CLOUD COMPUTING – CS623

A Cloud Based Image Recognition System

Sai Vithal Rithvik Macherla (eq2528)

Chandini Byrapuneni (gt3529)

Sri Sai Anirudh Khandavilli (ai4482)

Problem Statement:

To detect and recognize the image and videos of a celebrity by training the model with labelled dataset.

Description

The project is an online cloud-based image detection, recognition and localization project which uses state of the art algorithms of **Amazon Rekognition** in AWS. It is a custom trained project so it seeks help of the custom database. The databased image then needs labelling and as localization was intended thus the bounding boxes too were drawn around the object of interest (Fig 1 shows the process). The training is then performed remotely and performance parameters are available showing the usability of custom trained model.

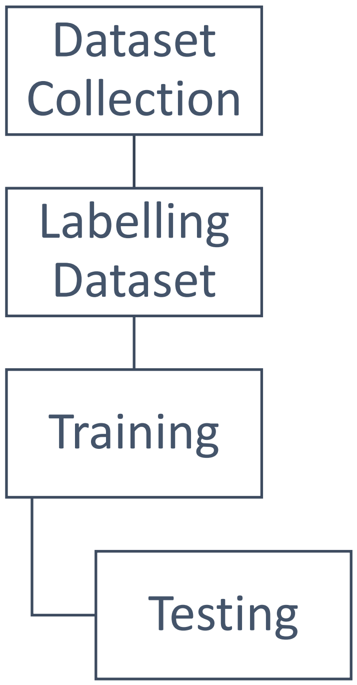

Figure 1 Depicting the process of Model creation on Amazon Recognition

The deployment of model was done remotely and python was used to test the model. Image processing was done on the images retrieved from the cloud server.

The test images are stored online in the Amazon S3 bucket. Our code will access those images from S3 bucket and use the model that we trained earlier to process those images. The result is then received back in form of a .json file and the image itself. Image processing is done to draw the bounding boxes from the data received in the .json file on the image and thus the result is displayed.

Services & Libraries Used

Following are the services used from AWS:

- **Boto3**

  Boto3 is a python SDK developed to connect with amazon AWS and allow the user to modify, create and delete files in the amazon server.

- **S3 Bucket**

  S3 Bucket is a public data storing space hosted by Amazon Web Services. Simple Storage Services (S3) is similar to the file folders which stores object as well as their descriptive metatags. We store the images in these buckets.

- **Amazon Rekogonition** – Custom Labels

  Amazon Rekognition is used to train objects, area of interests and scenes from the images that are tailored to user’s needs. It is an advanced feature that require expertise in data collection, handling and labelling as this feature require the data to be readied and model be trained.

The python libraries used are following:

- **PIL – Python Imagining Library**

  PIL is a python library that adds Image processing features to the python interpreter. It provides extensive file support and is a very powerful library that provides general foundational Image processing capabilities.

Relationship between Services:

1. We upload an image into an S3 bucket.
2. Amazon S3 invokes the first of the two AWS Lambda functions to create a new job in Amazon Elastic Transcoder (the code for this follows this list).
3. Amazons Rekognition is used to create labels for the trained images in the S3 Bucket.

Flow:

Amazon Rekognition Model is running in AWS Server.

The test images are stored in the S3 Bucket which is in the AWS Server.

We start the model from our System.

Amazon Rekognition labels the test images.

We stop the model from the system.

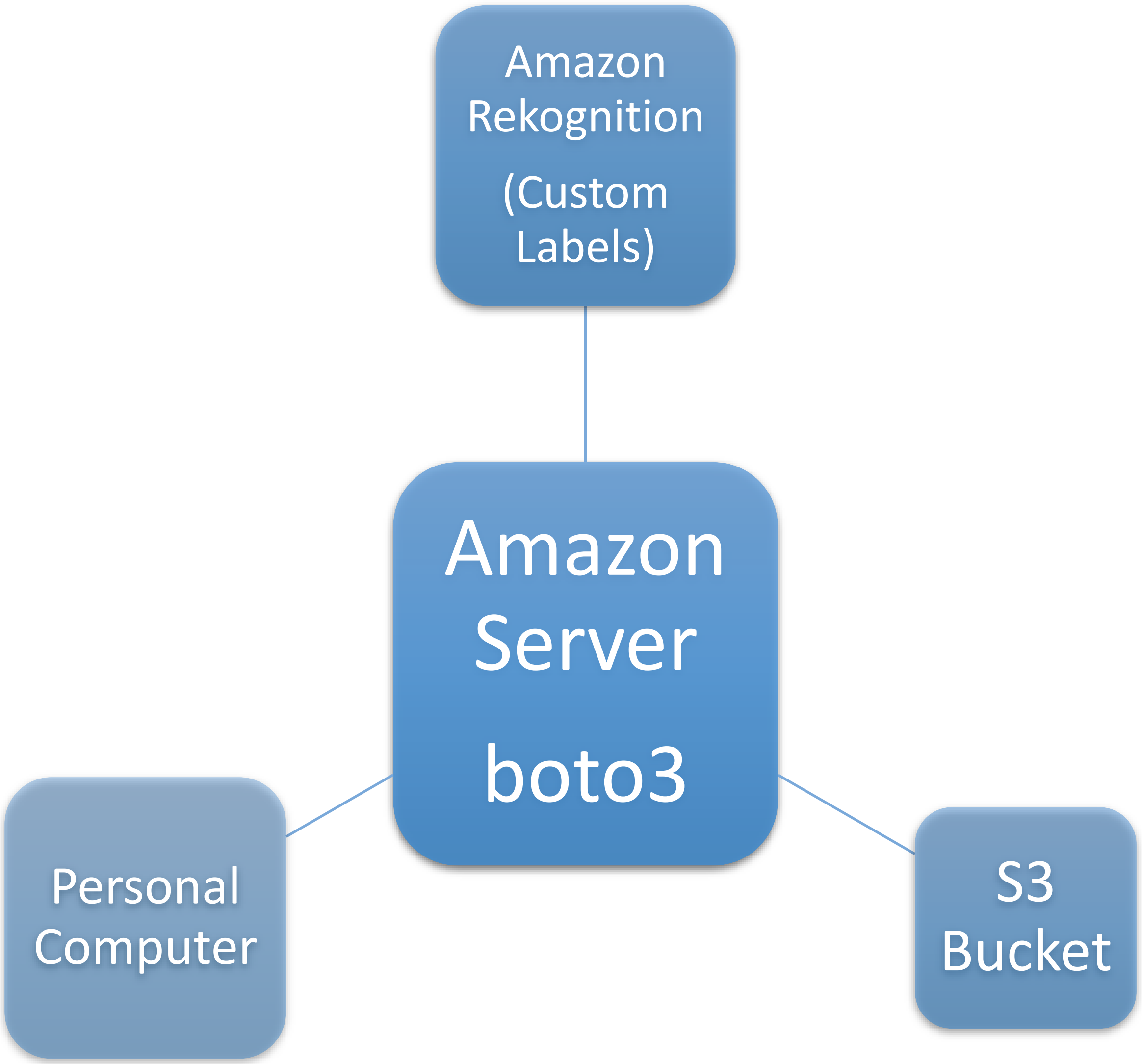

Figure 2 Showing the relationship between the services and User

Output:

The output is in form of a .json file which contains all the information of the image. The json file contains all the details including the no of classes and instances detected. That information is then used to draw bounding boxes on the image and display it as result by using PIL library in python.

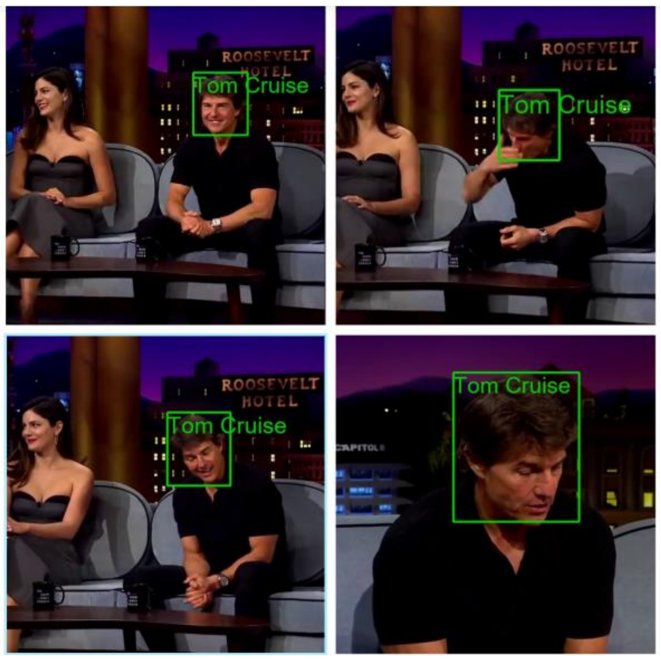

Figure 3 Results from the model trained on cloud

Performance & Functionality

The performance parameters of the model are following:

Table 1 Showcasing the results

Performance Parameter	Value
F1 Score	0.747
Precision	0.70
Recall	0.81

The performance parameters show promising result. The F1 score is 0.74 and generally it is considered good if it is more than 0.7. The project has high precision and recall too. The performance of the model can be further improved by increasing the training data. Generally, the model showed promising results.

Source Code:

Source code consists of three files. The first file will start the model – train the model while the second file will be used to test the image and use image processing to display it after drawing the bounding boxes. After using the model, the third file will be executed that will turn the model off.

Start_model.py:

```
import boto3

def start_model(project_arn, model_arn, version_name, min_inference_units):

    client=boto3.client('rekognition', region_name='ap-northeast-1')

    try:
        # Start the model
        print('Starting model: ' + model_arn)
        response=client.start_project_version(ProjectVersionArn=model_arn,
MinInferenceUnits=min_inference_units)
        # Wait for the model to be in the running state
        project_version_running_waiter =
client.get_waiter('project_version_running')
        project_version_running_waiter.wait(ProjectArn=project_arn,
VersionNames=[version_name])

        #Get the running status
        describe_response=client.describe_project_versions(ProjectArn=project_arn
,
            VersionNames=[version_name])
        for model in describe_response['ProjectVersionDescriptions']:
            print("Status: " + model['Status'])
            print("Message: " + model['StatusMessage'])
    except Exception as e:
        print(e)

    print('Done...')

def main():
    project_arn='arn:aws:rekognition:ap-northeast-
1:591284369655:project/SimpleImagerecognation/1668778312729'
    model_arn='arn:aws:rekognition:ap-northeast-
1:591284369655:project/SimpleImagerecognation/version/SimpleImagerecognation.2022
-11-25T21.47.19/1669394838883'
    min_inference_units=1
    version_name='SimpleImagerecognation.2022-11-25T21.47.19'
    start_model(project_arn, model_arn, version_name, min_inference_units)

if __name__ == "__main__":
    main()
```

test.py:

```python
from typing import Any
import boto3
import io
from PIL import Image, ImageDraw, ExifTags, ImageColor, ImageFont

def display_image(bucket,photo,response):
    # Load image from S3 bucket
    s3_connection = boto3.resource('s3','ap-northeast-1')

    s3_object = s3_connection.Object(bucket,photo)
    s3_response = s3_object.get()

    stream = io.BytesIO(s3_response['Body'].read())
    image=Image.open(stream)

    # Ready image to draw bounding boxes on it.
    imgWidth, imgHeight = image.size
    draw = ImageDraw.Draw(image)

    # calculate and display bounding boxes for each detected custom label
    print('Detected custom labels for ' + photo)
    for customLabel in response['CustomLabels']:
        print('Label ' + str(customLabel['Name']))
        print('Confidence ' + str(customLabel['Confidence']))
        if 'Geometry' in customLabel:
            box = customLabel['Geometry']['BoundingBox']
            left = imgWidth * box['Left']
            top = imgHeight * box['Top']
            width = imgWidth * box['Width']
            height = imgHeight * box['Height']

            fnt = ImageFont.truetype('/Library/Fonts/arial.ttf', 50)
            draw.text((left,top), customLabel['Name'], fill='#00d400', font=fnt)

            print('Left: ' + '{0:.0f}'.format(left))
            print('Top: ' + '{0:.0f}'.format(top))
            print('Label Width: ' + "{0:.0f}".format(width))
            print('Label Height: ' + "{0:.0f}".format(height))

            points = (
                (left,top),
                (left + width, top),
                (left + width, top + height),
                (left , top + height),
                (left, top))
            draw.line(points, fill='#00d400', width=5)

    image.show()

def show_custom_labels(model,bucket,photo, min_confidence):
    client=boto3.client('rekognition', region_name='ap-northeast-1')

    #Call DetectCustomLabels
    response = client.detect_custom_labels(Image={'S3Object': {'Bucket': bucket,
'Name': photo}},
        MinConfidence=min_confidence,
        ProjectVersionArn=model)

    # For object detection use case, uncomment below code to display image.
    # display_image(bucket,photo,response)

    return (len(response['CustomLabels']),response)

def main():

    bucket='osmebucket'
    photo='vlcsnap-2022-11-21-18h10m21s948.png'
    model='arn:aws:rekognition:ap-northeast-
1:591284369655:project/SimpleImagerecognation/version/SimpleImagerecognation.2022
-11-25T21.47.19/1669394838883'
    min_confidence=50

    label_count, res1=show_custom_labels(model,bucket,photo, min_confidence)
    print("Custom labels detected: " + str(label_count))
    print(res1)
    display_image(bucket, photo, res1)


if __name__ == "__main__":
    main()
```

stop_model.py

```
#Copyright 2020 Amazon.com, Inc. or its affiliates. All Rights Reserved.
#PDX-License-Identifier: MIT-0 (For details, see
https://github.com/awsdocs/amazon-rekognition-custom-labels-developer-
guide/blob/master/LICENSE-SAMPLECODE.)

import boto3
import time


def stop_model(model_arn):

    client=boto3.client('rekognition', region_name='ap-northeast-1')

    print('Stopping model:' + model_arn)

    #Stop the model
    try:
        response=client.stop_project_version(ProjectVersionArn=model_arn)
        status=response['Status']
        print ('Status: ' + status)
    except Exception as e:
        print(e)

    print('Done...')

def main():

    model_arn='arn:aws:rekognition:ap-northeast-
1:591284369655:project/SimpleImagerecognation/version/SimpleImagerecognation.2022
-11-25T21.47.19/1669394838883'
    stop_model(model_arn)

if __name__ == "__main__":
    main()
```

Limitations:

- Small training data.
- Performance can be optimized and improved using more training and larger dataset.
- Single class classification- Classifies and recognizes only 1 face.