# Source code for DIP Project

EMOTION RECOGNITION USING FACIAL EXPRESSIONS

# Team Members:

**KEVIN SIMEON JOSHUA 20MIS1088**

**PREM KUMAR 20MIS1179 DEEPAN 20MIS1160 HARISH S 20MIS1045**

## Code for loading the dataset:

import pandas as pd import cv2

import numpy as np

dataset\_path = 'fer2013/fer2013/fer2013.csv' image\_size=(48,48)

def load\_fer2013():

data = pd.read\_csv(dataset\_path) pixels = data['pixels'].tolist() width, height = 48, 48

faces = []

for pixel\_sequence in pixels:

face = [int(pixel) for pixel in pixel\_sequence.split(' ')] face = np.asarray(face).reshape(width, height)

face = cv2.resize(face.astype('uint8'),image\_size) faces.append(face.astype('float32'))

faces = np.asarray(faces)

faces = np.expand\_dims(faces, -1)

emotions = pd.get\_dummies(data['emotion']).as\_matrix() return faces, emotions

def preprocess\_input(x, v2=True): x = x.astype('float32')

x = x / 255.0 if v2:

x = x - 0.5

x = x \* 2.0 return x

## Code for training emotion classifier:

"""

Description: Train emotion classification model """

from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping from keras.callbacks import ReduceLROnPlateau

from keras.preprocessing.image import ImageDataGenerator from load\_and\_process import load\_fer2013

from load\_and\_process import preprocess\_input

from models.cnn import mini\_XCEPTION

from sklearn.model\_selection import train\_test\_split

# parameters batch\_size = 32

num\_epochs = 10000

input\_shape = (48, 48, 1)

validation\_split = .2

verbose = 1

num\_classes = 7

patience = 50 base\_path = 'models/'

# data generator

data\_generator = ImageDataGenerator(

featurewise\_center=False, featurewise\_std\_normalization=False, rotation\_range=10, width\_shift\_range=0.1, height\_shift\_range=0.1, zoom\_range=.1,

horizontal\_flip=True)

# model parameters/compilation

model = mini\_XCEPTION(input\_shape, num\_classes) model.compile(optimizer='adam', loss='categorical\_crossentropy',

metrics=['accuracy']) model.summary()

# callbacks

log\_file\_path = base\_path + '\_emotion\_training.log' csv\_logger = CSVLogger(log\_file\_path, append=False) early\_stop = EarlyStopping('val\_loss', patience=patience) reduce\_lr = ReduceLROnPlateau('val\_loss', factor=0.1,

patience=int(patience/4), verbose=1) trained\_models\_path = base\_path + '\_mini\_XCEPTION'

model\_names = trained\_models\_path + '.{epoch:02d}-{val\_acc:.2f}.hdf5' model\_checkpoint = ModelCheckpoint(model\_names, 'val\_loss', verbose=1,

save\_best\_only=True)

callbacks = [model\_checkpoint, csv\_logger, early\_stop, reduce\_lr]

# loading dataset

faces, emotions = load\_fer2013() faces = preprocess\_input(faces)

num\_samples, num\_classes = emotions.shape

xtrain, xtest,ytrain,ytest = train\_test\_split(faces, emotions,test\_size=0.2,shuffle=True) model.fit\_generator(data\_generator.flow(xtrain, ytrain,

batch\_size), steps\_per\_epoch=len(xtrain) / batch\_size,

epochs=num\_epochs, verbose=1, callbacks=callbacks, validation\_data=(xtest,ytest))

## Code for real time video:

from keras.preprocessing.image import img\_to\_array import imutils

import cv2

from keras.models import load\_model import numpy as np

# parameters for loading data and images

detection\_model\_path = 'haarcascade\_files/haarcascade\_frontalface\_default.xml' emotion\_model\_path = 'models/\_mini\_XCEPTION.102-0.66.hdf5'

# hyper-parameters for bounding boxes shape # loading models

face\_detection = cv2.CascadeClassifier(detection\_model\_path) emotion\_classifier = load\_model(emotion\_model\_path, compile=False) EMOTIONS = ["angry" ,"disgust","scared", "happy", "sad", "surprised", "neutral"]

#feelings\_faces = []

#for index, emotion in enumerate(EMOTIONS):

# feelings\_faces.append(cv2.imread('emojis/' + emotion + '.png', -1))

# starting video streaming cv2.namedWindow('your\_face') camera = cv2.VideoCapture(0) while True:

frame = camera.read()[1] #reading the frame

frame = imutils.resize(frame,width=300)

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY) faces =

face\_detection.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flag s=cv2.CASCADE\_SCALE\_IMAGE)

canvas = np.zeros((250, 300, 3), dtype="uint8") frameClone = frame.copy()

if len(faces) > 0:

faces = sorted(faces, reverse=True, key=lambda x: (x[2] - x[0]) \* (x[3] - x[1]))[0] (fX, fY, fW, fH) = faces

# Extract the ROI of the face from the grayscale image, resize it to a fixed 28x28 pixels, and then prepare

# the ROI for classification via the CNN roi = gray[fY:fY + fH, fX:fX + fW]

roi = cv2.resize(roi, (64, 64)) roi = roi.astype("float") / 255.0 roi = img\_to\_array(roi)

roi = np.expand\_dims(roi, axis=0)

preds = emotion\_classifier.predict(roi)[0] emotion\_probability = np.max(preds) label = EMOTIONS[preds.argmax()]

else: continue

for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)): # construct the label text

text = "{}: {:.2f}%".format(emotion, prob \* 100)

# draw the label + probability bar on the canvas # emoji\_face = feelings\_faces[np.argmax(preds)]

w = int(prob \* 300) cv2.rectangle(canvas, (7, (i \* 35) + 5),

(w, (i \* 35) + 35), (0, 0, 255), -1)

cv2.putText(canvas, text, (10, (i \* 35) + 23),

cv2.FONT\_HERSHEY\_SIMPLEX, 0.45,

(255, 255, 255), 2)

cv2.putText(frameClone, label, (fX, fY - 10), cv2.FONT\_HERSHEY\_SIMPLEX, 0.45, (0, 0, 255), 2)

cv2.rectangle(frameClone, (fX, fY), (fX + fW, fY + fH), (0, 0, 255), 2)

# for c in range(0, 3):

# frame[200:320, 10:130, c] = emoji\_face[:, :, c] \* \ # (emoji\_face[:, :, 3] / 255.0) + frame[200:320,

# 10:130, c] \* (1.0 - emoji\_face[:, :, 3] / 255.0)

cv2.imshow('your\_face', frameClone) cv2.imshow("Probabilities", canvas) if cv2.waitKey(1) & 0xFF == ord('q'):

break

camera.release() cv2.destroyAllWindows()