Project Jupyter Notebook

May 16, 2022

0.0.1 Please install required packages

```
In [1]: ! activate ai-azure-c1

        import sys

        sys.path.append("/opt/conda/envs/ai-azure-c1/lib/python3.8/site-packages")
```

```
In [25]: import io, glob, os, sys, time, uuid

         from matplotlib.pyplot import imshow
         import matplotlib.pyplot as plt

         from io import BytesIO
         from PIL import Image, ImageDraw

         from video_indexer import VideoIndexer
         from azure.cognitiveservices.vision.face import FaceClient
         from azure.cognitiveservices.vision.face.models import TrainingStatusType
         from azure.core.credentials import AzureKeyCredential
         from msrest.authentication import CognitiveServicesCredentials
```

```
In [11]: CONFIG = {
             'SUBSCRIPTION_KEY': '861997d23acc452798f4a5b2f526f1a1',
             'LOCATION': 'trial',
             'ACCOUNT_ID': 'd7ecf698-c493-4938-9d6e-6eebaae25d98'
         }

         va = VideoIndexer(
             vi_subscription_key=CONFIG['SUBSCRIPTION_KEY'],
             vi_location=CONFIG['LOCATION'],
             vi_account_id=CONFIG['ACCOUNT_ID']
         )
```

```
In [12]: vid ='51351402b5'
```

```
In [14]: info = va.get_video_info(vid, video_language='English')
```

```
Getting video info for: 51351402b5
```

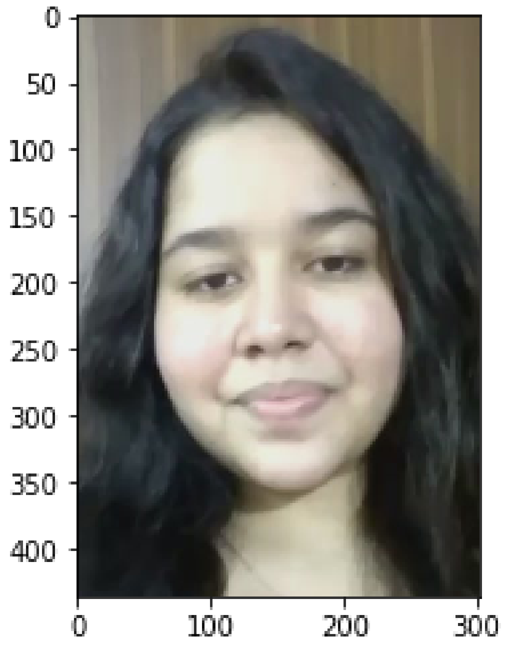
```
In [15]: if len(info['videos'][0]['insights']['faces'][0]['thumbnails']):
             print("We found {} faces in this video.".format(str(len(info['videos'][0]['insights
```

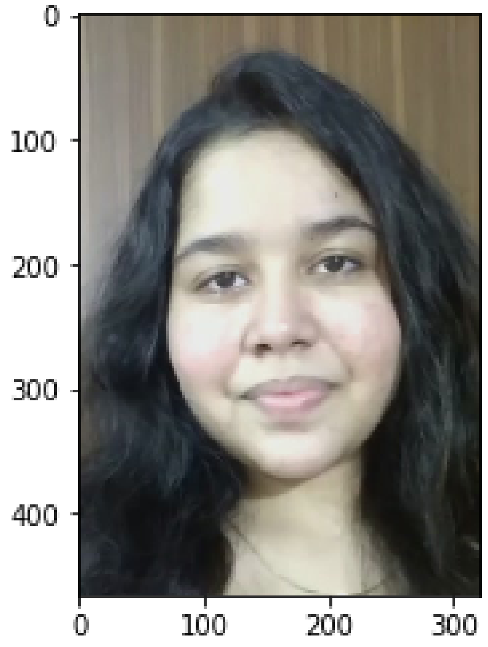
```
We found 7 faces in this video.
```

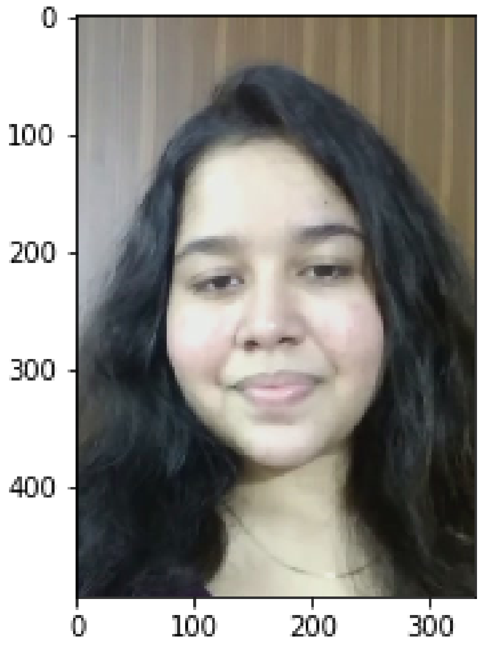
```
In [16]: images = []
         img_raw = []
         img_strs = []
         for each_thumb in info['videos'][0]['insights']['faces'][0]['thumbnails']:
             if 'fileName' in each_thumb and 'id' in each_thumb:
                 file_name = each_thumb['fileName']
                 thumb_id = each_thumb['id']
                 img_code = va.get_thumbnail_from_video_indexer(vid,  thumb_id)
                 img_strs.append(img_code)
                 img_stream = io.BytesIO(img_code)
                 img_raw.append(img_stream)
                 img = Image.open(img_stream)
                 images.append(img)
```

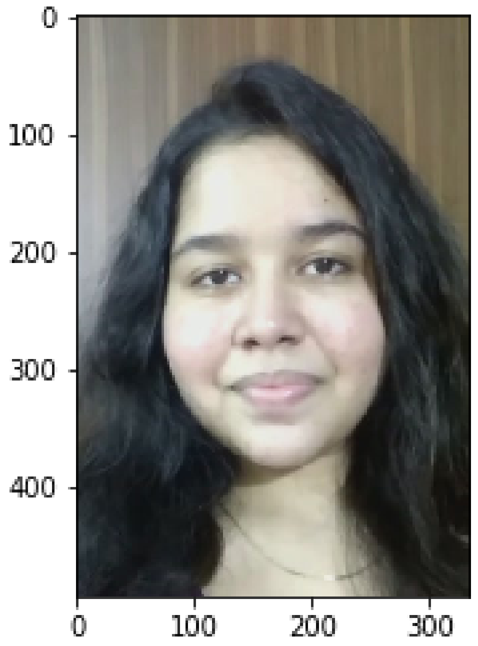
```
Getting thumbnail from video: 51351402b5, thumbnail: 0ea3d06e-4488-4d46-85e0-440263faaac5
Getting thumbnail from video: 51351402b5, thumbnail: c5bdcf40-2bc9-4fcf-b070-5133ba9120ac
Getting thumbnail from video: 51351402b5, thumbnail: 310781f7-14f2-4f74-a1f9-01507f6839a0
Getting thumbnail from video: 51351402b5, thumbnail: 79951e38-dc33-49a0-b3dd-fc39144b9df3
Getting thumbnail from video: 51351402b5, thumbnail: 6ea03188-db15-422d-98c3-0614f7afb302
Getting thumbnail from video: 51351402b5, thumbnail: b4b75df2-9559-456e-9b3c-a87a5c9d5beb
Getting thumbnail from video: 51351402b5, thumbnail: 02bd9e4c-7a00-4e24-bc36-aff04768f6d6
```

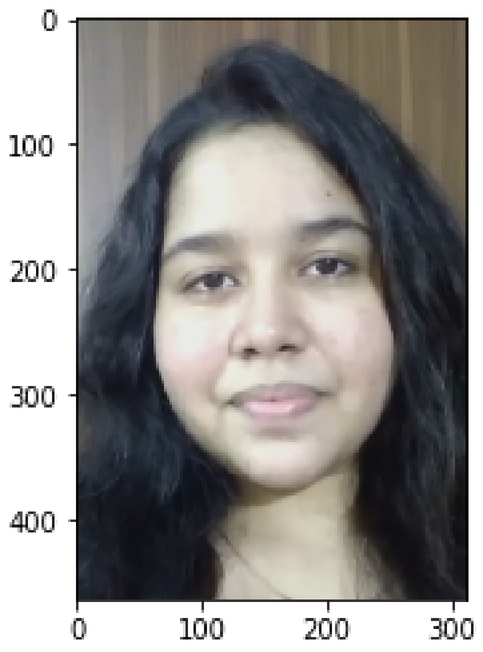
```
In [19]: for img in images:
             plt.figure()
             plt.imshow(img)
```

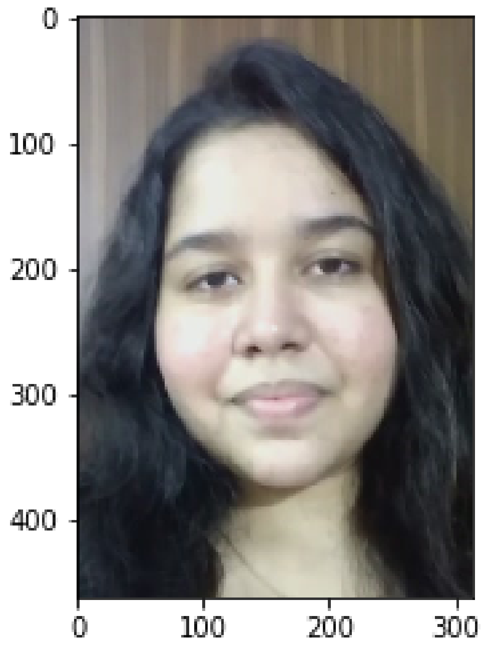

0
50
100
150
200
250
300
350
400
0
100
200
300

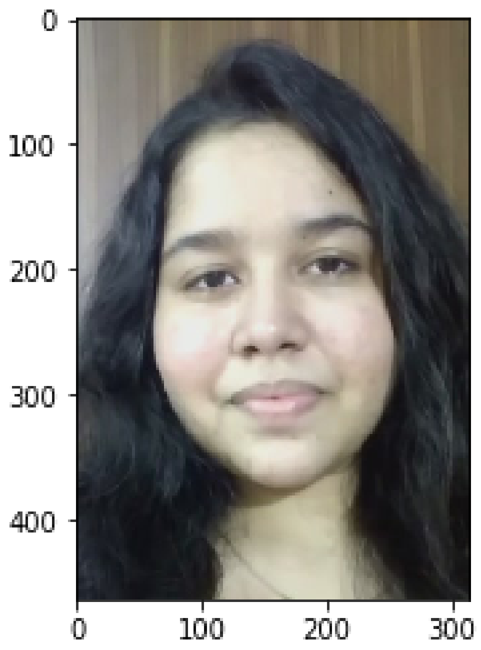

0
100
200
300
400
0
100
200
300


0
100
200
300
400
0
100
200
300


0
100
200
300
400
0
100
200
300


0
100
200
300
400
0
100
200
300


0
100
200
300
400
0
100
200
300



```
In [20]: i = 1
         for img in images:
             img.save('face1' + str(i) + '.jpg')
             i= i+ 1

In [21]: !ls face1*.jpg

face11.jpg  face13.jpg        face15.jpg  face17.jpg
face12.jpg  face14.jpg        face16.jpg


In [22]: info['summarizedInsights']['sentiments']
         info['summarizedInsights']['emotions']

Out[22]: []

In [23]: facekey='8128c6fcee8f4100a601689fbb845712'
         faceendpoint='https://faceapiudacity.cognitiveservices.azure.com/'

In [26]: faceclient = FaceClient(faceendpoint, CognitiveServicesCredentials(facekey))

In [29]: def build_person_group(client, person_group_id, pgp_name):
             print('Create and build a person group...')
             # Create empty Person Group. Person Group ID must be lower case, alphanumeric, and/
             print('Person group ID:', person_group_id)
             client.person_group.create(person_group_id = person_group_id, name=person_group_id)
```

```
        # Create a person group person.
        human_person = client.person_group_person.create(person_group_id, pgp_name)
        # Find all jpeg human images in working directory.
        human_face_images = [file for file in glob.glob('*.jpg') if file.startswith("face1"
        # Add images to a Person object
        for image_p in human_face_images:
            with open(image_p, 'rb') as w:
                client.person_group_person.add_face_from_stream(person_group_id, human_pers

        # Train the person group, after a Person object with many images were added to it.
        client.person_group.train(person_group_id)

        # Wait for training to finish.
        while (True):
            training_status = client.person_group.get_training_status(person_group_id)
            print("Training status: {}.".format(training_status.status))
            if (training_status.status is TrainingStatusType.succeeded):
                break
            elif (training_status.status is TrainingStatusType.failed):
                client.person_group.delete(person_group_id=PERSON_GROUP_ID)
                sys.exit('Training the person group has failed.')
            time.sleep(5)

    build_person_group(faceclient,'nandhini18','Nandhini1')
```

```
Create and build a person group...
Person group ID: nandhini18
Training status: running.
Training status: succeeded.
```

```
In [30]: def detect_faces(client, images):
             print('Detecting faces in images ...')

             face_ids = {} # Keep track of the image ID and the related image in a dictionary
             for image_name in images:
                 image = open(image_name, 'rb') # BufferedReader
                 print("Opening image: ", image.name)
                 time.sleep(5)

                 # Detect the faces in the query images list one at a time, returns list[Detecte
                 faces = client.face.detect_with_stream(image)

                 # Add all detected face IDs to a list
                 for face in faces:
                     print('Face ID', face.face_id, 'found in image', os.path.splitext(image.nam
                     # Add the ID to a dictionary with image name as a key.
```

```
                    # This assumes there is only one face per image (since you can't have dupli
                    face_ids[image.name] = face.face_id

            return face_ids
```

```
In [31]: images = [file for file in glob.glob('*.jpg') if file.startswith("face1")]
         ids = detect_faces(faceclient, images)
```

```
Detecting faces in images ...
Opening image:  face11.jpg
Face ID 4632e6e3-376e-4ddb-818d-142bbeb57cf0 found in image face11.jpg
Opening image:  face15.jpg
Face ID 0bae86fa-ef95-4a6c-9ae7-0ac9b66976a3 found in image face15.jpg
Opening image:  face12.jpg
Face ID 3bc8049c-d5a8-4ef1-92ea-2a088d6dda7c found in image face12.jpg
Opening image:  face16.jpg
Face ID fda45492-8ac9-4167-8624-80d9d6b80ee2 found in image face16.jpg
Opening image:  face14.jpg
Face ID 186d3227-93ea-40e5-a317-d428b10ac0e3 found in image face14.jpg
Opening image:  face13.jpg
Face ID fc8d56fc-afbe-4600-906c-ab493d49f13a found in image face13.jpg
Opening image:  face17.jpg
Face ID 91d924e1-cc89-4bce-bf4b-6c3a6a4dd2df found in image face17.jpg
```

```
In [32]: dl_image = open('Nandhinidl.jpeg', 'rb')
         dl_faces = faceclient.face.detect_with_stream(dl_image)
```

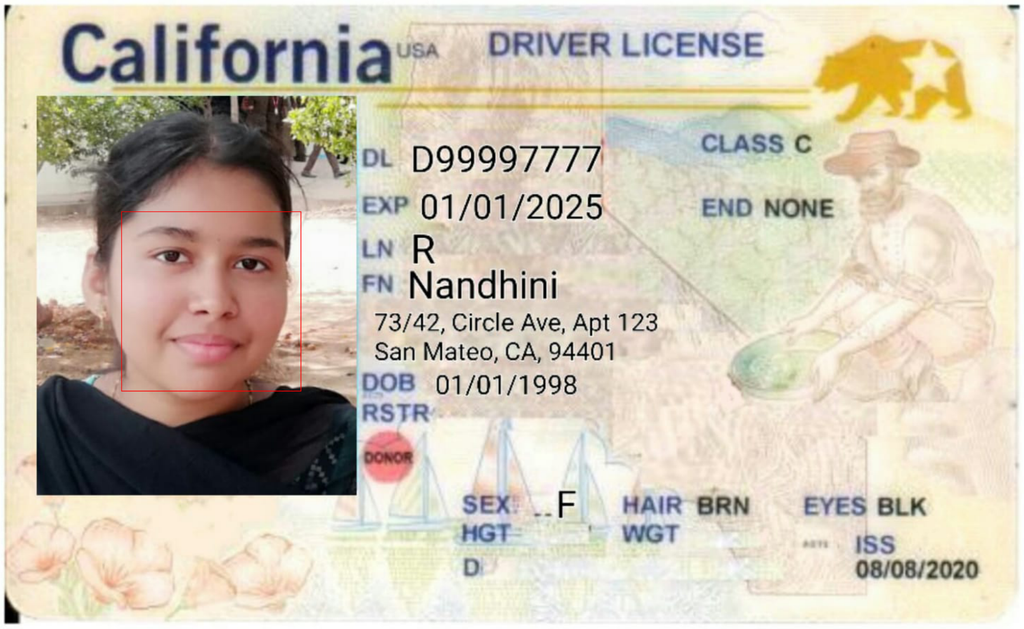
```
In [39]: def getRectangle(faceDictionary):
             rect = faceDictionary.face_rectangle
             left = rect.left
             top = rect.top
             right = left + rect.width
             bottom = top + rect.height

             return ((left, top), (right, bottom))

         def drawFaceRectangles(source_file, detected_face):
             img = Image.open(source_file)
             # Draw a red box around every detected faces
             draw = ImageDraw.Draw(img)
             for face in detected_face:
                 draw.rectangle(getRectangle(face), outline='red')
             return img
```

```
In [40]: drawFaceRectangles(dl_image, dl_faces)
```

```
Out[40]:
```


California USA DRIVER LICENSE

DL D99997777

EXP 01/01/2025

LN R

FN Nandhini

73/42, Circle Ave, Apt 123
San Mateo, CA, 94401

DOB 01/01/1998

RSTR

CLASS C

END NONE

DONOR

SEX F

HGT

D

HAIR BRN

WGT

EYES BLK

ISS 08/08/2020

```
In [41]: for face in dl_faces:
             ids['nandhinidl.jpg'] = face.face_id

In [42]: ids

Out[42]: {'face11.jpg': '4632e6e3-376e-4ddb-818d-142bbeb57cf0',
          'face15.jpg': '0bae86fa-ef95-4a6c-9ae7-0ac9b66976a3',
          'face12.jpg': '3bc8049c-d5a8-4ef1-92ea-2a088d6dda7c',
          'face16.jpg': 'fda45492-8ac9-4167-8624-80d9d6b80ee2',
          'face14.jpg': '186d3227-93ea-40e5-a317-d428b10ac0e3',
          'face13.jpg': 'fc8d56fc-afbe-4600-906c-ab493d49f13a',
          'face17.jpg': '91d924e1-cc89-4bce-bf4b-6c3a6a4dd2df',
          'nandhinidl.jpg': '1b57ad3b-c7bd-4f30-8faf-5fb680e91090'}

In [43]: verify_face= faceclient.face.verify_face_to_face(ids['face13.jpg'], ids['nandhinidl.jpg

         if verify_face.is_identical:
             print("Faces are of the same (Positive) person, similarity confidence: {}.".format(
         else:
             print("Faces are of different (Negative) persons, similarity confidence: {}.".forma

Faces are of the same (Positive) person, similarity confidence: 0.67269.


In [ ]:
```