School of Computer Science and Engineering

REAL TIME FACE EMOTION DETECTION

A project submitted
in partial fulfillment of the requirements for the degree of Bachelor of Technology (CSE)

By:

1)B.Pruna Praveen(AP18110010379)
2)Jenvith Manduva(AP18110010418)
3)Ramaneesh Valisetty (AP18110010446)

ABSTRACT:

Facial emotions or expressions can be recognized by computers and enhancing modern day machines to understand human emotions from their real life time.Through this project, i want to provide solution for real face expressions or emotions by video capture from emotion detector frame by Open-CV it will capture video by camera which is built-in to the machine or computer system. . The facial features are identified by different operations provided by OpenCV and the region consisting of parts of the face are made to surround or enclose by a contour. This region, enclosed by the contour is used as an input to the Convolutional Neural Network (CNN).The CNN model created consists of six activation layers, of which four are convolution layers and two are fully controlled layers.The scope of the project is to
demonstrate the accuracy and validation of Convolution Neural Network(CNN).

INTRODUCTION:

Human beings will communicate with each other by speech,actions and expressions (or) emotions.Facial expression recognition has its branches spread across various applications such as virtual reality,webinar technologies,online surveys and many other fields. Even though there are several advantages has been witnessed in this field,but there are several di-advantages that exist.The traditional features extraction methods shows very low response and lack in performance.For these traditional features extraction, it is very difficult to extract the required features and it is hazardous to us in real life. These emotions can be detected by machine or computer with respect to artificial intelligence and etc.In this project we are deducting expressions by Convolution neural networks(CNN).The facial expressions recognition is done by keras.The CNN will used to train the model.The trained model is deployed to a emotion detector by tensorflow frame.This developed model is used for real time faces,images and videos and its accuracy and validation is also analyzed.

OBJECTIVES:

The objective for this project are:-

1)Keggal(for data set)

2)keggal kernel(for train the data)

3) Virtual studio code(for emotion detector frame)

4)haarcascade-frontal face-default(for face recognition)

5)Convolution neural network(to train data)

6)Install keras

7)Install numpy

8)Install open CV

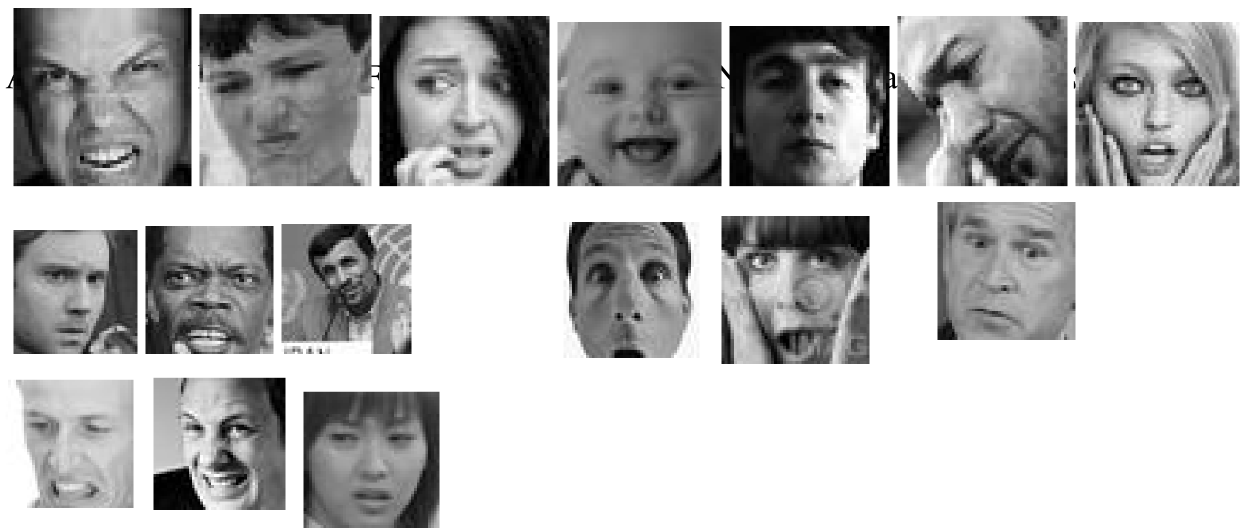
9) InstallTensor
flow 10)Install
Pandas 11)install
seaborn

_DATA SET:

In this project, the database used to train the mode is taken from the keggal website it is facial expression recognition dataset developed by author jonathan Oheix.In this dataset it has two directories one is train and second one is validation.In test directory it has 7 directories they are happy,sad,fear,anger,surprise,disgust,neutral.
And in validation directory it has 7 directories they are happy,sad,fear,anger,surprise, disgust,neutral.In this data set the images are grayscale and it has dimensions of 48*48 pixels.This data set was created by gathering from google images search for emotions. Every image of each emotion type is returned by the function OS module in python.The number of images in two directories with each emotion will display below and samples images are also attached below .Our aim is to design CNN model with better accuracy and validation.

S.NO	Type of Emotion	No.of images in the Train dataset	No.of images in the Validation dataset
1.	Angry	3993	960
2.	Disgust	436	111
3.	Fear	4103	1018
4.	Happy	7164	1825
5.	Neutral	4982	1216
6.	Sad	4938	1139
7.	Surprise	3205	797

_SAMPLE IMAGES:



Algorithm Used:

During training for this data set, to minimize the losses of the Neural Network we will use an algorithm called Mini-Batch Gradient Descent.It is type Gradient Descent algorithm used for finding weights and co-efficient artificial neural networks by dividing the training dataset into the small batches.This algorithm will provide more efficiency whilst training data

Generate,Training data set:

We will collect a dataset from the keggal website and do code edit in the keggal kernel. We will add a dataset to data in the kaggle kernel.

1) **Import Libraries**:

Then we will import libraries like numpy,pandas,seaborn,os,matplotlib.pyplot.And we will also import deep learning libraries like keras.preprocessing .image .load image,image to array,ImageDataGenerator.we will also import different 7 layers for CNN to train the dataset.Those layers are Dense,Input,FlobalAveragePooling2D,falttern,Conv2D,Batch Normalization,Activation,Maxpooling2D.Next we will import Sequential and optimizers.

2) Display Images:

After importing necessary libraries then we will display images.we will standardize the each image and folder path from the taken dataset.
After that we will plot some images from the dataset for that we will take a certain expression ,its range,subplot(matrix),load image by os listdir and show image.

os.listdir=(floder_path+train+expression)

3) Making Training and Validation data:

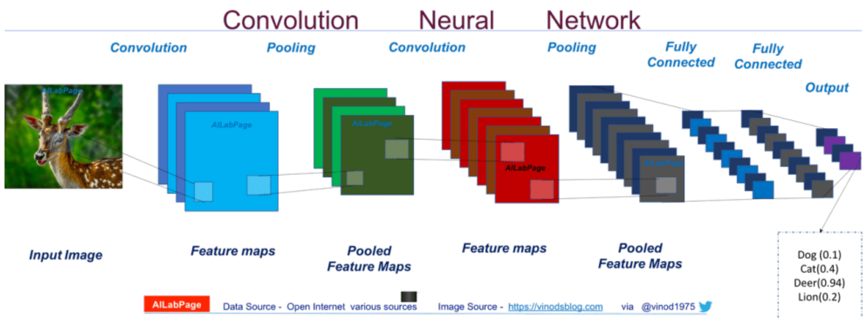
In this we will give Batch Size it will tell how many training examples schoul take our model for one iteration.we will give two variables one is datagen_train and second one is datagen_val.Both are defining ImageDataGenerator.After this we are defining two variables one is train_set and second is test-set.In train-set it will directory of datagen_train.In datagen_train it will contain folder path and train.The images in folder path will get trained and with specific size,colour,class_mode,shuffle.same in test_set also it will contain directory of datagen_val.In datagen-val it contains folder_path and train.Then the images which are in folder path will be trained with specific size,colour,class-mode,shuffle.Then it will found '7' classes in both datagen_train and datagen_val.

4) Model Building:

Here we will use Convolutional neural network to recognize different expressions.

Convolutional neural network:-

Here we will define the number of classes as 7 and outputs will be 7 expressions and the model we are defining in sequential. We are defining seven layers which are part of CNN.These all 7 layers will make an artificial neural network or deep neural network.In each and every layer we will define Conv2d filter and kernel size,padding,input_shape,Batch Normalization layer,Activation layer (we will take liner),Maxpooling2D(we should define size then it will extract important information from the point where we kept size) ,Dropout(to prevent our model to get over fitted).The input_shape should be defined in 1st layer no need to be define in every layer.we will define these layers in every CNN layer expect input_shape.Then we will crete flatten layer.The function of flatten layer is to collapse the input size to one dimension array which easily fed into the system.Then we will fully connected to the 1st layer and 2nd layer with the help of "Dense".Then we will define adam optimizer(learning rate=0.001) next we will define model compiler in that optimizer will fed into it with categorical cross entropy and we will define metrics with accuracy.Finally we will we wil print model summary.



Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 48, 48, 64)	640
batch_normalization (BatchNo	(None, 48, 48, 64)	256
activation (Activation)	(None, 48, 48, 64)	0
max_pooling2d (MaxPooling2D)	(None, 24, 24, 64)	0
dropout (Dropout)	(None, 24, 24, 64)	0
conv2d_1 (Conv2D)	(None, 24, 24, 128)	204928
batch_normalization_1 (Batch	(None, 24, 24, 128)	512
activation_1 (Activation)	(None, 24, 24, 128)	0
max_pooling2d_1 (MaxPooling	(None, 12, 12, 128)	0
dropout_1 (Dropout)	(None, 12, 12, 128)	0
conv2d_2 (Conv2D)	(None, 12, 12, 512)	590336
batch_normalization_2 (Batch	(None, 12, 12, 512)	2048
activation_2 (Activation)	(None, 12, 12, 512)	0
max_pooling2d_2 (MaxPooling2	(None, 6, 6, 512)	0

dropout_2 (Dropout)	(None, 6, 6, 512)	0
conv2d_3 (Conv2D)	(None, 6, 6, 512)	2359808
batch_normalization_3 (Batch	(None, 6, 6, 512)	2048
activation_3 (Activation)	(None, 6, 6, 512)	0
max_pooling2d_3 (MaxPooling2	(None, 3, 3, 512)	0
dropout_3 (Dropout)	(None, 3, 3, 512)	0
flatten (Flatten)	(None, 4608)	0
dense (Dense)	(None, 256)	1179904
batch_normalization_4 (Batch	(None, 256)	1024
activation_4 (Activation)	(None, 256)	0
dropout_4 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 512)	131584
batch_normalization_5 (Batch	(None, 512)	2048
activation_5 (Activation)	(None, 512)	0
dropout_5 (Dropout)	(None, 512)	0
dense_2 (Dense)	(None, 7)	3591

===

Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968

5) g the model with Training and Validation data:

For training and validation we should import the ModelCheckpoint,earlyStopping,Reduce ROnPlateau from keras.Now checkpoint (it will check every point in our model) it will save your model. We have saved the model in our output keggle in the form of model.h5,monitor will monitor my validation accuracy, mode will be maximum, verbose will be one.Now Earlystopping (if my model is not increasing accuracy then epoch is continuous then we will use earlystopping) it will monitor the validation accuracy and we will use some parameters like min_delta,patience.verbose.restore_best_weight(it will restore the best weights or best model).Then reduce_learningrate(if my model is not cope up with certain learning rate then it will reduce it) in this we will use parameters like factor,patience,verbose,min_delta.Then we will define call back list it will contain checkpoint,reduce learning rate,early stopping and we will define epochs.Then we will fit my model with training set and test set from training and validation data.Then we will consider generator as training set,steps_per_epoch,validation data will be as test_set,validation_steps and call back list.

STEPS_PER_EPOCH:-train_set.n//train_set.batch_size

Validation_steps:-test_set.n//test_set.batch_size

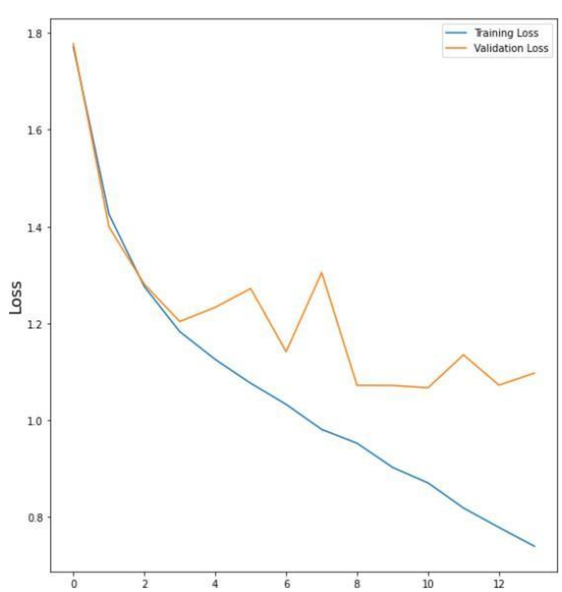
Epoch 1/48
225/225 [==============================] - 180s 799ms/step - loss: 1.7711 - accuracy: 0.3183 - val_loss: 1.7775 - val_accuracy: 0.3482 Epoch
2/48
225/225 [==============================] - 22s 99ms/step - loss: 1.4261 - accuracy: 0.4517 - val_loss: 1.3999 - val_accuracy: 0.4814 Epoch
3/48
225/225 [==============================] - 22s 97ms/step - loss: 1.2760 - accuracy: 0.5113 - val_loss: 1.2810 - val_accuracy: 0.5163 Epoch
4/48
225/225 [==============================] - 22s 100ms/step - loss: 1.1831 - accuracy: 0.5490 - val_loss: 1.2041 - val_accuracy: 0.5491 Epoch
5/48
225/225 [==============================] - 23s 100ms/step - loss: 1.1257 - accuracy: 0.5724 - val_loss: 1.2334 - val_accuracy: 0.5278 Epoch
6/48
225/225 [==============================] - 22s 98ms/step - loss: 1.0765 - accuracy: 0.5909 - val_loss: 1.2720 - val_accuracy: 0.5112 Epoch
7/48
225/225 [==============================] - 23s 101ms/step - loss: 1.0328 - accuracy: 0.6091 - val_loss: 1.1415 - val_accuracy: 0.5705 Epoch
8/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9813 - accuracy: 0.6282 - val_loss: 1.3052 - val_accuracy: 0.5300 Epoch
9/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9526 - accuracy: 0.6418 - val_loss: 1.0722 - val_accuracy: 0.6038

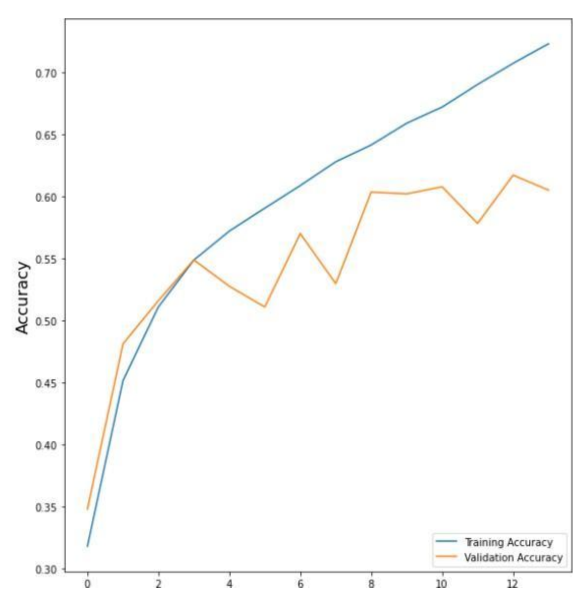
Epoch 10/48
225/225 [==============================] - 27s 120ms/step - loss: 0.9028 - accuracy: 0.6593 - val_loss: 1.0720 - val_accuracy: 0.6024 Epoch
11/48
225/225 [==============================] - 23s 104ms/step - loss: 0.8707 - accuracy: 0.6724 - val_loss: 1.0670 - val_accuracy: 0.6081 Epoch
12/48
225/225 [==============================] - 22s 98ms/step - loss: 0.8188 - accuracy: 0.6906 - val_loss: 1.1353 - val_accuracy: 0.5786 Epoch
13/48
225/225 [==============================] - 23s 104ms/step - loss: 0.7788 - accuracy: 0.7076 - val_loss: 1.0726 - val_accuracy: 0.6175 Epoch
14/48
225/225 [==============================] - ETA: 0s - loss: 0.7399 - accuracy: 0.7234Restoring model weights from the end of the best epoch.

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
225/225 [==============================] - 23s 102ms/step - loss: 0.7399 - accuracy: 0.7234 - val_loss: 1.0975 - val_accuracy: 0.6054 Epoch
00014: early stopping

6) g accuracy & loss:

We will plot the accuracy and loss.To plot then we will define plot style,figure,subplot,suptitle,ylabel,plot history loss,plot history validation loss,legend,plot show(to display).



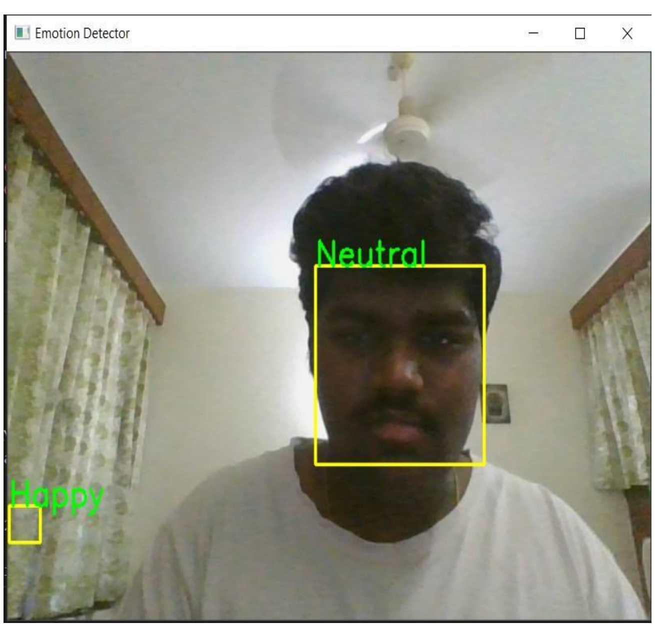


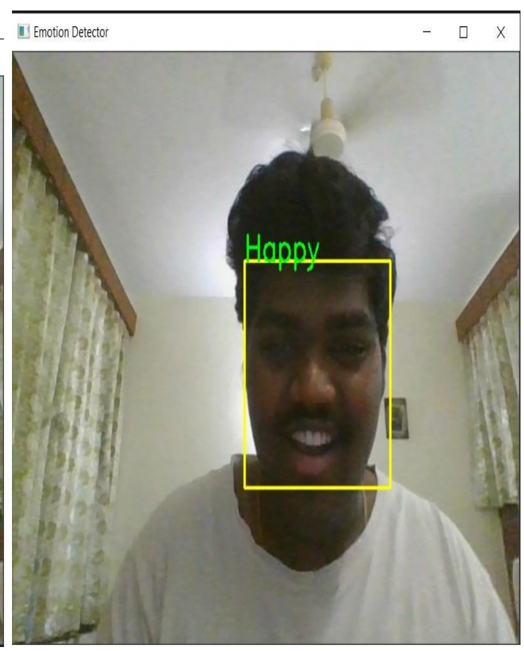
→ After all these we will get model.h5 in output .And we have to download a haarcascade xml file which is useful to deduct faces.we should keep this both hard cascade and model.h5 file in one folder and copy those folder url.

_OPEN-CV MODEL:

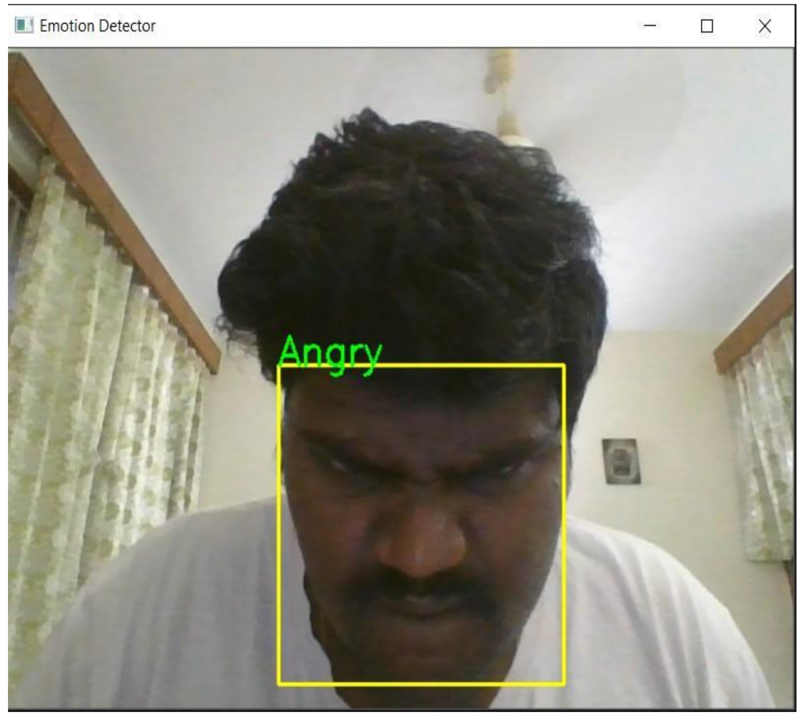
Import all wanted files from keras model class like load model(to load model and haar cascade files),preprocessor image to array,preprocessor load image from kerras,cv2,numpy.Then define two classifiers one for haar cascade and another one for model.h5 load them by folder url.After define emotion labels like angry,sad,happy,fear,surprise,disgust,neutral.After define cap for video capture from
machine or computer.Then define frame which can read by opencv.convert frame into gray scale and all the images will be gray. Faces variable contains all faces and returns in four parameters for each face it will draw a rectangular box around the face.The region inte**rest will be face only, image size will be 48*48 pixels because my model is
trained on 48*48 pixel size only.Then the image will convert into an array in the region of interest.By using classifiers it will predict the face emotion by emotion labels.If it
doesn't find any faces it will pop up NO faces.Then it will show Emotion detector frame and if we press "q" it will destroy all.

_RESULTS:






Emotion Detector
Surprise


Emotion Detector
Angry


Emotion Detector
Sad

CODE:

Importing Libraries

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries

from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import
Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activation,MaxPo
oling2D
from keras.models import Model,Sequential
from keras.optimizers import Adam,SGD,RMSprop
```

Displaying Images

```
picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"

expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(folder_path+"train/"+expression+"/"+
                  os.listdir(folder_path + "train/" + expression)[i], target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()
```

Making Training and Validation Data

```
batch_size = 128

datagen_train = ImageDataGenerator()
```

```
datagen_val = ImageDataGenerator()

train_set = datagen_train.flow_from_directory(folder_path+"train",
                                              target_size = (picture_size,picture_size),
                                              color_mode = "grayscale",
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=True)


test_set = datagen_val.flow_from_directory(folder_path+"validation",
                                              target_size = (picture_size,picture_size),
                                              color_mode = "grayscale",
                                              batch_size=batch_size,
                                              class_mode='categorical',
                                              shuffle=False)
```

Model Building

```
from keras.optimizers import Adam,SGD,RMSprop


no_of_classes = 7

model = Sequential()

#1st CNN layer
model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(0.25))

#2nd CNN layer
model.add(Conv2D(128,(5,5),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#3rd CNN layer
model.add(Conv2D(512,(3,3),padding = 'same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
```

```
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout (0.25))

#4th CNN layer
model.add(Conv2D(512,(3,3), padding='same'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

#Fully connected 1st layer
model.add(Dense(256))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))


# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

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



opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

Fitting The Model With Training And Validation Data

```
from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1, save_best_only=True,
mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                          min_delta=0,
```

```
                patience=3,
                verbose=1,
                restore_best_weights=True
                )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                    factor=0.2,
                    patience=3,
                    verbose=1,
                    min_delta=0.0001)

callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 48

model.compile(loss='categorical_crossentropy',
        optimizer = Adam(lr=0.001),
        metrics=['accuracy'])

history = model.fit_generator(generator=train_set,
                    steps_per_epoch=train_set.n//train_set.batch_size,
                    epochs=epochs,
                    validation_data = test_set,
                    validation_steps = test_set.n//test_set.batch_size,
                    callbacks=callbacks_list
                    )
```

Plotting Accuracy & Loss

```
plt.style.use('dark_background')

plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
```

```
plt.show()
```

OPEN_CV:-

```
from keras.models import load_model from time
import sleep
from keras.preprocessing.image import img_to_array from
keras.preprocessing import image
import cv2 import numpy
as np

face_classifier = cv2.CascadeClassifier(r'C:\Users\battu\urop\haarcascade_frontalface_default.xml ')
classifier =load_model(r'C:\Users\battu\urop\model.h5')

emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad', 'Surprise']

cap = cv2.VideoCapture(0)



while True:
    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)

    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
[redacted]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)



[redacted]
            roi = roi_gray.astype('float')/255.0
[redacted]
            roi = np.expand_dims(roi,axis=0)

[redacted]
            label=emotion_labels[prediction.argmax()]
[redacted]

cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
[redacted]
```

```
[redacted]
Faces',(30,80),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
  [redacted]

cap.release()
cv2.destroyAllWindows()
```

ACCURACY:

THE TRAINING ACCURACY:-0.71

THE TRAINING LOSS:-0.568

THE VALIDATION ACCURACY:-0.6054

THE VALIDATION LOSS:-1.098

CONCLUSION AND FUTURE HOPE:

Using the kaggle facial expression data set , a test accuracy is 60% is attained with this designed CNN model.The achieved results are satisfactory as the average accuracies and therefore, this CNN model is accurate.For an improvement in this model and its outcome , it is recommended to to change the parameters wherever useful in CNN model and removing unwanted parameters.Resize the learning rate and adapting with in the location may helpful to improve the accuracy and model.The number of epochs can be set to higher number to attaining the accuracy as output.But by increasing the number of epoch may lead to overfitting.

This similar CNN model can be used to different datasets to be trained and tested and check for its accuracy.

REFERENCES:

1) h ttps://www.researchgate.net/publication/342107269_Real-time_facial_expression_r ecognition_using_CNN

2) h ttps://www.youtube.com/watch?v=Bb4Wvl57LIk&t=857s

3)h ttps://ieeexplore.ieee.org/document/8866540

4)h ttps://web.stanford.edu/class/cs231a/prev_projects_2016/emotion-ai-real.pdf

5)h ttps://iopscience.iop.org/article/10.1088/1742-6596/1193/1/012004/pdf