Mined Hackathon 2023

Company Name : JK Lakshmi Cement Ltd. & Udaipur Cement Works Ltd.
TRACK NAME: MANUFACTURING
TEAM NAME:- The Matrix
UNIVERSITY NAME- Nirma University

Group Members:-
UTSAV HARIDWARI (Team Leader)(20BCE302)
RIKIN SOLANKI (20BCE284)
MAYUR VADHADIYA(20BCE303)

Libraries Used:-
For the Project we have used the following libraries:-

Cv2-: cv2 is a popular library for vision tasks in images and video processing.
Numpy:- Numpy library is a library used to deal with array operations.
Pandas:- For creation of dataframe and storing the data into csv file.
Keras.models (Load_model) :- From keras API we have imported load_model to import specific model.
Os- os library is used to access files in user's machine.
MTCNN- It is library for face detection. We have applied MTCNN fo face detection.
Datetime- it is used to track time for our frame when face is detected.

Keras Facenet- keras facenet uses a convolutional neural network (CNN) to generate a high-dimensional embedding for a face image that can be used for face recognition and clustering.
GC- for memory management

```
import cv2
import numpy as np
import pandas as pd
from keras.models import load_model
import os
from mtcnn import MTCNN
import tensorflow as tf
from scipy.spatial.distance import cosine
from datetime import datetime
from keras_facenet import FaceNet
import gc
```

Embeddings and Detector:
They consist of Facenet and MTCNN.

```
embedder = FaceNet()
detector = MTCNN()
```

Storing persons and person embeddings.-

Now we loading the images in the folder and storing it into the dictionary, key of the dictionary is the name of the file and value of the dictionary is the image file.

```
path = 'dataset1/img/'
list_dir = os.listdir(path)
ids = []
persons = {}
persons_embedding = {}

# Load the images of all persons to be recognized
# person1 = cv2.imread('person1.jpg')
```

```
# person1_embedding = None

for img in list_dir:
    ids.append(img.replace(".jpg", ""))
    persons[img.replace(".jpg", "")] = cv2.imread(path+img)
    persons_embedding[img.replace(".jpg", "")] = None
```

First we are resizing the image and then expanding the dimensions at axis 0 and then normalising the values of persons in the range of -1, 1.Accordingly we store the embeddings of the persons as values in the person_embeddings dictionary and key is the file name.

```
for k in persons.keys():
    person = persons[k]
    person = cv2.resize(person, (160, 160))
    person = np.expand_dims(person, axis=0)
    person = (person - 127.5) / 128.0
    persons_embedding[k] = embedder.embeddings(person)
```

Face detection and comparison part in video clip :

```
cap = cv2.VideoCapture('dataset1/Video_clip/VID_20230302_170157.mp4')
names = []
detected = []
cnt = 0
speed = 10 # frames to be skiped
```

After that we get details of the frame like width, height, fps.

```
w = cap.get(cv2.CAP_PROP_FRAME_WIDTH)
h = cap.get(cv2.CAP_PROP_FRAME_HEIGHT)
fps = cap.get(cv2.CAP_PROP_FPS)
size = (int(w),int(h))
```

Here we create one video write object type of MP4. and then we provide the output video file name, format, speed, dimensions of the video file.

```
fourcc = cv2.VideoWriter_fourcc(*'MP4V')
out = cv2.VideoWriter('output.mp4', fourcc, int(fps/speed), (400, 400))
```

This is the main loop for video.

```
while cap.isOpened():
    # Read a frame from the video stream
```

Here Reading the frames.

```
    ret, frame = cap.read()
    frame = cv2.resize(frame, (400, 400))
```

Next code executes when the cnt mod speed equals to 0.

```
    if int(cnt%speed) == 0:
        cnt+=1
    else:
        cnt+=1
        continue
```

After detecting the face and comparing it with all other persons encodings.

```
    if ret:
        # Detect faces in the frame using MTCNN
        result = detector.detect_faces(frame)
        # For each face detected, extract the face embedding and
compare it to the embeddings of all persons
        for face in result:
            x, y, w, h = face['box']
            x1, y1 = max(0, x), max(0, y)
            x2, y2 = min(frame.shape[1], x+w), min(frame.shape[0], y+h)
            face_img = frame[y1:y2, x1:x2]
            face_img = cv2.resize(face_img, (160, 160))
            face_img = np.expand_dims(face_img, axis=0)
            face_img = (face_img - 127.5) / 128.0  # make range of -1
to 1
            face_embedding = embedder.embeddings(face_img)
            # print(face_embedding)
```

```
            # Compare the face embedding to the embeddings of all
persons
            for ID in ids:
                if ID not in detected:
                    if cosine(face_embedding[0],
persons_embedding[ID][0]) < 0.4:
                        detected.append(ID)
                        names.append([datetime.now(),ID])
            cv2.rectangle(frame, (x,y), (x+w,y+h), (255,0,0),
thickness=2)
            del(x)
            del(y)
            del(w)
            del(h)
            del(x1)
            del(y1)
            del(x2)
            del(y2)
            del(face_img)
            del(face_embedding)
            gc.collect()
            # del()
        cv2.imshow("clip", frame)
        out.write(frame)

        if (cv2.waitKey(1) & 0xFF) == ord('f'):
            break
cap.release()
out.release()
cv2.destroyAllWindows()
# print(names)
```

Here we making csv file named attendance.csv and adding datetime, id into csv file.

```
df = pd.DataFrame(names, columns=["DateTime", "ID"])
df.to_csv("attendence.csv")
print(df)
```