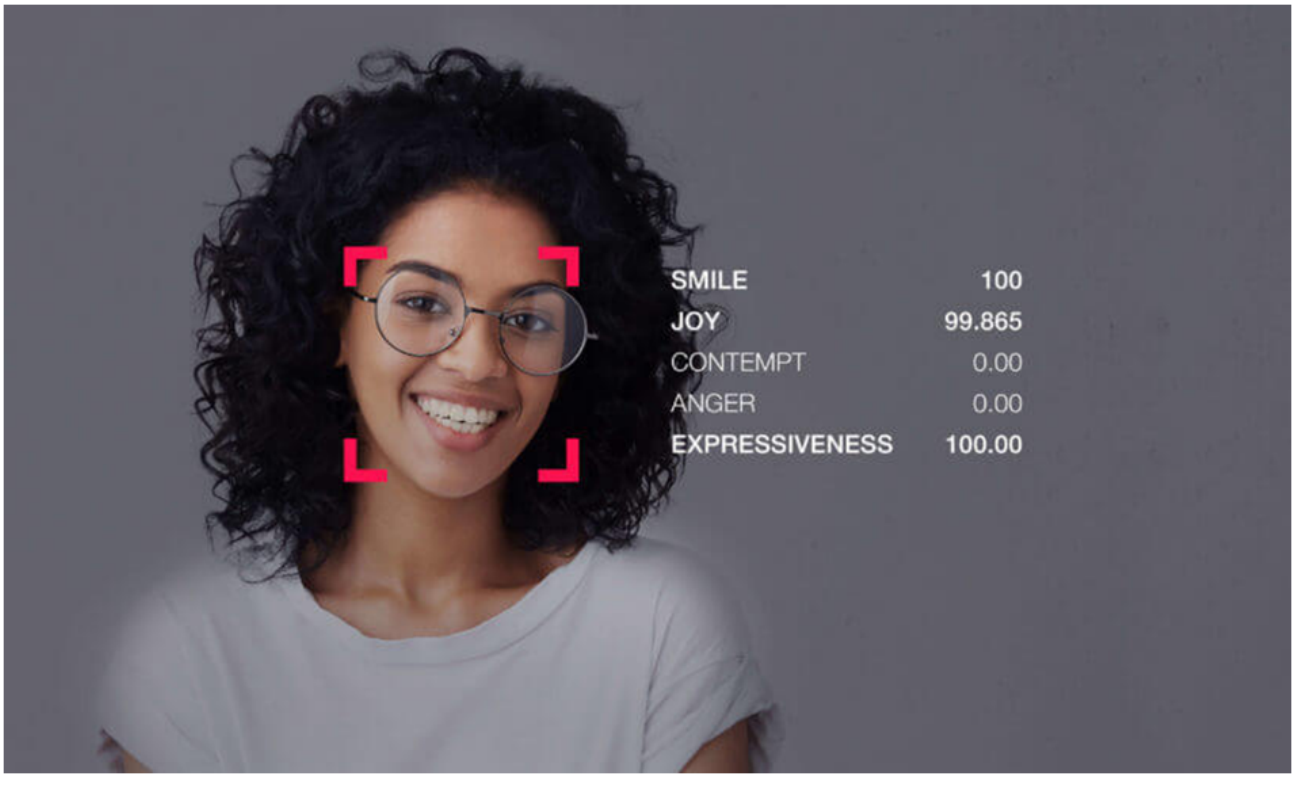
Image Recognition with IBM Cloud Visual Recognition

Phase 4 Submission Document

Project Title : Facial-emotion-recognition-AI



Introduction:

· In this second development phase, we venture into the integration of face detection and emotion recognition, forming a comprehensive image recognition system.

· At the heart of this endeavor lies the "Haar Cascade Classifier," a cutting-edge technology in computer vision primarily designed for detecting frontal faces within images.

· Its role in identifying and locating faces is pivotal, providing a foundation for subsequent emotion recognition tasks.

· The phase also involves a series of code snippets designed to facilitate the development of this system.

· It begins with the preparation of a machine learning model for facial emotion recognition, configuring data generators, creating and fine-tuning the deep learning model, and preparing it for training.

· Users have the option to select from a range of pre-trained deep learning models for emotion recognition, each with its unique architecture.

· The training and evaluation process monitors the model's performance, with early stopping mechanisms and insightful visualizations of accuracy and loss.

· Additionally, the option to save the trained model and associated performance metrics adds a layer of practicality to this multifaceted image recognition project.

This phase bridges the world of face detection and emotion recognition, offering a comprehensive solution for image analysis and understanding

Datasets:

Source: https://github.com/misbah4064/facial_expressions.git

this source provide the required dataset for training the image for facial emotion expression. It includes seven different emotion labels, making it suitable for training and testing emotion recognition models.

STEP 1 : Clone Repository

```
!git clone https://github.com/misbah4064/facial_expressions.git
```

STEP 2 : Creating necessary directories

```
%cd facial_expressions/

%mkdir -p data_set/{anger,happy,neutral,sad,surprise}
```

STEP 3 :Extracting Images with expressions

```
import cv2
with open('happy.txt','r') as f:
    img = [line.strip() for line in f]
for image in img:
    loadedImage = cv2.imread("images/"+image)
    cv2.imwrite("data_set/happy/"+image,loadedImage)
print("done writing")
```

STEP 4: Creating Dataset Folder

```
%mkdir dataset
```

STEP 5:Creating Data Set of Faces

```
import cv2
with open('surprise.txt','r') as f:
    images = [line.strip() for line in f]
face_detector = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')

# For each Emotion, enter one numeric face id
face_id = input('\n Enter Emotion id end press <return> ==>  ')
count = 0
for image in images:
    img = cv2.imread("data_set/surprise/"+image)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = face_detector.detectMultiScale(gray, 1.3, 5)
for (x,y,w,h) in faces:
 cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2)
        count += 1
# Save the captured image into the datasets folder
        cv2.imwrite("dataset/User." + str(face_id) + '.' + str(count) + ".jpg", gray[y:y+h,x:x+w])
print("\n Done creating face data")
```

STEP 6: Creating Trainer Folder

```
%mkdir trainer
```

STEP 7:_Training Images

```
import cv2
import numpy as np
from PIL import Image
import os
# Path for face image database
path = 'dataset'
recognizer = cv2.face.LBPHFaceRecognizer_create()
detector = cv2.CascadeClassifier("haarcascade_frontalface_default.xml");

# function to get the images and label data
def getImagesAndLabels(path):
 imagePaths = [os.path.join(path,f) for f in os.listdir(path)]
    faceSamples=[]
    ids = []
 for imagePath in imagePaths:
      PIL_img = Image.open(imagePath).convert('L') # convert it to grayscale
        img_numpy = np.array(PIL_img,'uint8')
       id = int(os.path.split(imagePath)[-1].split(".")[1])
        faces = detector.detectMultiScale(img_numpy)
```

```
for (x,y,w,h) in faces:
        faceSamples.append(img_numpy[y:y+h,x:x+w])
        ids.append(id)
return faceSamples,ids
print ("\n [INFO] Training faces....")
faces,ids = getImagesAndLabels(path)
recognizer.train(faces, np.array(ids))
# Save the model into trainer/trainer.yml
recognizer.write('trainer/trainer.yml')

# Print the numer of Emotions trained and end program
print("\n [INFO] {0} Emotions trained. Exiting Program".format(len(np.unique(ids))))
```

STEP 8: Recognition (Testing)

```
import cv2
import numpy as np
import os

recognizer = cv2.face.LBPHFaceRecognizer_create()
recognizer.read('trainer/trainer.yml')
cascadePath = "haarcascade_frontalface_default.xml"
faceCascade = cv2.CascadeClassifier(cascadePath);

font = cv2.FONT_HERSHEY_SIMPLEX

#iniciate id counter
id = 0

# Emotions related to ids: example ==> Anger: id=0,  etc
names = ['Anger', 'Happy', 'neutral', 'sad', 'surprise', 'None']

# Initialize and start realtime video capture
cam = cv2.VideoCapture(0)
cam.set(3, 640) # set video widht
cam.set(4, 480) # set video height

# Define min window size to be recognized as a face
minW = 0.1*cam.get(3)
minH = 0.1*cam.get(4)

# ret, img =cam.read()
img = cv2.imread("dwayne.jpg")
# img = cv2.flip(img, -1) # Flip vertically

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

faces = faceCascade.detectMultiScale(
    gray,
    scaleFactor = 1.2,
```

```
    minNeighbors = 5,
    minSize = (int(minW), int(minH)),
    )
for(x,y,w,h) in faces:
    cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2)
    id, confidence = recognizer.predict(gray[y:y+h,x:x+w])
 # Check if confidence is less them 100 ==> "0" is perfect match
    if (confidence < 100):
        id = names[id]
        confidence = "  {0}%".format(round(100 - confidence))
    else:
        id = "unknown"
        confidence = "  {0}%".format(round(100 - confidence))
       cv2.putText(img, str(id), (x+5,y-5), font, 1, (255,255,255), 2)
       cv2.putText(img, str(confidence), (x+5,y+h-5), font, 1, (255,255,0), 1)
 cv2.imwrite("dwayne_johnson.jpg",img)
print("\n [INFO] Done detecting and Image is saved")
cam.release()
cv2.destroyAllWindows()
```

STEP 9 : Display Detected Images

```
import cv2
import matplotlib.pyplot as plt
%matplotlib inline
image = cv2.imread("dwayne_johnson.jpg")
height, width = image.shape[:2]
resized_image = cv2.resize(image,(3*width, 3*height), interpolation = cv2.INTER_CUBIC)
fig = plt.gcf()
fig.set_size_inches(18, 10)
plt.axis("off")
plt.imshow(cv2.cvtColor(resized_image, cv2.COLOR_BGR2RGB))
plt.show()
```