Assignment 5.1 – SIT 788

Computer Vision

This task detects faces in an image and face attributes for the detected face. Following steps explain the steps required for this task.

Step 1: Creating Resource

Azure Face detection API [2] has been used to detect faces as well as attributes. To use this API first we create the face detection resource in the azure portal. This can be accessed by the combination of endpoint and any one of the keys shown in fig 1.1. For this task, Key 1 has been used.

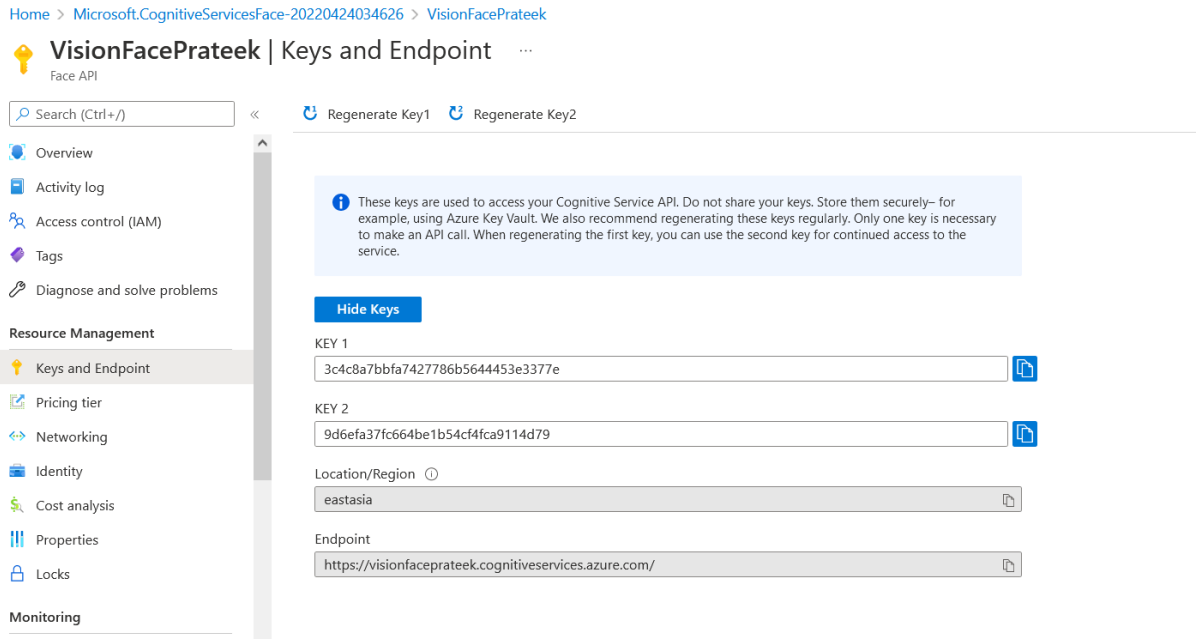

Fig 1.1: Shows the created resource, keys, and endpoint.

Step 2: Installing dependencies

Once the resource has been created, face module from azure cognitive services is installed, this will enable us to call the face detection API.

```
In [11]: !pip install --upgrade azure-cognitiveservices-vision-face
Collecting azure-cognitiveservices-vision-face
  Downloading azure_cognitiveservices_vision_face-0.6.0-py2.py3-none-any.whl (67 kB)
Requirement already satisfied: msrest>=0.5.0 in c:\users\singh\anaconda3\lib\site-packages (from azure-cognitiveservices-vision
-face) (0.6.21)
Requirement already satisfied: azure-common~=1.1 in c:\users\singh\anaconda3\lib\site-packages (from azure-cognitiveservices-vi
sion-face) (1.1.28)
Requirement already satisfied: requests-oauthlib>=0.5.0 in c:\users\singh\anaconda3\lib\site-packages (from msrest>=0.5.0->azur
e-cognitiveservices-vision-face) (1.3.1)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\singh\anaconda3\lib\site-packages (from msrest>=0.5.0->azure-cogn
itiveservices-vision-face) (2020.12.5)
Requirement already satisfied: isodate>=0.6.0 in c:\users\singh\anaconda3\lib\site-packages (from msrest>=0.5.0->azure-cognitiv
eservices-vision-face) (0.6.1)
Requirement already satisfied: requests~=2.16 in c:\users\singh\anaconda3\lib\site-packages (from msrest>=0.5.0->azure-cognitiv
eservices-vision-face) (2.25.1)
Requirement already satisfied: six in c:\users\singh\anaconda3\lib\site-packages (from isodate>=0.6.0->msrest>=0.5.0->azure-cog
nitiveservices-vision-face) (1.15.0)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in c:\users\singh\anaconda3\lib\site-packages (from requests~=2.16->msrest
>=0.5.0->azure-cognitiveservices-vision-face) (1.26.4)
Requirement already satisfied: idna<3,>=2.5 in c:\users\singh\anaconda3\lib\site-packages (from requests~=2.16->msrest>=0.5.0->
azure-cognitiveservices-vision-face) (2.10)
Requirement already satisfied: chardet<5,>=3.0.2 in c:\users\singh\anaconda3\lib\site-packages (from requests~=2.16->msrest>=0.
5.0->azure-cognitiveservices-vision-face) (4.0.0)
Requirement already satisfied: oauthlib>=3.0.0 in c:\users\singh\anaconda3\lib\site-packages (from requests-oauthlib>=0.5.0->ms
rest>=0.5.0->azure-cognitiveservices-vision-face) (3.2.0)
Installing collected packages: azure-cognitiveservices-vision-face
Successfully installed azure-cognitiveservices-vision-face-0.6.0
```

Fig 2.1: Installing azure cognitive service face module

Step 3: Importing libs

After installing the dependencies, the libraries are imprted. Opencv and matplotlib are used for displaying the results. Azure libs are used for calling face detection API.

```
In [4]: from msrest.authentication import CognitiveServicesCredentials
        from azure.cognitiveservices.vision.face import FaceClient

        import cv2
        from IPython.display import Image
        from matplotlib import pyplot as plt
```

Fig 3.1: Importing required libs

Step 4: Connecting to the resource

Now, we connect to the resource created in **step 1,** this is done by creating a client which reaches out to the specified endpoint using the secret key associated with the resource.

```
In [5]: subscription_key = "3c4c8a7bbfa7427786b5644453e3377e"
        endpoint = "https://visionfaceprateek.cognitiveservices.azure.com/"
        cv_face_client = FaceClient(endpoint, CognitiveServicesCredentials(subscription_key))
```

Fig 4.1: Connecting to resource

Step 5: Reading image and detecting faces

To detect faces, images from Flickr-Faces-HQ (FFHQ) dataset [1] are used. The image is read from the disk into memory and then face detection API is called. Because the task also focusses on face attributes, a list specifying the attributes **[age, emotion, gender, and accessories]** to be detected is also passed along with the image. Age returns a floating-point value; emotion returns dictionary of emotions with corresponding confidence value and gender returns string (male/female). These attributes are returned for each face detected in the image.

```
In [16]: image_name = "./SIT788_5_1_Data//69999_test_2000.png"
         #image_name = "./SIT788_5_1_Data//69954.png"
         image = open(image_name, 'rb')
```

```
In [18]: #Detect face with age, emotion, gender and accessories
         attributes_list = ['age', 'emotion', 'gender', 'accessories']
         detected_faces = cv_face_client.face.detect_with_stream(image,
                                                                 return_face_attributes=attributes_list)
         if len(detected_faces) > 0:
             print ("Number of faces detected: ", len(detected_faces))
             faces_rect = []
             for faces_itr in range(len(detected_faces)):
                 faces_rect.append([detected_faces[faces_itr].face_rectangle.left, detected_faces[faces_itr].face_rectangle.top,
                 detected_faces[faces_itr].face_rectangle.width,
                 detected_faces[faces_itr].face_rectangle.height])
         else:
             print("No face detected")
         print("List of face coordinates: ", faces_rect)

         Number of faces detected:  3
         List of face coordinates:  [[1225, 180, 213, 213], [518, 330, 193, 193], [867, 425, 175, 175]]
```

Fig 5.1: Reading image and calling the detection API

Step 6: Processing detection results

Once we get the detection results, we can process it for each face detected. The processing step includes iterating through each detected face and parsing out age, gender, emotion, and accessory information associated with the face.

- For age and gender the values for the detected face are directly used.
- For emotion, the emotion with highest confidence value is used.
- For accessory, all the detected accessories are iterated and displayed for the face.

```
In [19]: disp_image = cv2.imread(image_name)

for faces_itr in range(len(detected_faces)):
    gender = detected_faces[faces_itr].face_attributes.gender.value
    age = detected_faces[faces_itr].face_attributes.age
    emotions_dict = detected_faces[faces_itr].face_attributes.emotion.as_dict()
    emotion_name = max(zip(emotions_dict.values(), emotions_dict.keys()))[1]

    org = (detected_faces[faces_itr].face_rectangle.left,
           detected_faces[faces_itr].face_rectangle.top)
    font = cv2.FONT_HERSHEY_SIMPLEX
    fontScale = 4
    color = (0, 0,255)
    thickness = 10
    text = "Face" + str(faces_itr)

    image = cv2.putText(disp_image, text, org, font,
                    fontScale, color, thickness, cv2.LINE_AA)


    disp_image = cv2.rectangle(disp_image, faces_rect[faces_itr], (0,0,255), 5)

    print ("Age for Face", faces_itr, ":", age, "Emotion: ", emotion_name, "Gender:", gender)
    accessories_list = detected_faces[faces_itr].face_attributes.accessories
    if len(accessories_list) > 0:
        for acc_itr in range(len(accessories_list)):
            print("Accessories:", accessories_list[acc_itr].as_dict(), "for Face",
              faces_itr)
    else:
        print("No Accessories found for Face",
              faces_itr)
    print("_____________________________")
disp_image_ = cv2.cvtColor(disp_image, cv2.COLOR_BGR2RGB)
plt.imshow(disp_image_)
plt.show()
```

Fig 6.1: Processing and parsing the results associated with each face.

Output image shows the face with red bounding box and their serial number. All the attributes associated are printed on the console preceding the image.

All the outputs received from the face detection API are shown in fig 6.2 and fig 6.3.

In the first example we can see that the face detection API detects three faces. While **no accessory** is found for face 0 and **glasses** are found for face1 and face2 with 94% confidence in both the cases.

```
Age for Face 0 : 52.0 Emotion:  happiness Gender: male
No Accessories found for Face 0
_____________________________
Age for Face 1 : 64.0 Emotion:  happiness Gender: male
Accessories: {'type': 'glasses', 'confidence': 0.94} for Face 1
_____________________________
Age for Face 2 : 58.0 Emotion:  happiness Gender: female
Accessories: {'type': 'glasses', 'confidence': 0.94} for Face 2
_____________________________
```

Fig 6.2: Image Source [1]

In this example, a different accessory has been detected as **headwear** with 99% confidence.

```
Age for Face 0 : 60.0 Emotion:  neutral Gender: male
Accessories: {'type': 'headWear', 'confidence': 0.99} for Face 0
_____________________________
```

Fig 6.3: Image Source [1]

References

[1] https://github.com/NVlabs/ffhq-dataset

[2] https://azure.microsoft.com/en-us/services/cognitive-services/face/