Computer Vision Face Detection Lab - Student Notebook

In this lab, you will use Amazon Rekognition to perform a facial detection of a known face.

The high-level steps you will perform in this lab are:

1. Create a collection.
2. Upload an image of a face that you want to detect to the Amazon SageMaker notebook.
3. Add the image to the collection.
4. View the bounding box that was created for the image.
5. List the faces in the collection.
6. Use the collection to find a face.
7. View the bounding box of the face that was found.
8. Delete the collection.

Importing Python packages

Start by importing the Python packages that you need.

In the following code:

- *matplotlib* provides plotting functions
- *skimage* represents scikit-image, which provides several useful image manipulation tools
- *boto3* represents the AWS SDK for Python (Boto3), which is the Python library for AWS
- *numpy* represents NumPy, which is a library for manipulating data
- *PIL* represents the Python Imaging Library, which contains a set of tools for drawing images

In [ ]:
```
from skimage import io
from skimage.transform import rescale
from matplotlib import pyplot as plt

import boto3

import numpy as np
```

```
from PIL import Image, ImageDraw, ImageColor, ImageOps
```

Task 1: Creating a collection

In this task, you create a collection in Amazon Rekognition.

You only need to run this step once.

In [ ]:
```
client = boto3.client('rekognition')
collection_id = 'Collection'
response = client.create_collection(CollectionId=collection_id)
print('Collection ARN: ' + response['CollectionArn'])
print('Status Code:' + str(response['StatusCode']))
print('Done...')
```

Task 2: Uploading an image to search

Use the provided sample image, which is named *mum.jpg*, and upload it to this notebook.

Then, look at the image by running the cell.

In [ ]:
```
filename = "mum.jpg"

faceimage = io.imread(filename)

plt.imshow(faceimage)
```

Make sure that the image size is less than 4096 x 4096 pixels. If the image is larger than this size, then you must resize it by using the following code:

`faceimage = rescale(faceimage, 0.50, mode='constant')`

Note: The numeric value represents the factor to scale by. A value of *0.5* will scale the image to 50 percent of the original.

When the image is resized, save the file.

`io.imsave(filename, faceimage)`

Tip: You must copy the code into a code cell and run it.

Task 3: Adding the image to the collection

Add the image to the collection that you created earlier.

In [ ]:
```
externalimageid = filename
```

```
with open(filename, 'rb') as fimage:
    response = client.index_faces(CollectionId = collection_id,
                                  Image={'Bytes': fimage.read()},
                                  ExternalImageId=externalimageid,
                                  MaxFaces=1,
                                  QualityFilter="AUTO",
                                  DetectionAttributes=['ALL'])

print('Results for ' + filename)
print('Faces indexed:')
for faceRecord in response['FaceRecords']:
     print('  Face ID: ' + faceRecord['Face']['FaceId'])
     print('  Location: {}'.format(faceRecord['Face']['BoundingBox']))

print('Faces not indexed:')
for unindexedFace in response['UnindexedFaces']:
    print(' Location: {}'.format(unindexedFace['FaceDetail']['BoundingBox']))
    print(' Reasons:')
    for reason in unindexedFace['Reasons']:
        print('   ' + reason)
```

Task 4: Viewing the bounding box for the detected face

If a face was found, the results should include the location of the face that was detected. Examine the bounding box on the image.

To do this, use the PIL library, which you imported earlier in this lab. By extracting the BoundingBox, you can draw a set of lines around the image.

In [ ]:
```
img = Image.open(filename)
imgWidth, imgHeight = img.size

draw = ImageDraw.Draw(img)
for faceRecord in response['FaceRecords']:
    box = faceRecord['Face']['BoundingBox']
    left = imgWidth * box['Left']
    top = imgHeight * box['Top']
    width = imgWidth * box['Width']
    height = imgHeight * box['Height']

    points = ((left,top),(left+width,top),(left+width,top+height),(left,top+height)

    draw.line(points,fill='#00d400', width=15)

plt.imshow(img)
```

Task 5: Listing faces in the collection

Examine the image that you have in the collection.

In [ ]:
```
maxResults=2
faces_count=0
tokens=True

response=client.list_faces(CollectionId=collection_id,
                           MaxResults=maxResults)

print('Faces in collection ' + collection_id)

while tokens:

    faces=response['Faces']

    for face in faces:
        print (face)
        faces_count+=1
    if 'NextToken' in response:
        nextToken=response['NextToken']
        response=client.list_faces(CollectionId=collection_id,
                                   NextToken=nextToken,MaxResults=maxResults)
    else:
        tokens=False
```

Task 6: Finding a face by using the collection

In this step, you will use the collection to detect a face in an image.

Use the provided sample image that's named *target.jpg* and upload it to this notebook.

In [ ]:
```
targetfilename = "target.jpg"

targetimage = Image.open(targetfilename)
plt.imshow(targetimage)
```

Next, call the `search_faces_by_image` operation and see if you get a match.

In [ ]:
```
threshold = 70
maxFaces=2

with open(targetfilename, 'rb') as timage:
    response2=client.search_faces_by_image(CollectionId=collection_id,
                            Image={'Bytes': timage.read()},
                            FaceMatchThreshold=threshold,
                            MaxFaces=maxFaces)

faceMatches=response2['FaceMatches']
print ('Matching faces')
for match in faceMatches:
        print ('FaceId:' + match['Face']['FaceId'])
        print ('Similarity: ' + "{:.2f}".format(match['Similarity']) + "%")
        print ('ExternalImageId: ' + match['Face']['ExternalImageId'])
        print
```

Task 7: Drawing a bounding box around the discovered face

Draw a bounding box around the discovered face.

In [ ]:
```
imgWidth, imgHeight = targetimage.size

draw = ImageDraw.Draw(targetimage)

box = response2['SearchedFaceBoundingBox']
left = imgWidth * box['Left']
top = imgHeight * box['Top']
width = imgWidth * box['Width']
height = imgHeight * box['Height']

points = ((left,top),(left+width,top),(left+width,top+height),(left,top+height),(le
draw.line(points,fill='#00d400', width=15)

plt.imshow(targetimage)
```

Task 8: Deleting the collection

When you are finished, delete the collection. To do this, run the following code.

In [ ]:
```
print('Attempting to delete collection ' + collection_id)
status_code=0
try:
    response=client.delete_collection(CollectionId=collection_id)
    status_code=response['StatusCode']
    print('All done!')
    print(status_code)

except ClientError as e:
    if e.response['Error']['Code'] == 'ResourceNotFoundException':
        print ('The collection ' + collection_id + ' was not found ')
    else:
        print ('Error other than Not Found occurred: ' + e.response['Error']['Messa
    status_code=e.response['ResponseMetadata']['HTTPStatusCode']
```

Congratulations!

You have completed this lab, and you can now end the lab by following the lab guide instructions.