SEATTLE PACIFIC UNIVERSITY

ISM 6900
Image Mining
Facial Recognition

Done by
-Sheetal Murali

Table of Contents

INTRODUCTION

FACIAL RECOGNITION:

A face analyzer is software that uses a person's face to identify or validate their identification. It recognizes and measures facial traits in images. Facial recognition can recognize human faces in images or videos, determine whether the face in two images belongs to the same person, or search for a face in a big collection of existing images. Facial recognition is used in biometric security systems to uniquely identify persons during user onboarding or logins, as well as to strengthen user authentication activity. Face analyzer technology is also extensively used for device security in mobile and personal devices.

PROCESS OF FACIAL RECOGNITION:

Facial recognition works in three steps: detection, analysis, and recognition.

Detection

The technique of detecting a face in a picture is known as detection. Facial recognition, which is enabled by computer vision, can detect and identify individual faces in an image including one or more people's faces.

Computer Vision

Machines utilize computer vision to identify people, places, and things in photographs with accuracy comparable to or greater than that of humans, and at a far quicker and more effective rate. Computer vision automates the extraction, analysis, classification, and interpretation of meaningful information from picture data by utilizing complicated artificial intelligence (AI) technology.

Analysis

The facial recognition system then analyzes the image of the face. It maps and reads face geometry and facial expressions. It identifies facial landmarks that are key to distinguishing a face from other objects.

The system then turns the face recognition data into a string of numbers or points known as a faceprint. Each person has a distinct faceprint, similar to a fingerprint. The information received from facial recognition can also be utilized to digitally rebuild a person's face.

Recognition

Facial recognition can identify a person by comparing the faces in two or more images and determining how likely it is of a face match. For example, it can verify that the face shown in a selfie taken by a mobile camera matches or does not match the face in an image of a government-issued ID like a driver's license or passport.

AMAZON REKOGNITION

This is a service by Amazon Web Services where we can add image analysis to any application. The Amazon Rekognition service makes it easy to enable facial recognition that can find similar faces in a large collection of images. If we provide an image or video to the Amazon Rekognition API, the service can:

• Identify labels (objects, concepts, people, scenes, and activities) and text
• Detect inappropriate content
• Provide highly accurate facial analysis, face comparison, and face search capabilities

With Amazon Rekognition's face recognition APIs, you can detect, analyze, and compare faces for a wide variety of use cases, including user verification, cataloging, people counting, and public safety. It is based on Amazon's very popular deep learning technology developed by the computer vision scientists to analyze billions of images every day from Amazon Prime Photos. This technology is proven to be highly scalable because it enables us to build a solution to create, maintain and query our own collection of faces. This can be done for many use cases like automated detection of people within a library or in social media tagging, building access control etc.

In this paper I have explained how I have built my own face recognition service by combining the capabilities of Amazon Rekognition with other AWS services like Amazon DynamoDB, Lambda and S3 Bucket. I have used the AWS Amazon Rekognition Developer guide as reference to build the infrastructure

PRE-PROCESS

The Pre-process involves preparing the Resources for indexing. This is a very important part of the process because it acts as an infrastructure for the entire image mining process.

I. IAM Role

To Execute any command in AWS, the user must have permissions in AWS Identity and Access Management (IAM) to perform the actions. In this Facial recognition project, I have added the following policies to the user/role.

- AmazonRekognitionFullAccess
- AmazonDynamoDBFullAccess
- AmazonS3FullAccess
- IAMFullAccess

II. Creating a Collection

First step is to create a collection within Amazon Rekognition. A collection refers to a container for persisting faces detected by Indexfaces API. Depending on the use case, we can create one container to store all the faces or multiple containers to store faces in groups. For my project I have created 1 container for all the faces to find a match for a face within a collection of faces (my face as in this project). Face match can be very useful in whitelisting a group of people for a VIP experience, blacklisting to identify bad actors, or supporting logging scenarios. For this we would create a single collection that contains a large number of faces.

Command for creating collection in AWS CLI:

aws rekognition create-collection --collection-id face_recognition --region us-west-2

III. Create Dynamodb Table.

DynamoDB is a fully managed cloud database that supports both document and key-value store models. Amazon DynamoDB is a serverless, NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools.

In this project, I have created a DynamoDB table and used it as a simple key-value store to maintain a reference of the FaceId returned from Amazon Rekognition and my full name (or the person)

Command for creating Dynamodb Table in AWS CLI:

```
aws dynamodb create-table --table-name face_recognition \
--attribute-definitions AttributeName=RekognitionId,AttributeType=S \
--key-schema AttributeName=RekognitionId,KeyType=HASH \
--provisioned-throughput ReadCapacityUnits=1,WriteCapacityUnits=1 \
 --region us-west-2
```

IV. Creating S3 bucket

Amazon Simple Storage Service (Amazon S3) is an object storage service that offers industry-leading scalability, data availability, security, and performance. We can upload the images to an Amazon S3 bucket for the IndexFaces operation, I have made the images available to Amazon Rekognition inside an Amazon S3 bucket. An *object* is a file and any metadata that describes the file. A *bucket* is a container for objects. To store your data in Amazon S3, you first create a bucket and specify a bucket name and AWS Region. Then, you upload your data to that bucket as objects in Amazon S3. Each object has a *key name*, which is the unique identifier for the object within the bucket. S3 provides features that you can configure to support your specific use case. For example, you can use S3 Versioning to keep multiple versions of an object in the same bucket, which allows you to restore objects that are accidentally deleted or overwritten.

Buckets and the objects in them are private and can be accessed only if you explicitly grant access permissions. You can use bucket policies, AWS Identity and Access Management (IAM) policies, access control lists (ACLs), and S3 Access Points to manage access.

We can create a bucket either from the AWS Management Console or from the AWS CLI by Using the following command, which is documented in the CLI Command Reference.

Command for creating S3 bucket in AWS CLI:

aws s3 mb s3://muralis-face-recognition --region us-west-2

V. AWS Lambda Function

AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers. You can trigger Lambda from over 200 AWS services and software as a service (SaaS) applications, and only pay for what you use.

In the architecture diagram in Figure 1, we have seen that the process of Facial recognition is of 2 parts. Although I have set up the infrastructure in AWS CLI, Lambda function needs to be used to process the images to the Amazon S3 bucket.

VI. Granting Lambda function Access to initiate the process (Analysis).

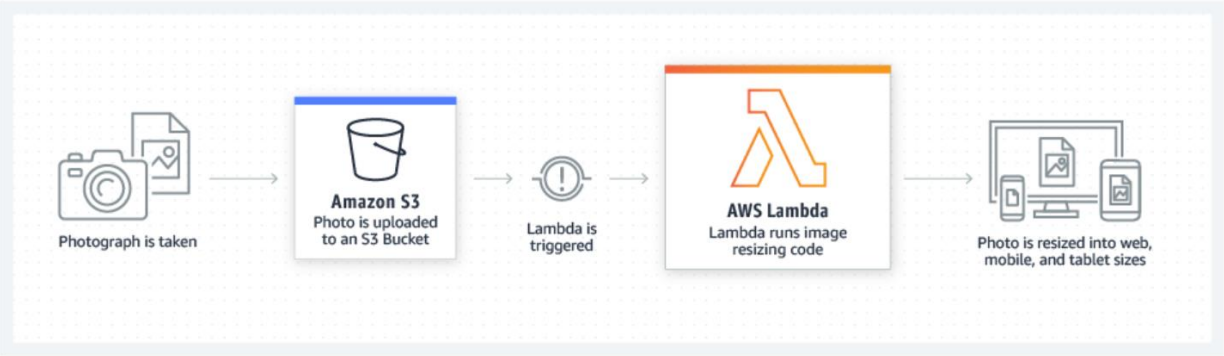

Figure 2. File processing when Lambda function is triggered

For this, It is important to create an IAM role that grants our function the rights to access the objects from Amazon S3, initiate the IndexFaces function of Amazon Rekognition, and create multiple entries within Amazon DynamoDB key-value store for a mapping between the FaceId and the person's full name.

To create the service role for Lambda, I have used the console to allow Lambda to “Assume role” and used JSON files that describe the trust and access policies: As described in the documentation, I first created the role that includes the trust policy and this is followed by attaching the actual access policy to the role.

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
         ],
         "Resource": "arn:aws:logs:*:*:*"
      },
      {
         "Effect": "Allow",
         "Action": [
            "s3:GetObject"
         ],
         "Resource": [
            "arn:aws:s3:::muralis-face-recognition/*"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "dynamodb:PutItem"
         ],
         "Resource": [
            "arn:aws:dynamodb:us-west-2:179074980226:table/face_recognition"
         ]
      },
      {
         "Effect": "Allow",
         "Action": [
            "rekognition:IndexFaces"
         ],
         "Resource": "*"
      }
   ]
```

PROCESS

The process of Image mining using Amazon Rekognition is separated into 2 main parts:

1. INDEXING – Blue flow
2. ANALYSIS – Black Flow

The figure 1 below shows the application workflow in which Blue flow is the process of importing images of faces into the collection for analysis and black flow is the process of querying the collection of faces for matches within the index.

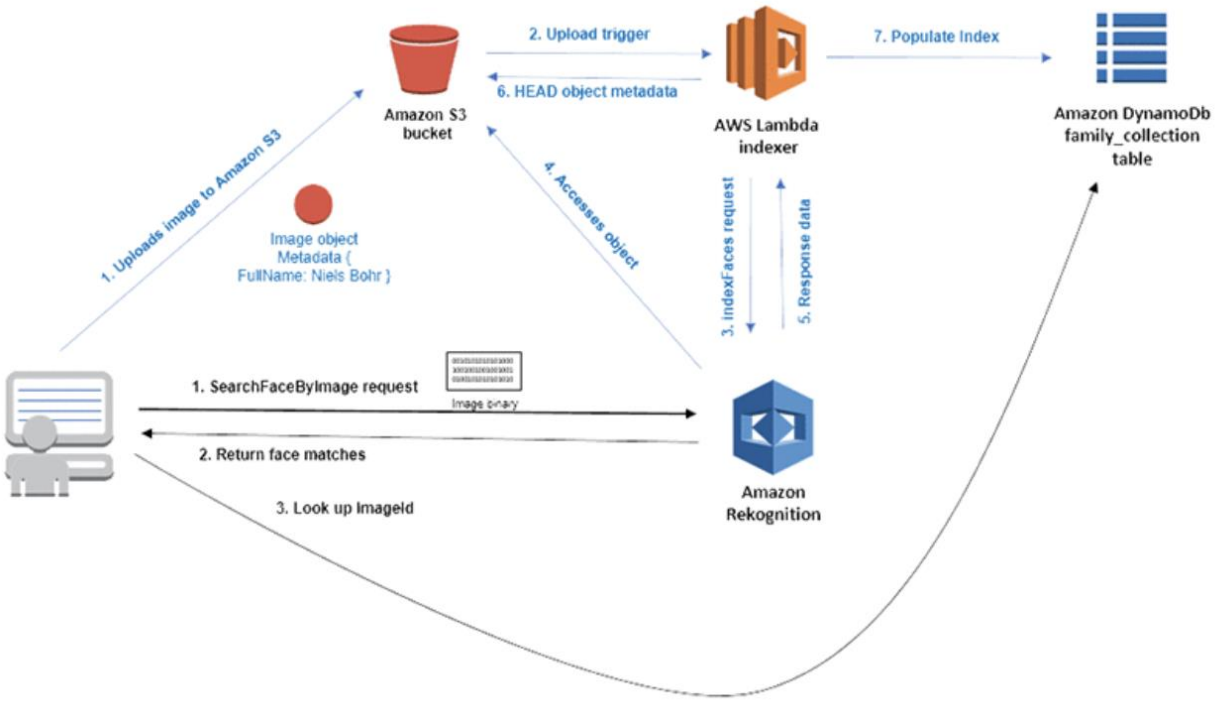

Figure1. Application workflow, Source: AWS website

I. Indexing

1) Upload Image to S3 bucket (MetaData)

After triggering the Lambda Function, we need to upload images to Amazon S3 and seed the face collection. Here we add additional metadata to the objects in Amazon S3. The Lambda function uses this metadata to extract the person's full name within a given image.

I have added multiple references for a single person to the image collection because adding multiple reference images per person greatly enhances the potential match rate for a person. It also provides additional matching logic to further enhance the results.

2) Triggering Lambda Function

For the facial recognition to process, we need to create a Lambda function that is triggered every time we upload a new picture to Amazon S3 bucket. Below are the steps to follow :

i. Go to the configure triggers page
ii. Select S3 then the name of the bucket, here it is (*muralis-face-recognition)*
iii. Configure event type as "All objects Created"
iv. Configure Prefix as "Index/"
v. On the next page, Give the Lambda function a name (*muralis-face-recognition-lambda*)
vi. Choose the Python 2.7
vii. Run the Python code. *(I have created the code given below*)
viii. Choose an existing role in role selection (*end of the page*)
ix. Click Next
x. Choose create Function

This ensures that Lambda function is triggered only when new objects that start with a key matching the **index/** pattern are created within the bucket.

```
import boto3

dynamodb_client = boto3.client('dynamodb')
s3_client = boto3.client('s3')
rekognition_client = boto3.client('rekognition')

def lambda_handler(event, context):
    bucket = event['Records'][0]['s3']['bucket']['name']
    key = event['Records'][0]['s3']['object']['key']

    try:
        # get full name from s3 metadata
        s3_header_object = s3_client.head_object(Bucket=bucket, Key=key)
        full_name = s3_header_object['Metadata']['fullname']

        # Call Amazon rekognition api to get face print
        response = rekognition_client.index_faces(
            Image={"S3Object": {"Bucket": bucket, "Name": key}},
            CollectionId="face_recognition")

        if response['ResponseMetadata']['HTTPStatusCode'] == 200:
            # extract face_print from the response.
            face_print = response['FaceRecords'][0]['Face']['FaceId']

            # update the faceprint and name in dynamodb.
            dynamodb_client.put_item(
                TableName='face_recognition',
                Item={
                    'RekognitionId': {'S': face_print},
                    'FullName': {'S': full_name}
                })
        return response
    except Exception as e:
        raise e
```

To summarize, The code performs 3 main activities:

1. It uses the Amazon Rekognition IndexFaces API to give a faceprint of the input image.
2. If successful, it retrieves the full name of the person from the metadata of the object in Amazon S3.
3. Then it stores this as a key-value tuple with the Faceprint and the person name in the DynamoDB.

II. Analysis

Recognize Face: During the Indexing phase we indexed the faces in the Rekognition collection. In the Analysis phase, we can detect faces from the new image by querying this collection using "SearchFacesByImage" API.

```
import boto3
import io
from PIL import Image

rekognition = boto3.client('rekognition', region_name='us-west-2')
dynamodb = boto3.client('dynamodb', region_name='us-west-2')

image_path = input("Image path: ")
print(f'Reading image: {image_path}')
image = Image.open(image_path)
stream = io.BytesIO()
image.save(stream,format="JPEG")
image_binary = stream.getvalue()

print(f'Calling Amazon recognition to search face')
response = rekognition.search_faces_by_image(
        CollectionId='face_recognition',
        Image={'Bytes':image_binary}
        )

present =  False
for match in response['FaceMatches']:
    face = dynamodb.get_item(
        TableName='face_recognition',
        Key={'RekognitionId': {'S': match['Face']['FaceId']}}
    )
    if 'Item' in face:
        print('Person is: ',face['Item']['FullName']['S'])
        present = True
        break

if not present:
    print("Person not found")
```

To summarize, the code performs 3 main activities:

1. It reads an image from the local disk and calls Amazon Rekognition API (here I have used "search_faces_by_image" API)
2. The "search_faces_by_image" API returns a faceprint
3. We search the faceprint in the DynamoDB to get the person name
4. If it retrieve a person name, we can return the recognized person else the code says we cannot detect face.

CONCLUSION:

In this Project, I have used AWS Rekognition service to build my own facial Recognition Service. I have listed a few pointers and codes by following the documentation given in the Amazon AWS website which has helped me decide on the strategy for using collections, AWS Rekognition, AWS Lambda Function, AWS DynomoDb, IAM, and Amazon S3 bucket. With understanding these tools, I have built my own image-mining solution that detects, indexes, and recognizes faces.

Amazon Rekognition tries to find a match for only the most prominent face (or largest) within an image. As avenues for future research, we can detect faces even if the image contains multiple people, we would first need to use the DetectFaces API to retrieve the individual bounding boxes of the faces within the image.

3 W's

1. What went well?

    - Was able to explore AWS which was a completely new tool for me.
    - The Book reading helped me with the theoretical concepts behind faceprint and how image vectors are formed.
    - AWS has excellent documentation which helped in understanding the concepts.

2. What did not go well?

    - Setting up the infrastructure consumes a lot of time because understanding the technical terms used in AWS required a deep dive into the documentation which was a little confusing.
    - Tried to use Emotions API but this required a paid subscription of AWS

3. What would I do differently next time?

    - Currently I'm only using solo pictures, next time I would use a group picture and use extract API
    - Use secondary match logic: Currently Amazon Recognition gives a confidence score for the images and the result is shown according to the highest confidence score. But using the secondary match logic, we can predict fuzzy images by returning probability of who the person could be.

BOOK REVIEW

"Content-Based Image Classification
Efficient Machine Learning Using Robust Feature Extraction Techniques."

- By Rik Das

The author has explained how Facial recognition is multi-class classification where each person represents a different class. Deep learning algorithms are increasingly being used to picture data in recent years and most content-based image classification tasks can be accurately executed using deep learning techniques. In his book, Rik Das presents the basics of deep learning-based approaches. Primarily, the book covers various techniques of image processing including implementation of image binarization, image transforms, texture analysis, and morphological approach. It illustrates why all of these techniques are important for effective feature extraction from image content.

One of the major take-away from this book is that to explore information present in an image we need to implement diverse feature extraction techniques on low-level image characteristics such as color, texture, and shape. The two most important stages in content-based image classification consist of robust feature extraction from image data followed by the classification of image categories with the help of extracted signatures.

Below are some of the techniques I found very relevant to my project and believe that Amazon uses these techniques to determine the faceprint in the image vectors:

Extraction of Features with Color Contents:

One of the prominent characteristics of an image is the color components for the extraction of meaningful features. One way to extract color is to consider three horizontal regions in the image that do not overlap, and from each of the zones, the first three moments for each separate color channel are extracted to a feature vector of 27 floating points.

Extraction of Features with Image Binarization

Segregation of the required content in the picture from its background for proficient extraction of its characteristics is rightly carried out with the binarization method. The method of image binarization has comprehensively implemented the selection of appropriate threshold to differentiate the region of interest in an image from its background.

Extraction of Features with Morphological Processing

The contour of the objects present in an image can be resourcefully analyzed by mathematical morphology, which is based on set theory. A binary image is considered a set, and the application of set operators, namely, intersection, union, inclusion, and complement, can be well performed on the image.

Overall, I found this book very helpful in understanding the theoretical concepts behind image mining because the author covers a wide range of strategies for implementing content-based image recognition with increased precision. It demonstrates the use of various machine-learning approaches that contribute in the process of content-based image recognition. The book serves as a guide to the appropriate use of Amazon's software and APIs required to pursue the process of content-based image identification with high accuracy and minimal computing overhead.

To conclude, It is important to note that Amazon Rekognition doesn't store copies of the analyzed images. Instead, it stores face feature vectors as the mathematical representation of a face within the collection. This is often referred to as a *thumbprint* or *faceprint*. Amazon Recognition uses Color contents, Image binarization, and Morphological processing feature extraction techniques to generate a feature vector(faceprint). We can use this faceprint to correctly classify the face in the image.

REFERENCES:

1. Amazon Rekognition Developer guide:
https://docs.aws.amazon.com/pdfs/rekognition/latest/dg/rekognition-dg.pdf

2. AWS Website:
https://aws.amazon.com/what-is/facial-recognition/#:~:text=It%20works%20by%20identifying%20and,large%20collection%20of%20existing%20images.

3. S3 bucket command reference:
s3 — AWS CLI 1.29.31 Command Reference (amazon.com)

4. AWS Documentation for creating role:
Creating a role to delegate permissions to an AWS service - AWS Identity and Access Management (amazon.com)

5. Book review:
Das, R. (2020, December 17). *Content-Based Image Classification: Efficient Machine Learning Using Robust Feature Extraction Techniques*. CRC Press.