TRAINING

# import required packages

import cv2

from keras.models import Sequential

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

from tensorflow.keras.optimizers import Adam

from keras.preprocessing.image import ImageDataGenerator

# Initialize image data generator with rescaling

train\_data\_gen = ImageDataGenerator(rescale=1./255)

validation\_data\_gen = ImageDataGenerator(rescale=1./255)

# Preprocess all test images

train\_generator = train\_data\_gen.flow\_from\_directory(

        'data/train',

        target\_size=(48, 48),

        batch\_size=64,

        color\_mode="grayscale",

        class\_mode='categorical')

# Preprocess all train images

validation\_generator = validation\_data\_gen.flow\_from\_directory(

        'data/test',

        target\_size=(48, 48),

        batch\_size=64,

        color\_mode="grayscale",

        class\_mode='categorical')

# create model structure

emotion\_model = Sequential()

emotion\_model.add(Conv2D(32, kernel\_size=(3, 3), activation='relu', input\_shape=(48, 48, 1)))

emotion\_model.add(Conv2D(64, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Conv2D(128, kernel\_size=(3, 3), activation='relu'))

emotion\_model.add(MaxPooling2D(pool\_size=(2, 2)))

emotion\_model.add(Dropout(0.25))

emotion\_model.add(Flatten())

emotion\_model.add(Dense(1024, activation='relu'))

emotion\_model.add(Dropout(0.5))

emotion\_model.add(Dense(7, activation='softmax'))

cv2.ocl.setUseOpenCL(False)

emotion\_model.compile(loss='categorical\_crossentropy', optimizer=Adam(lr=0.0001, decay=1e-6), metrics=['accuracy'])

# Train the neural network/model

emotion\_model\_info = emotion\_model.fit\_generator(

        train\_generator,

        steps\_per\_epoch=28709 // 64,

        epochs=50,

        validation\_data=validation\_generator,

        validation\_steps=7178 // 64)

# save model structure in jason file

model\_json = emotion\_model.to\_json()

with open("emotion\_model.json", "w") as json\_file:

    json\_file.write(model\_json)

# save trained model weight in .h5 file

emotion\_model.save\_weights('emotion\_model.h5')

TESTING

import cv2

import numpy as np

from keras.models import model\_from\_json

emotion\_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3: "Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}

# load json and create model

json\_file = open('model/emotion\_model.json', 'r')

loaded\_model\_json = json\_file.read()

json\_file.close()

emotion\_model = model\_from\_json(loaded\_model\_json)

# load weights into new model

emotion\_model.load\_weights("model/emotion\_model.h5")

print("Loaded model from disk")

# start the webcam feed

#cap = cv2.VideoCapture(0)

# pass here your video path

# you may download one from here : https://www.pexels.com/video/three-girls-laughing-5273028/

cap = cv2.VideoCapture("C:\\JustDoIt\\ML\\Sample\_videos\\emotion\_sample6.mp4")

while True:

    # Find haar cascade to draw bounding box around face

    ret, frame = cap.read()

    frame = cv2.resize(frame, (1280, 720))

    if not ret:

        break

    face\_detector = cv2.CascadeClassifier('haarcascades/haarcascade\_frontalface\_default.xml')

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

    # detect faces available on camera

    num\_faces = face\_detector.detectMultiScale(gray\_frame, scaleFactor=1.3, minNeighbors=5)

    # take each face available on the camera and Preprocess it

    for (x, y, w, h) in num\_faces:

        cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (0, 255, 0), 4)

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

        cropped\_img = np.expand\_dims(np.expand\_dims(cv2.resize(roi\_gray\_frame, (48, 48)), -1), 0)

        # predict the emotions

        emotion\_prediction = emotion\_model.predict(cropped\_img)

        maxindex = int(np.argmax(emotion\_prediction))

        cv2.putText(frame, emotion\_dict[maxindex], (x+5, y-20), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (255, 0, 0), 2, cv2.LINE\_AA)

    cv2.imshow('Emotion Detection', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):

        break

cap.release()

cv2.destroyAllWindows()