio=1')         elif x== 2:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg_BxrbJNY5gHrK
sCsyon6vgJhxs72AH&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=PLVAJ90ZhCcL896
CZDbuIz2HGKeVekfEee&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_z2lNqsfj
F4ahut28056cJtz')         elif x== 2:

webbrowser.open('https://www.youtube.com/watch?v=pgNvvVVxMA&list
=PLDcnymzs18LU4Kexrs91TVdfnplU3I5zs&start_radio=1')         elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJ
zmzH7kkxjfdVoPPj&start_radio=12')         elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=UAWcs5HqgQ&li
st=PLzzwfO_D01M4nNqJKR828zz6r2wGikC5a')         elif x==5:

webbrowser.open('https://www.youtube.com/watch?v=4Tr0otuiQuU&list=RDQMqk8OvYGJVW
M&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_BxrbJNY5gHrK
sCsyon6vgJhxs72AH&start_radio=1')        elif x== 2:
            webbrowser.open('https://www.youtube.com/watch?v=BDocp-
VpCwY&list=PLhInz4MOzRUsuBj8wF6383E7zm2dJfqZ&start_radio=1')        elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=xTlNMmZKwpA&start_radio=1&list=PL
H6pfBXQXHEC2uDmDy5oi3tHW6X8kZ2Jo')
        elif x==4:

webbrowser.open('https://www.youtube.com/watch?v=kJQP7kiw5Fk&list=PL64E6BD94546734
D8')
        else:
            webbrowser.open('https://www.youtube.com')
        print("Playing " + Excited[x-1])
else:

webbrowser.open('https://www.youtube.com/watch?v=aJOTlE1K90k&list=PLwVjHDlEOgvtnnnq
WlTqByAtC7tXBg6D')
```

2. 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=pgNvvVVxMA&list=PLDcnymzs18LU4Kexrs
91TVdfnplU3I5zs&start_radio=1')

       elif x== 1:

webbrowser.open('https://www.youtube.com/watch?v=lTx3G6h2xyA&list=PLUg_BxrbJNY5gHrKsCsy
on6vgJhxs72AH&start_radio=1')

       elif x==2:
```

```
webbrowser.open('https://www.youtube.com/watch?v=21LGv8Cf0us&list=PLMcThd22goGYitNKu2
O8b4YMtwSTK9b9&start_radio=1')
        elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=VQH8ZTgna3Q&list=PLVAJ90ZhCcL896CZD
buIz2HGKeVekfEee&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_z2lNqsfjF4a
hut28056cJtz')
        elif x== 1:

webbrowser.open('https://www.youtube.com/watch?v=pgNvvVVxMA&list=PLDcnymzs18LU4Kexrs
91TVdfnplU3I5zs&start_radio=1')
        elif x==2:

webbrowser.open('https://www.youtube.com/watch?v=6Ejga4kJUts&list=PLhd1HyMTk3f5PzRjJzmz
H7kkxjfdVoPPj&start_radio=12')
        elif x==3:

webbrowser.open('https://www.youtube.com/watch?v=UAWcs5HqgQ&list=PLzzwfO_D01M4nNqJK
R828zz6r2wGikC5a')
        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_BxrbJNY5gHrKsCsy
on6vgJhxs72AH&start_radio=1')
        elif x== 1:
            webbrowser.open('https://www.youtube.com/watch?v=BDocp-
VpCwY&list=PLhInz4MOzRUsuBj8wF6383E7zm2dJfqZ&start_radio=1')
        elif x==2:

webbrowser.open('https://www.youtube.com/watch?v=xTlNMmZKwpA&start_radio=1&list=PLH6pf
BXQXHEC2uDmDy5oi3tHW6X8kZ2Jo')
        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=PLwVjHDlEOgvtnnnqWlT
qByAtC7tXBg6D')
```

```
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

I 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.