HUMAN EMOTION DETECTION USING NUERAL NETWORK

A PROJECT BY

GAURAV KUMAR SINGH

Abstract

Human emotion recognition plays an important role in the relational relationship. The automatic recognition of emotions has been a functioning examination theme from early times. Hence, there are a few advances made in this field. Emotions are reflected from discourse, hand and signals of the body and through outward appearances. Henceforth extracting and understanding of emotion has a high significance of the collaboration among human and machine correspondence. This project depicts the advances made in this field and the different methodologies utilized for recognition of emotions. The main objective of the paper is to propose a real-time implementation of emotion recognition system.

Contents

1. Introduction……………………………………………………………………….....01

2. Problem Statement…………………………………………………………………..01

3. Aim of the Project………..……………………………………………………..…01

4. Literature Survey………..……………………………………………………………02
   4.1. Dataset Used…………………..…………………………………………………….02
   4.2. Attribute Details…………...…………………………………… …………….02
   4.3. Existing System……………………………………………………………………04

5. Design and Implementation….………………………………………….……………04
   5.1. Proposed System……… ……..…………………………………………………04
   5.2. Training Phase……...……...…... ……………………………………………….06
   5.3. Classification and Detection.………………………………………………….10
      5.3.1. Single Face Detection……………………………………………………10
      5.3.2. Multiple Face Detection…………………………………………………..12

6. Code… ……………………………………………………………………...…..13

7. Conclusion……………………………………………………………………………21

8. References……………………………………………………………………………21

1. Introduction

One of the current top applications of artificial intelligence using neural networks is the recognition of faces in photos and videos. Most techniques process visual data and search for general patterns present in human faces. Face recognition can be used for surveillance purposes by law enforcers as well as in crowd management. Other present-day applications involve automatic blurring of faces on Google Street view footage and automatic recognition of Facebook friends in photos. An even more advanced development in this field is emotion recognition. In addition to only identifying faces, the computer uses the arrangement and shape of e.g. eyebrows and lips to determine the facial expression and hence the emotion of a person. One possible application for this lies in the area of surveillance and behavioural analysis by law enforcement. The success of service robotics decisively depends on a smooth robot to user interaction. Thus, a robot should be able to extract information just from the face of its user, e.g. identify the emotional state or deduce gender. Interpreting correctly any of these elements using machine learning (ML) techniques has proven to be complicated due the high variability of the samples within each task. This leads to models with millions of parameters trained under thousands of samples. Furthermore, the human accuracy for classifying an image of a face in one of 7 different emotions is 65% ± 5%. One can observe the difficulty of this task by trying to manually classify the FER-2013 dataset images in Figure 1 within the following classes {“angry”, “disgust”, “fear”, “happy”, “sad”, “surprise”, “neutral”}.

2. Problem Statement

Many actions can be defined via observing the emotion the person is expressing. One of the current top applications of artificial intelligence using neural networks is the recognition of faces in photos and videos. Most techniques process visual data and search for general patterns present in human faces.

3. Aim of the Project

Face recognition can be used for surveillance purposes by law enforcers as well as in crowd management. Other present-day applications involve automatic blurring of faces on Google Street view footage and automatic recognition of Facebook friends in photos. An even more advanced development in this field is emotion recognition. In addition to only identifying faces, the computer uses the arrangement and shape of e.g. eyebrows and lips to determine the facial expression and hence the emotion of a person. One possible application for this lies in the area of surveillance and behavioural analysis by law enforcement.

4. Literature Survey

4.1. Dataset Used

The dataset has been taken from Kaggle on the URL: https://www.kaggle.com/deadskull7/fer2013. The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less cantered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples. The public test set used for the leader board consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples. This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project. They have graciously provided the workshop organizers with a preliminary version of their dataset to use for this contest.

Fig. 1: Samples of the FER-2013 emotion dataset.

4.2. Attribute Details

The dataset has total 3 attributes out of which 2 are independent variables and one is the dependent variable i.e. target variable which classifies the image into an emotion. There are 2 independent variables:

1. **Pixels:** The 48x48 grey-scale image is made in an array of 48x48 entries where each pixel is represented by its intensity.

2. **Usage:** Categorized into 3 parts Training, Public testing and private testing

3. **emotion:** Containing no. 0 to 6 for the emotions: 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral

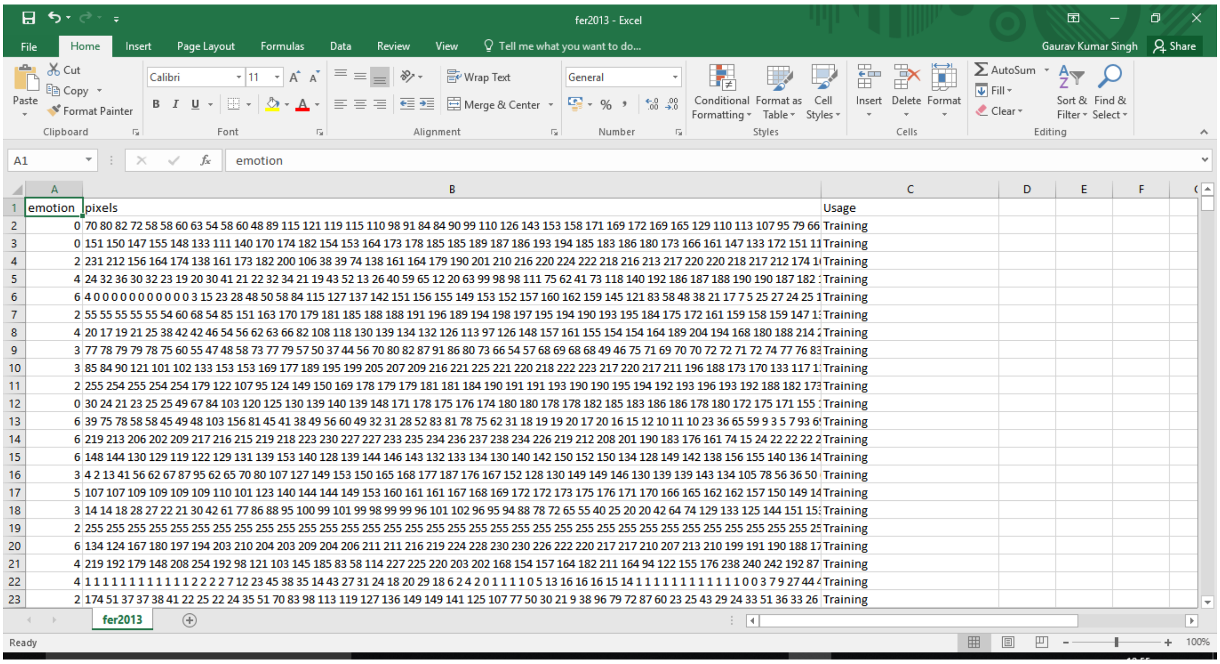


Fig. 2: Screenshot of the fer2-13 dataset.

4.3. Existing Systems

Comparative Study				
Title	**Technique**	**Database**	**Performance (%)**	**Remarks**
Statistical Moments based Facial expression Analysis	**Feature Extraction:** Zernike moments **Classification:** Naive Bayesian classifier	JAFFE (Japanese Female Facial expression) database 60 images used for experiment.	Average accuracy for six emotions is 81.66% in time less than 2 seconds.	Emotion accuracy graph shows highest recognition rate of happiness and lowest recognition rate of sadness.
Facial expression recognition with Auto-Illumination correction	Expressions on the face are determined with Action Units (AU's)	Single and Multiple face image	60% recognition rate for multiple face image	Illumination on image plays vital role.
Identification-driven Emotion recognition system for a Social Robot	Hybrid approach used for personalized emotion recognition,	MUG facial expression database used. More than 50 people frontal face database used aged between 20-25 years.	82% performance achieved with KNN Classifiers.	3D model facial image used.KNN classifier gives good performance for emotion recognition.
The application study of learner's face detection and location in the teaching network system based on emotion	SVM(Support Vector Machine) classifier based Adaboost algorithm used	PIE face image database used	Detection and Correction rate 95% or more.	Presents application of face emotion recognition with of E-learning system.

5. Design and Implementation

5.1. Proposed System

A Convolutional neural networks with pooling layers is used to detect the several emotions. The model relies on the idea of eliminating completely the fully connected layers. The architecture combines the deletion of the fully connected layer and the inclusion of the combined depth-wise separable convolutions and residual modules. I have also implement multiple face emotion recognition which can detect multiple faces on a single frame. Major steps involved are:

- First, the use of small CNNs alleviate us from slow performances in hardware-constrained systems such robot platforms.
- And second, the reduction of parameters provides a better generalization under an Occam's razor framework.
- The model relies on the idea of eliminating completely the fully connected layers.
- The architecture combines the deletion of the fully connected layer and the inclusion of the combined depth-wise separable convolutions and residual modules.
- Both architectures were trained with the ADAM optimizer use Batch Normalization for better accuracy.
- Convolutional 2D network has been used to train the models.

- Take the real-time video or video file as an input
- Put it under processing
- Output is displayed.

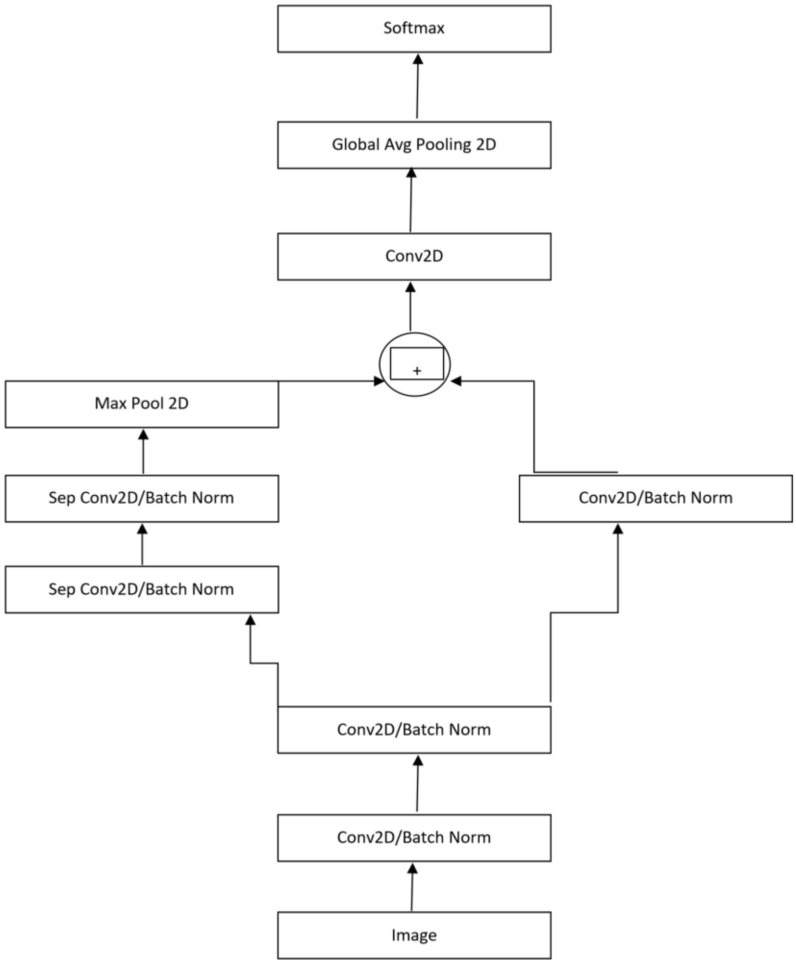
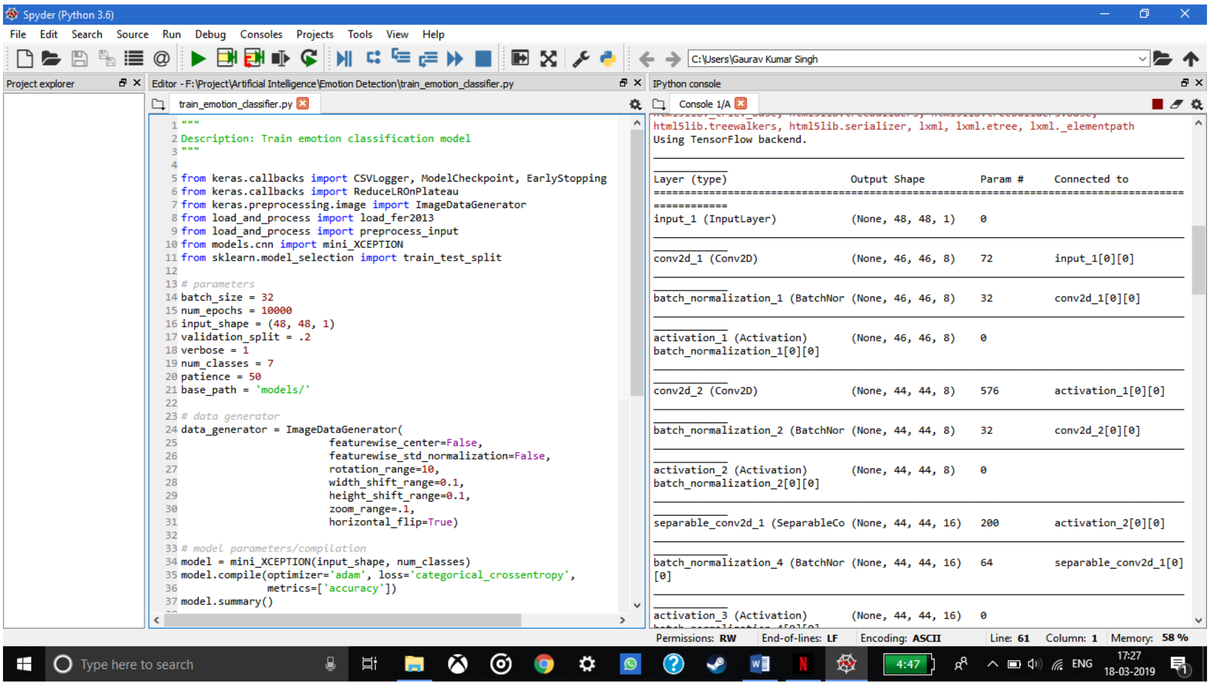
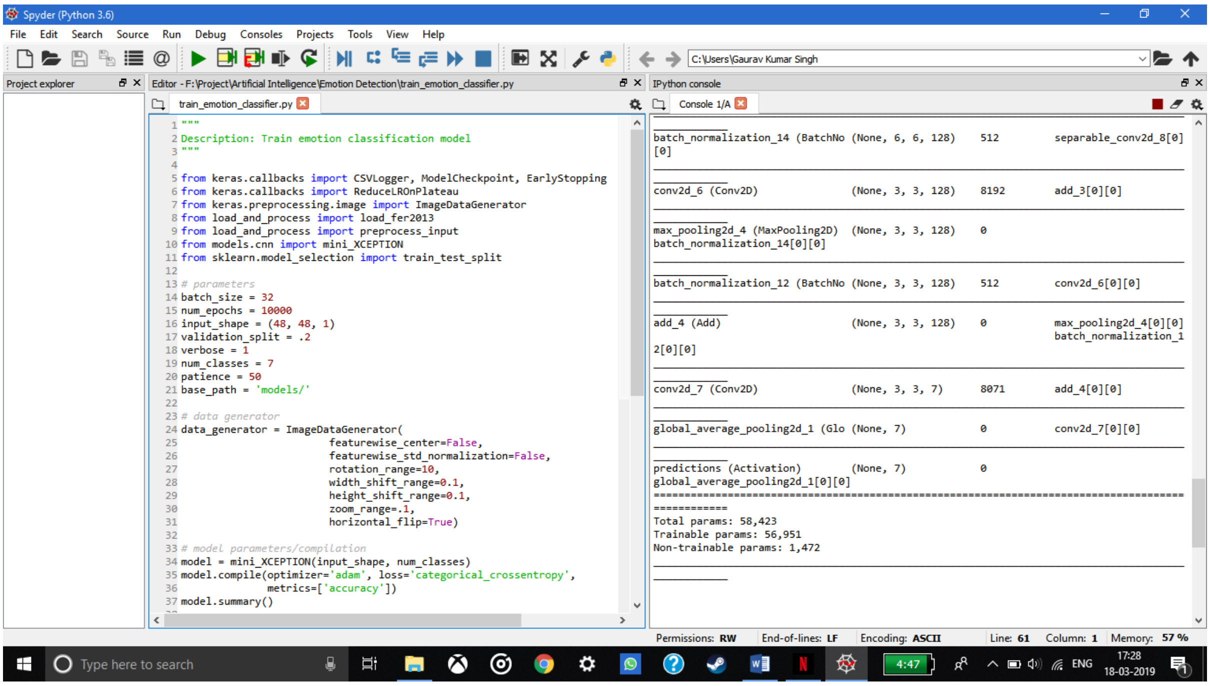


Fig. 3: Architecture of the proposed system.

5.2. Training Phase

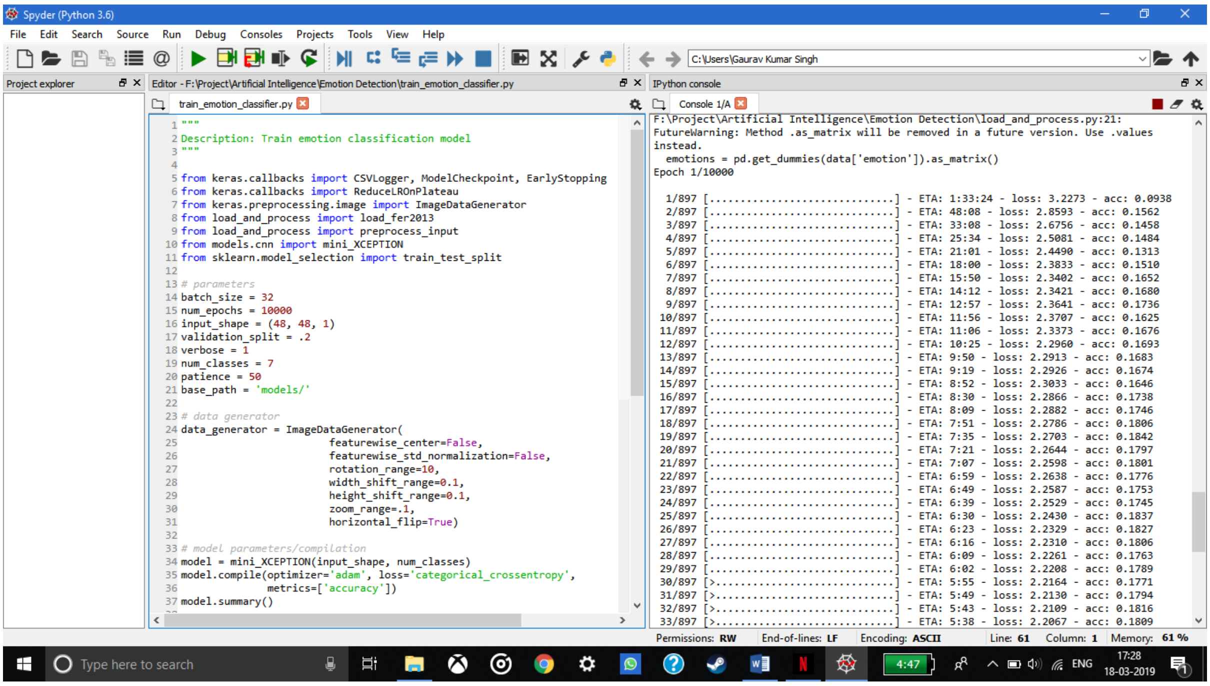
The model was trained with the fer2013 dataset and the model achieves an accuracy of 66%. The training process ran for 106 epochs in which it garnered the accuracy. A new model was made only when the accuracy increased and error decreased. The following are the screenshots of the training process:

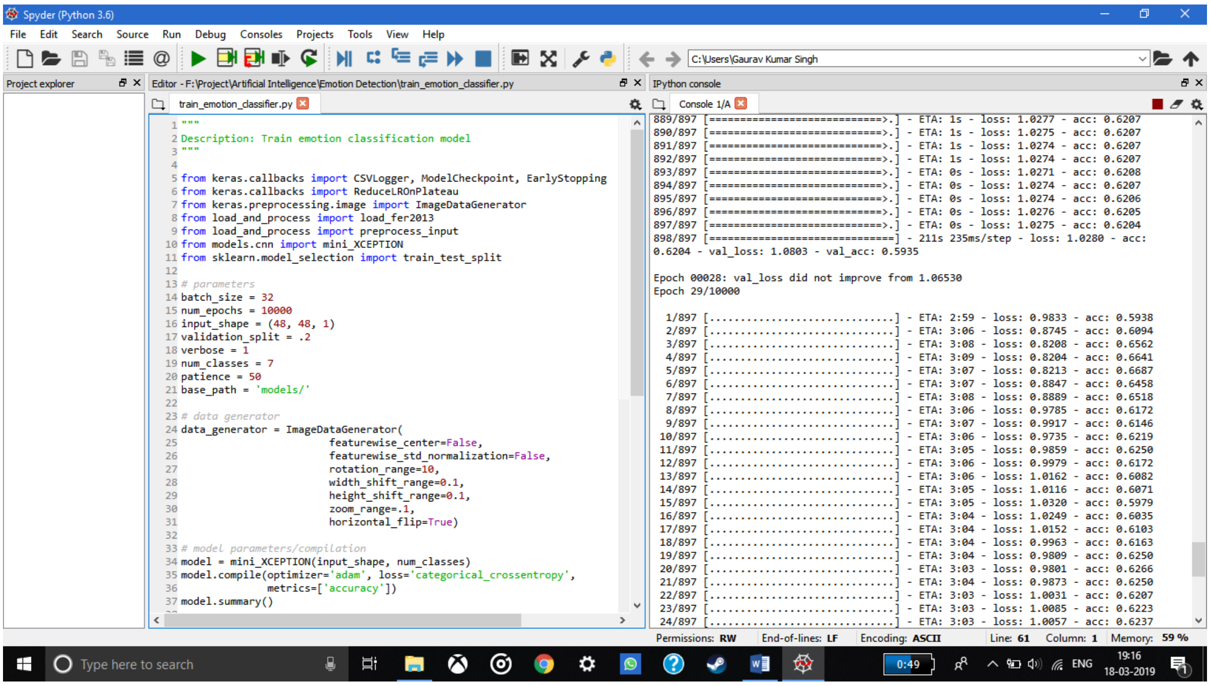




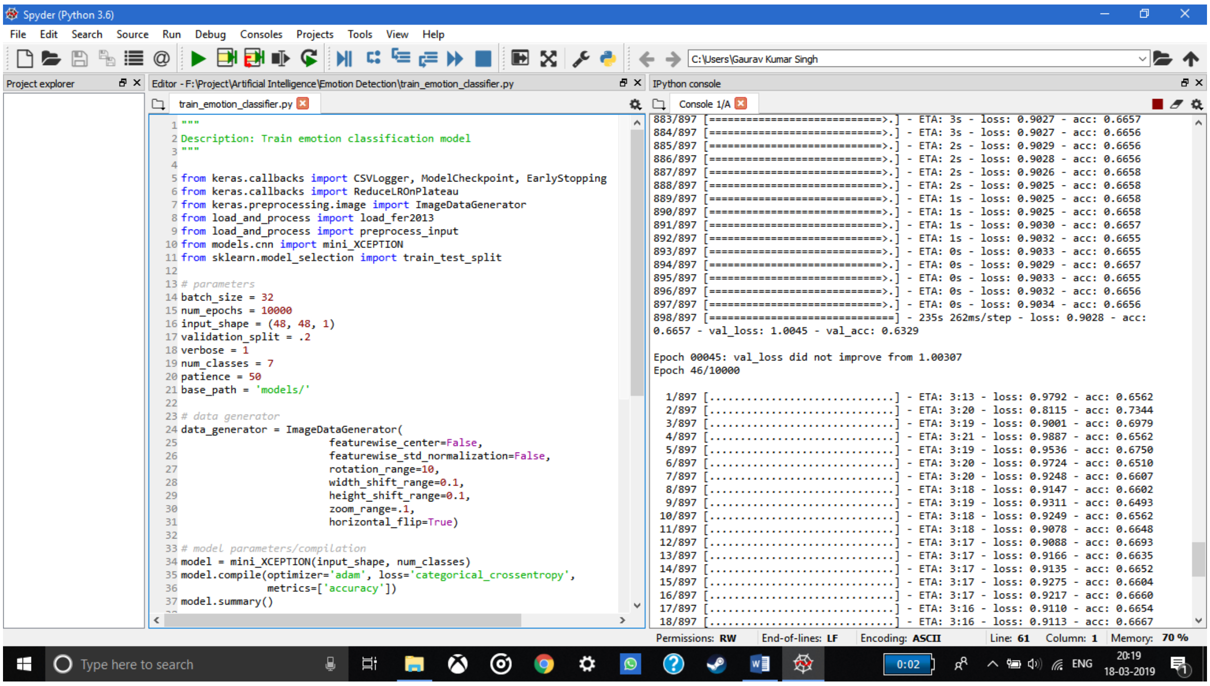
The above screenshots describe the parameters of the training phase.

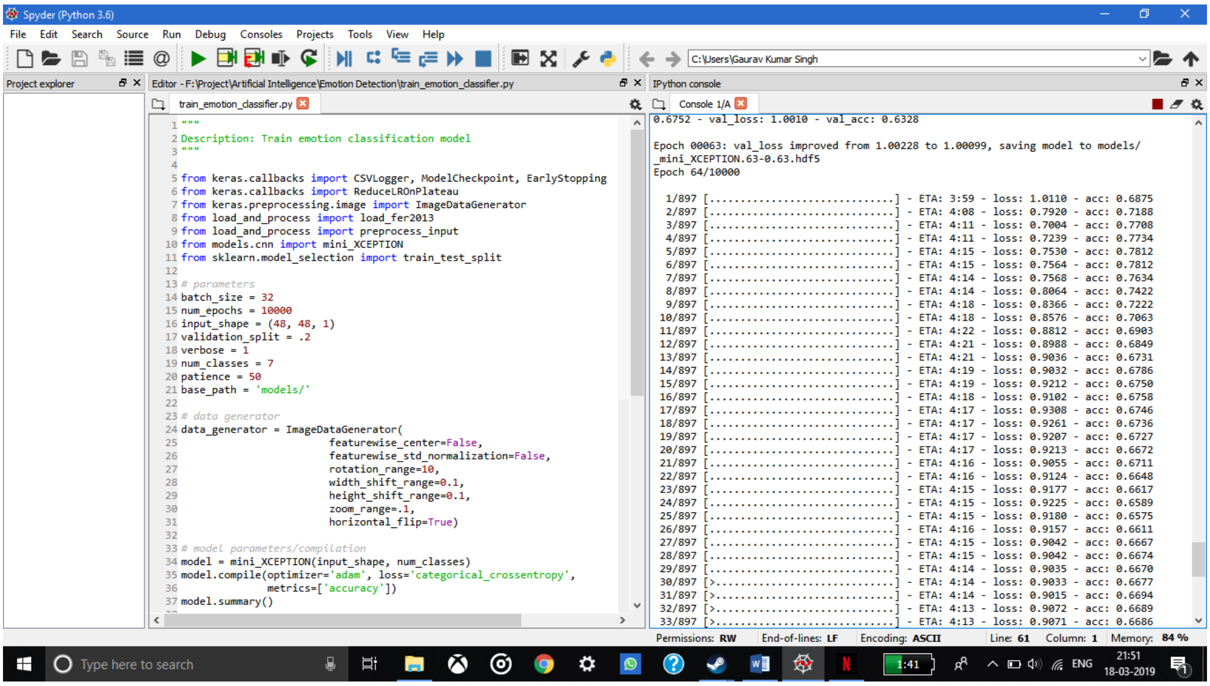
The following are the actual training process starting with epoch 1:





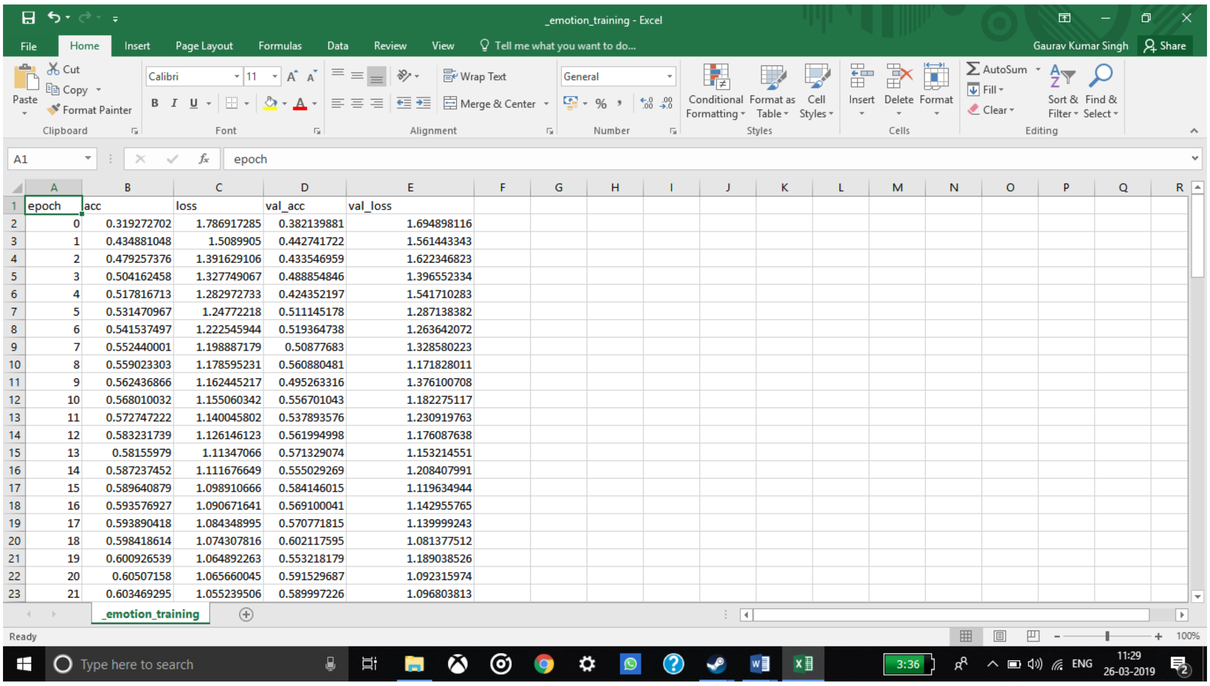
In the above figure the val_loss doesn't improve hence no model is made.





Here in the above figure the model is made as the val_loss improves.

The training log file maintained during the training phase:



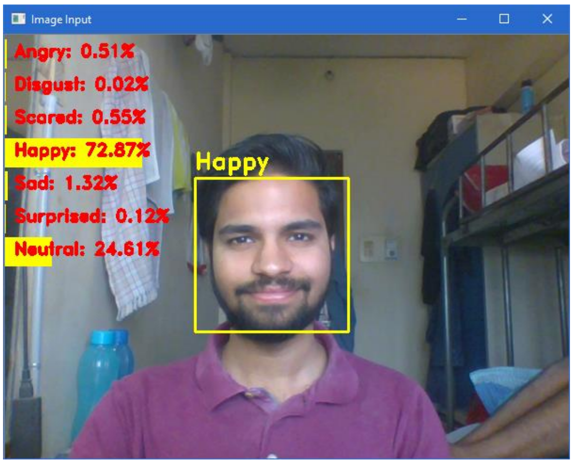
epoch	acc	loss	val_acc	val_loss
0	0.319272702	1.786917285	0.382139881	1.694898116
1	0.434881048	1.5089905	0.442741722	1.561443343
2	0.479257376	1.391629106	0.433546959	1.622346823
3	0.504162458	1.327749067	0.488854846	1.396552334
4	0.517816713	1.282972733	0.424352197	1.541710283
5	0.531470967	1.24772218	0.511145178	1.287138382
6	0.541537497	1.222545944	0.519364738	1.263642072
7	0.552440001	1.198887179	0.50877683	1.328580223
8	0.559023303	1.178595231	0.560880481	1.171828011
9	0.562436866	1.162445217	0.495263316	1.376100708
10	0.568010032	1.155060342	0.556701043	1.182275117
11	0.572747222	1.140045802	0.537893576	1.230919763
12	0.583231739	1.126146123	0.561994998	1.176087638
13	0.58155979	1.11347066	0.571329074	1.153214551
14	0.587237452	1.111676649	0.555029269	1.208407991
15	0.589640879	1.098910666	0.584146015	1.119634944
16	0.593576927	1.090671641	0.569100041	1.142955765
17	0.593890418	1.084348995	0.570771815	1.139999243
18	0.598418614	1.074307816	0.602117595	1.081377512
19	0.600926539	1.064892263	0.553218179	1.189038526
20	0.60507158	1.065660045	0.591529687	1.092315974
21	0.603469295	1.055239506	0.589997226	1.096803813

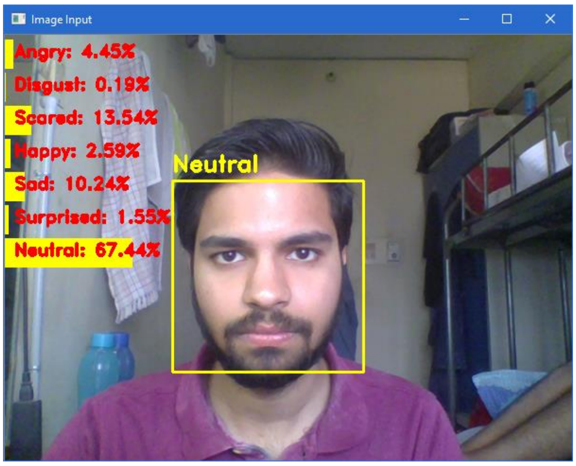
5.2. Classification and Detection

After the model is created we take the input and predict the emotion using the model. There are two types of programs. The first one detects a single face and also draws the probabilities of different emotion in the same frame as a function of the probabilities in a bar graph form. The second one detects and classifies multiple face simultaneously without displaying probabilities.

5.2.1. Single Face Detection

The following are the screenshots of the single face detection:





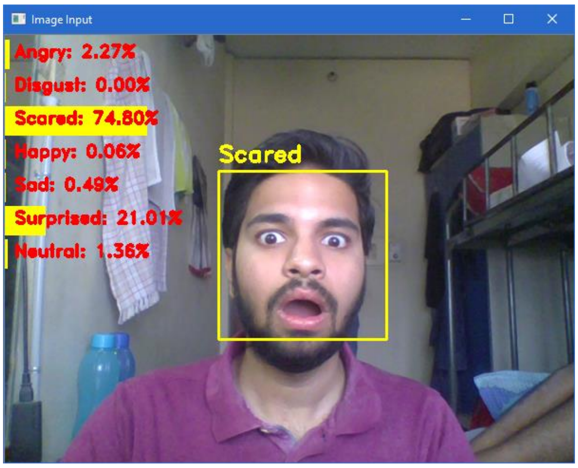


Image Input
Angry: 2.27%
Disgust: 0.00%
Scared: 74.80%
Happy: 0.06%
Sad: 0.49%
Surprised: 21.01%
Neutral: 1.36%
Scared

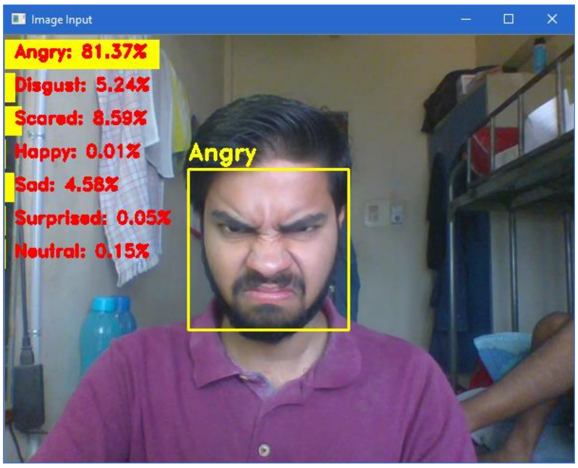


Image Input
Angry: 81.37%
Disgust: 5.24%
Scared: 8.59%
Happy: 0.01%
Sad: 4.58%
Surprised: 0.05%
Neutral: 0.15%
Angry

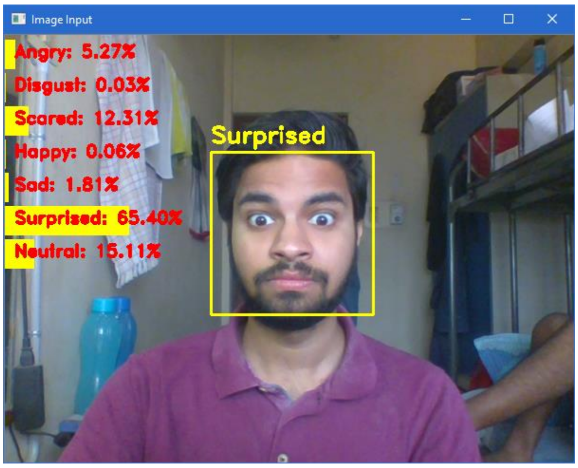


Image Input
Angry: 5.27%
Disgust: 0.03%
Scared: 12.31%
Happy: 0.06%
Sad: 1.81%
Surprised: 65.40%
Neutral: 15.11%
Surprised

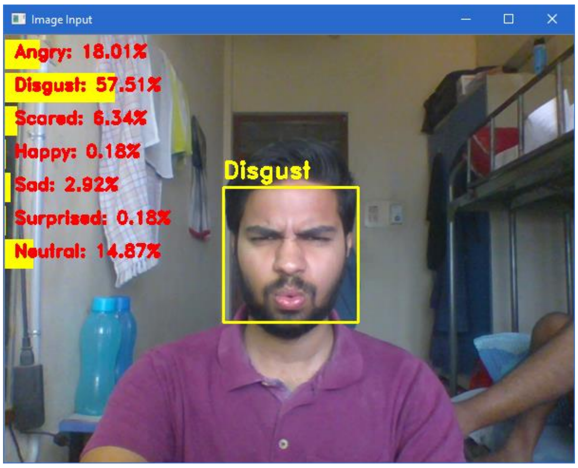


Image Input
Angry: 18.01%
Disgust: 57.51%
Scared: 6.34%
Happy: 0.18%
Sad: 2.92%
Surprised: 0.18%
Neutral: 14.87%
Disgust

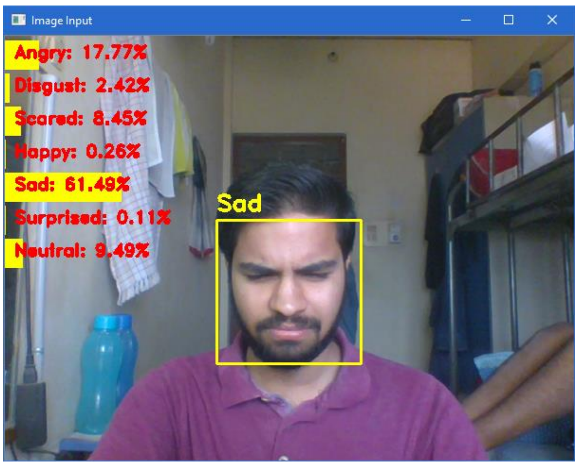


Image Input
Angry: 17.77%
Disgust: 2.42%
Scared: 8.45%
Happy: 0.26%
Sad: 61.49%
Surprised: 0.11%
Neutral: 9.49%
Sad

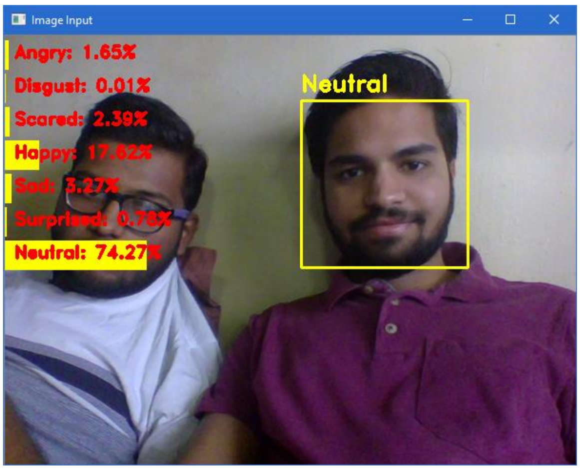
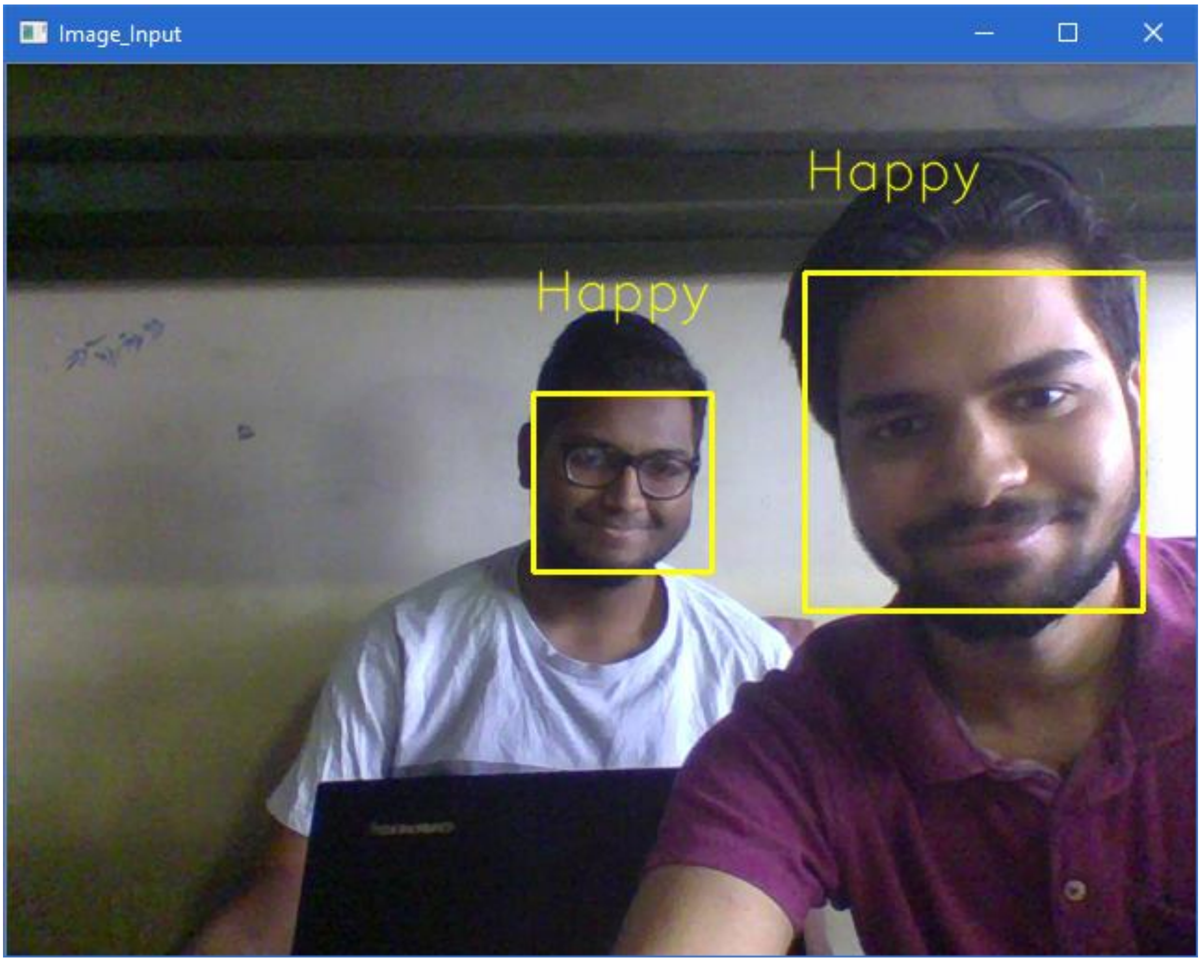
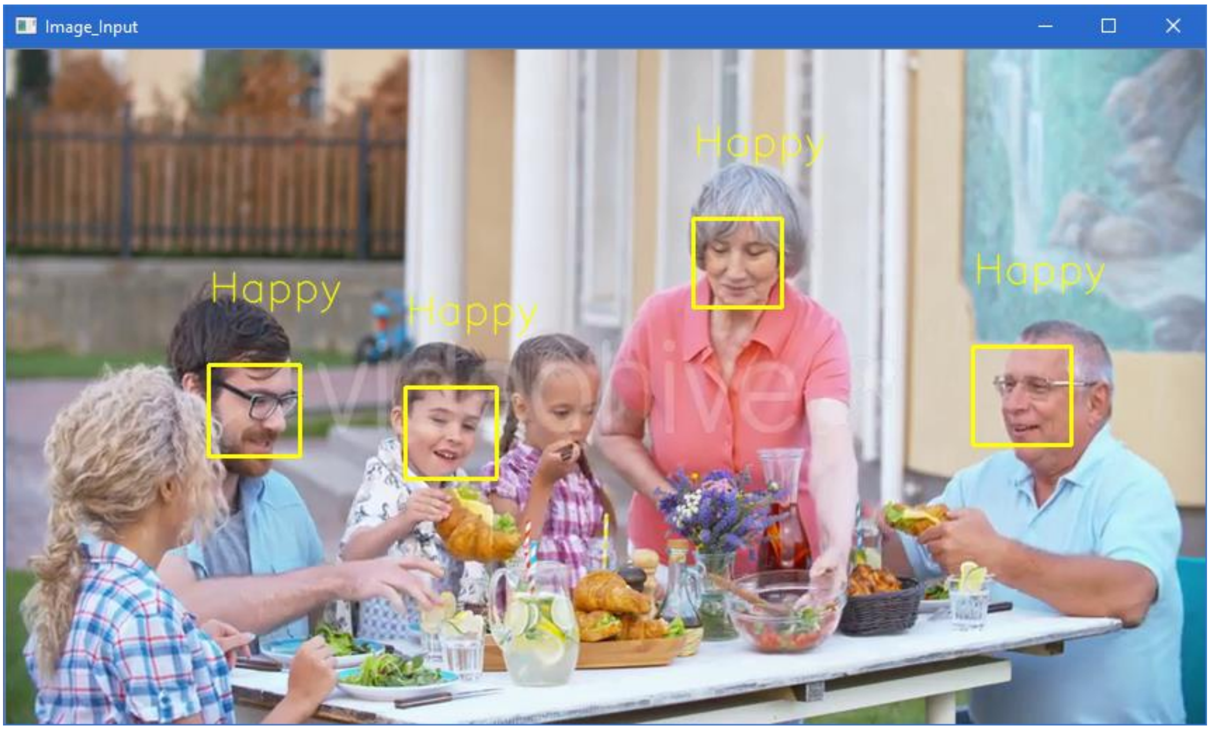


Image Input
Angry: 1.65%
Disgust: 0.01%
Scared: 2.39%
Happy: 17.62%
Sad: 3.27%
Surprised: 0.78%
Neutral: 74.27%
Neutral

5.2.2. Multiple Face Detection

The following are the screenshots of the multiple face detection:





6. Code

load_and_process:

```
import pandas as pd

import cv2

import numpy as np



dataset_path = 'Dataset/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
```

train_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))
```

Single:

```
import cv2

from keras.preprocessing.image import img_to_array

import imutils

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"]




# starting video streaming

camera = cv2.VideoCapture(0)

while True:

   frame = camera.read()[1]

   #reading the frame

   frame = imutils.resize(frame,width=600)

   gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

   faces =
face_detection.detectMultiScale(gray,scaleFactor=1.1,minNeighbors=5,minSize=(30,30),flags=cv2.CASCADE_S
CALE_IMAGE)

   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()]


    for (i, (emotion, prob)) in enumerate(zip(EMOTIONS, preds)):
            # construct the label text
            text = "{}: {:.2f}%".format(emotion, prob * 100)

            w = int(prob * 200)
            cv2.rectangle(frame, (0, (i * 35) + 5),
            (w, (i * 35) + 35), (0, 255, 255), -1)
            cv2.putText(frame, text, (10, (i * 35) + 23),
            cv2.FONT_HERSHEY_DUPLEX, 0.6,
            (0, 0, 255), 2)
            cv2.putText(frame, label, (fX, fY - 10),
            cv2.FONT_HERSHEY_DUPLEX, 0.8, (0, 255, 255), 2)
            cv2.rectangle(frame, (fX, fY), (fX + fW, fY + fH),
                        (0, 255, 255), 2)

    cv2.imshow('Image Input', frame)
```

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

camera.release()
cv2.destroyAllWindows()
```

Multiple:

```
import cv2
import numpy as np
from keras.models import load_model
from statistics import mode
#from utils.datasets import get_labels
#from utils.inference import detect_faces
from utils.inference import draw_text
from utils.inference import draw_bounding_box
from utils.inference import apply_offsets
#from utils.inference import load_detection_model
from utils.preprocessor import preprocess_input



USE_WEBCAM = True
# If false, loads video file source

# parameters for loading data and images
emotion_model_path = 'models/_mini_XCEPTION.102-0.66.hdf5'
#emotion_model_path = 'models/_mini_XCEPTION.63-0.63.hdf5'
emotion_labels=["Angry" ,"Disgust","Scared", "Happy", "Sad", "Surprised", "Neutral"]

# hyper-parameters for bounding boxes shape
frame_window = 10
emotion_offsets = (20, 40)
```

```
# loading models
face_cascade = cv2.CascadeClassifier('./haarcascade_files/haarcascade_frontalface_default.xml')
emotion_classifier = load_model(emotion_model_path,compile=False)

# getting input model shapes for inference
emotion_target_size = emotion_classifier.input_shape[1:3]

# starting lists for calculating modes
emotion_window = []

# starting video streaming

cv2.namedWindow('Image_Input')
video_capture = cv2.VideoCapture(0)

# Select video or webcam feed
cap = None
if (USE_WEBCAM == True):
    cap = cv2.VideoCapture(0) # Webcam source
else:
    cap = cv2.VideoCapture('dinner.mp4') # Video file source

while cap.isOpened(): # True:
    ret, bgr_image = cap.read()

    #bgr_image = video_capture.read()[1]

    gray_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2GRAY)
    rgb_image = cv2.cvtColor(bgr_image, cv2.COLOR_BGR2RGB)

    faces = face_cascade.detectMultiScale(gray_image, scaleFactor=1.1, minNeighbors=5,
```

```
            minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)

for face_coordinates in faces:

  x1, x2, y1, y2 = apply_offsets(face_coordinates, emotion_offsets)
  gray_face = gray_image[y1:y2, x1:x2]
  try:
    gray_face = cv2.resize(gray_face, (emotion_target_size))
  except:
    continue

  gray_face = preprocess_input(gray_face, True)
  gray_face = np.expand_dims(gray_face, 0)
  gray_face = np.expand_dims(gray_face, -1)
  emotion_prediction = emotion_classifier.predict(gray_face)
  emotion_probability = np.max(emotion_prediction)
  emotion_label_arg = np.argmax(emotion_prediction)
  emotion_text = emotion_labels[emotion_label_arg]
  emotion_window.append(emotion_text)

  if len(emotion_window) > frame_window:
    emotion_window.pop(0)
  try:
    emotion_mode = mode(emotion_window)
  except:
    continue
  color=np.asarray((255,255,0))
  color = color.astype(int)
  color = color.tolist()

  draw_bounding_box(face_coordinates, rgb_image, color)
  draw_text(face_coordinates, rgb_image, emotion_mode,
```

```
        color, 0, -45, 1, 1)


    bgr_image = cv2.cvtColor(rgb_image, cv2.COLOR_RGB2BGR)
    cv2.imshow('Image_Input', bgr_image)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


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

9. Conclusion

A real time CNN model is built which can identify the emotion of the user standing in front of the came air can identify them via image or video input. The proposed architecture has been systematically built in order to reduce the amount of parameters. We began by eliminating completely the fully connected layers and by reducing the amount of parameters in the remaining convolutional layers via depth-wise separable convolutions.

10. References

[1] Real-time Convolutional Neural Networks for Emotion and Gender Classification by Octavio Arriaga, Paul G. Ploger and Matias Valdenegro

[2] Human Emotion Recognition System by Dilbag Singh

[3] https://en.wikipedia.org/wiki/Emotion_recognition

[4] https://docs.opencv.org/3.4.3/d7/d8b/tutorial_py_face_detection.html

[5] https://www.geeksforgeeks.org/image-classifier-using-cnn/