**Name: Nikhil H. Sahu**

FACIAL RECOGNITION

AIM: **To create an application as a interface which utilizes expface.h5 model file for facial emotion recognition to give prediction in live feed video or webcam (application of Computer Vision).**

## ****What is the purpose of the project?****

* The main purpose of the project is to detect faces and recognize emotion associated with it.
* The objective is to create a Computer Vision which tells emotions in live feed.
* One of the challenges is to predict each faces in every frame.
* The application will get used for reading customer experience in something.
* The application will be used as a research about human emotions.

## ****What are the advantages of using python instead of doing manually?****

* **Python has tensorflow framework which provides deep learning concept implementation.**
* **Python is adaptive, versatile and user-friendly and it focuses on readability.**
* Python allows you to take the best of different paradigms of programming.
* Python is open source and writing code in a matter of minutes.
* It has all the libraries you can imagine that means more support.
* It also performs automatic memory management.
* Interactive, Interpreted, Modular, Dynamic, Object-oriented, Portable, High Level, Extensible in C and C++.

# TOOLS USED:

## Python 3.7.4:

* Python 3.7. 4, documentation released on 08 July 2019.
* Python is a dynamic object-oriented programming language that can be used for many kinds of software development.
* Many Python programmers report substantial productivity gains and feel the language encourages the development of higher quality, more maintainable code.
* It offers strong support for integration with other languages and tools, comes with extensive standard libraries, and can be learned in a few days.
* Python is distributed under an OSI-approved open source license that makes it free to use, even for commercial products.
* Python runs on Windows, Linux/Unix, Mac OS X, OS/2, Amiga, Palm Handhelds, and Nokia mobile phones. Python has also been ported to the Java and .NET virtual machines.
* Python programming is widely used in Artificial Intelligence, Natural Language Generation, Neural Networks and other advanced fields of Computer Science.

## IDLE (Python 3.7 64-bit)

* IDLE is Integrated Development and Learning Environment or Integrated Development Environment.
* IDLE is intended to be a simple IDE and suitable for beginners, especially in an educational environment. To that end, it is cross-platform, and avoids feature clutter.
* It is packaged as an optional part of the Python packaging with many Linux distributions.
* Multi-window text editor with syntax highlighting, autocompletion, smart indent and other.
* Integrated debugger with stepping, persistent breakpoints, and call stack visibility.
* It is completely written in Python and the Tkinter GUI toolkit (wrapper functions for T It is completely written in Python and the [Tkinter](https://en.wikipedia.org/wiki/Tkinter) GUI toolkit (wrapper functions for [Tcl](https://en.wikipedia.org/wiki/Tcl)/[Tk](https://en.wikipedia.org/wiki/Tk_(framework)).cl/Tk).

# LIBRARIES USED:

1. **import cv2**

* OpenCV-Python is a library of Python bindings designed to solve computer vision problems.
* cv2.imread() method loads an image from the specified file.
* Training OpenCV was started at Intel in 1999 by Gary Bradsky, and the first release came out in 2000.
* OpenCV-Python is a Python wrapper for the original OpenCV C++ implementation.

1. **import uuid**

* The UUID, Universal Unique Identifier, is a python library which helps in generating random objects of 128 bits as ids.
* It provides the uniqueness as it generates ids on the basis of time, Computer hardware (MAC etc.).
* Can be used as general utility to generate unique random id.
* Can be used in cryptography and hashing applications.
* Useful in generating random documents, addresses etc.

## from datetime import datetime

* A date in Python is not a data type of its own, but we can import a module named datetime to work with dates as date objects.
* Datetime module comes built into Python, so there is no need to install it externally.
* The date – An idealized naive date, assuming the current Gregorian calendar always was, and always will be, in effect. Its attributes are year, month and day.
* The time – An idealized time, independent of any particular day, assuming that every day has exactly 24\*60\*60 seconds. Its attributes are hour, minute, second, microsecond, and tzinfo.
* The datetime – Its a combination of date and time along with the attributes year, month, day, hour, minute, second, microsecond, and tzinfo.
* The timedelta – A duration expressing the difference between two date, time, or datetime instances to microsecond resolution.

## from face\_emocpython37 import predict

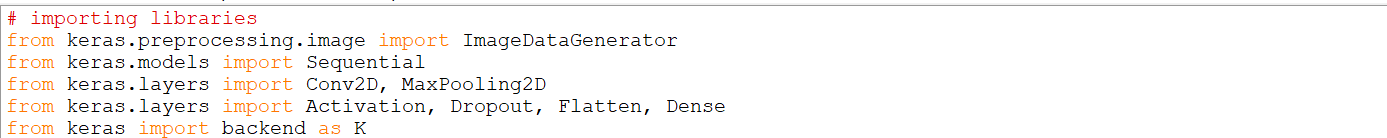
* Dropout It is a compiled python file from which predict() function is imported
* This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass.
* Activation functions are mathematical equations that determine the output of a neural network.
* The function is attached to each neuron in the network, and determines whether it should be activated (“fired”) or not, based on whether each neuron’s input is relevant for the model’s prediction.
* Dense layer is the regular deeply connected neural network layer.
* The output shape of the Dense layer will be affected by the number of neuron / units specified in the Dense layer.
* In between the convolutional layer and the fully connected layer, there is a ‘Flatten’ layer.
* Flattening transforms a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier.

## from keras import backend as K

* We should start by creating a TensorFlow session and registering it with Keras.
* This means that Keras will use the session we registered to initialize all variables that it creates internally.
* In this case, we use Keras only as a syntactical shortcut to generate an op that maps some tensor(s) input to some tensor(s) output, and that's it.
* The optimization is done via a native TensorFlow optimizer rather than a Keras optimizer. We don't even use any Keras Model at all!
* Keras manages a global state, which it uses to implement the Functional model-building API and to uniquify autogenerated layer names.

# Code Explanation:

## Importing Libraries:



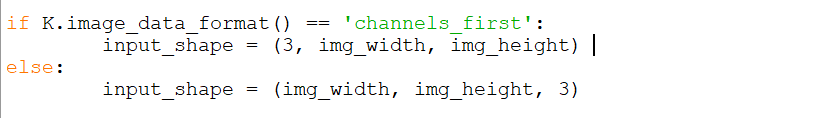
Importing all the required libraries from keras library.

## Defining Necessities



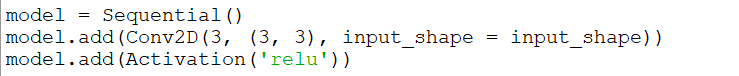
* **Defining image width and height for training data and validation data.**
* **Training and validation directory is defined for accessing images.**
* **No. of training samples, no. of validation samples, epoch and batch size is defined.**

## ****K.image\_data\_format()****



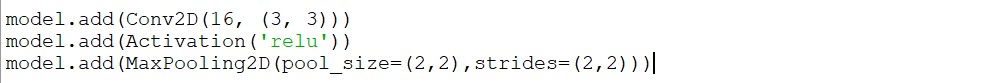
If K.image\_data\_format() contains no. of channels first in the written format, then 3 i.e. no. of channels (r,g,b channels) is written first along with image width and height.

## Layer design #1



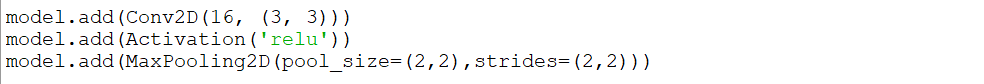
* Creating object of Sequential() class.
* Convolution is done by Conv2D. 3 kernels or nodes of size 3\*3 does the work of convolution given input\_shape is provided. 3 nodes are present in this single hidden layer.se None to keep searching forever.
* Activation function ‘relu’ is applied to each node.

## Layer design #2



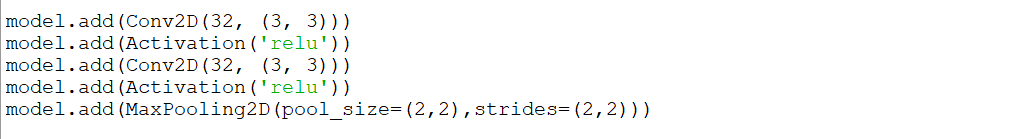
* Convolution layer having 16 nodes of kernel size 3\*3.
* Activation on all nodes.
* Maxpooling reduces pixels matrix size by taking consideration of only large pixel values.

## Layer design #3



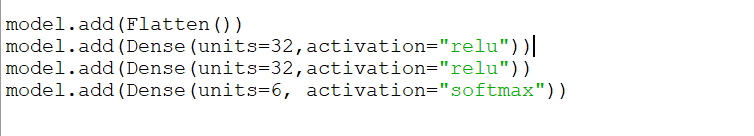
Same as Layer design #2.

## Layer design #4



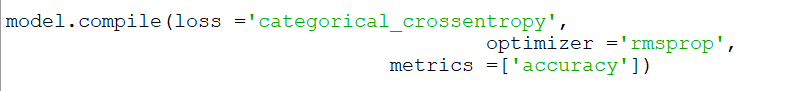
* Convolution: 32, 3\*3
* Activation: relu
* Convolution: 32, 3\*3
* Activation: relu
* Maxpooling: size 2\*2, moves 2 bit right.

1. **Layer design #5**



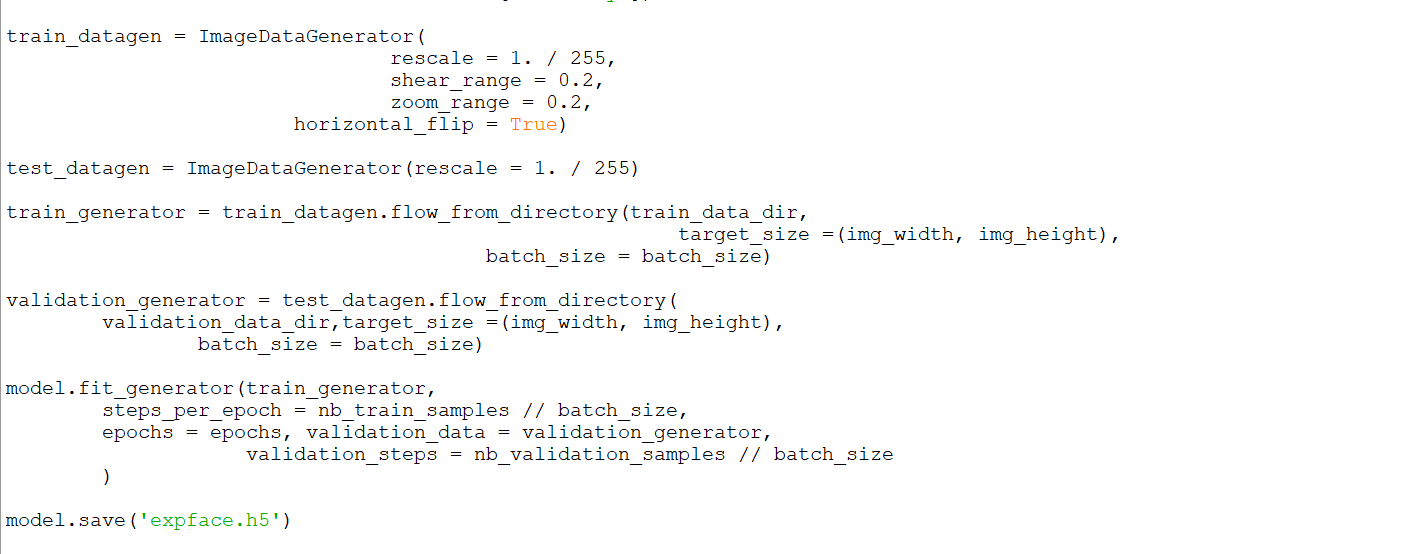
* This is how Flatten works converting Matrix to single array.
* Dense layer is also called fully connected layer and works as same.
* Last activation fuction is softmax for categorical classification.

1. **Compile**



* Compiles the model for execution.
* Output will categorical because more than 2 input classes are there. Hence, categorical\_crossentropy.
* Optimizer will update weights in every forward and backward propogation.
* We have to update weights in order to increase accuracy.

1. **Loading Images and fitting to model**



* train\_datagen and test\_datagen are objects of ImageDataGenerator.
* Flow\_from\_directory function are applied on these objects to load images from directory to the variables.
* Model.fit\_generator initializes neural network by providing images to the neural network and defines running steps parameters.
* Finally model is saved by name expface.h5.

# CODE (in text):

* 1. EXPERIMENTAL.PY

# importing libraries

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Sequential

from keras.layers import Conv2D, MaxPooling2D

from keras.layers import Activation, Dropout, Flatten, Dense

from keras import backend as K

#

img\_width, img\_height = 224, 224

train\_data\_dir = 'data/train'

validation\_data\_dir = 'data/test'

nb\_train\_samples = 1200

nb\_validation\_samples = 300

epochs = 10

batch\_size = 16

if K.image\_data\_format() == 'channels\_first':

input\_shape = (3, img\_width, img\_height)

else:

input\_shape = (img\_width, img\_height, 3)

model = Sequential()

model.add(Conv2D(3, (3, 3), input\_shape = input\_shape))

model.add(Activation('relu'))

model.add(Conv2D(16, (3, 3)))

model.add(Activation('relu'))

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

model.add(Conv2D(16, (3, 3)))

model.add(Activation('relu'))

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

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

model.add(Activation('relu'))

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

model.add(Activation('relu'))

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

model.add(Flatten())

model.add(Dense(units=32,activation="relu"))

model.add(Dense(units=32,activation="relu"))

model.add(Dense(units=6, activation="softmax"))

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

optimizer ='rmsprop',

metrics =['accuracy'])

train\_datagen = ImageDataGenerator(

rescale = 1. / 255,

shear\_range = 0.2,

zoom\_range = 0.2,

horizontal\_flip = True)

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

train\_generator = train\_datagen.flow\_from\_directory(train\_data\_dir,

target\_size =(img\_width, img\_height),

batch\_size = batch\_size)

validation\_generator = test\_datagen.flow\_from\_directory(

validation\_data\_dir,target\_size =(img\_width, img\_height),

batch\_size = batch\_size)

model.fit\_generator(train\_generator,

steps\_per\_epoch = nb\_train\_samples // batch\_size,

epochs = epochs, validation\_data = validation\_generator,

validation\_steps = nb\_validation\_samples // batch\_size

)

model.save('expface.h5')

* 1. DETECT.PY

import keras

from keras.models import \*

from keras.models import Sequential

import cv2

import numpy as np

img\_width, img\_height = 224, 224

input\_shape = (img\_width, img\_height, 3)

model = Sequential()

model =load\_model('expface.h5')

img = cv2.imread('download\_result.jpg')

img = cv2.resize(img,(224,224))

img = np.reshape(img,[1,224,224,3])

classes = np.argmax(model.predict(img), axis=-1)

#print(classes[0])

a=classes

if(a[0]==1):

print('Angry')

elif(a[0]==2):

print('Happy')

elif(a[0]==3):

print('Surprise')

elif(a[0]==4):

print('Disgust')

elif(a[0]==5):

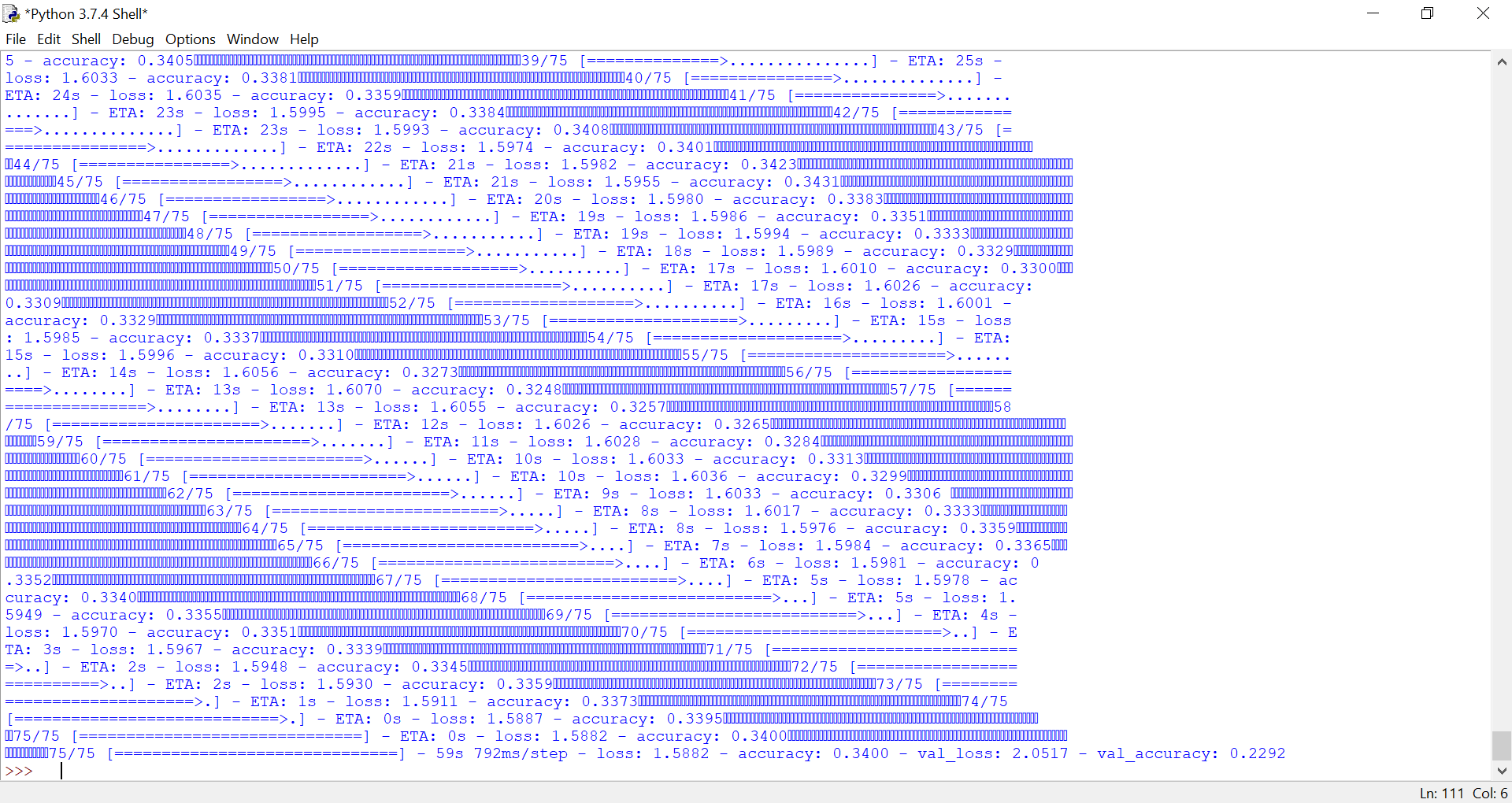
print('Sad')

elif(a[0]==6):

print('Fear')

# OUTPUT:

1. EXPERIMENTAL.PY



1. DETECT.PY

INPUT IMAGE:



