**MASK IDENTIFICATION**

**INTRODUCTION:**

Safety became an import factor during the pandemic situation and gaining the trust of customers became more difficult.

In regard to the problem faced, an application is to be built which would scan humans to check if they are wearing a mask or not. The methodology which this lab follows is using the AWS Cloud. Using the Rekognition, Lambda, S3 and API Gateway services of the AWS Cloud, this application will be built.

**WORKING:**

You’ll be able to work with custom labels of AWS Rekognition, dynamodb, lambda functions and API Gateway and build a flask application which will detect masks and update the dynamodb.

**If you are using anaconda navigator, follow below steps to download OpenCV, boto3, requests and flask:**

Open anaconda prompt as administrator.  
Type "pip install opencv-python” and Click enter.  
Type "pip install flask” and click enter.  
Type "pip install boto3” and click enter.  
Type "pip install requests” and click enter.

the above steps allow you to install the packages in the anaconda environment

**If you are using any other IDLE, please follow the below steps to install OpenCV, boto3, requests, and flask:**

Open Command Prompt.  
Type "pip install opencv-python” and Click enter.  
Type "pip install flask” and click enter.  
Type "pip install boto3” and click enter.  
Type "pip install requests” and click enter.

the above steps allow you to install OpenCV, boto3, requests and flask.

------------------------------------------------

**Create “Rekognition” Service And Upload The Dataset**

Step 1: After logging in, search for “Rekognition” service either under the “Services” dropdown or in the search bar.

Step 2: Choose “Custom Labels”. Check if the region selected in “N. Virginia” (us-east-1)

Step 3: Give the project name (mask-detect) > click on “Create project”

Step 4: Click on Create dataset

Step 5: Give the dataset name, choose to upload images from the computer and click on submit.

Step 6: To add the images on to dataset, click on “Add”. Images can be added in a batch of 30 images. Upload all the images into the dataset.

**Assign Labels To The Images And Draw The Bounding Boxes.**

Step 1: Select the images in the first page (9 images) > click on "Assign Labels" > choose "mask" label > click on "Assign" > click on "Save Changes"

Step 2: Move to the second page and assign labels to all the images present in the dataset.

Step 3: After all the images are labelled, you'll have (0) in the "Unlabelled" section.

Step 4: Choose the images again and click on "Draw Bounding Box". Draw the bounding box over the masks present in the pictures.

Step 5: After specifying the bounding boxes for all the batch of images > click on "Done" > "Save changes"

Step 6: Repeat the process for all the images present in the dataset.

Note: If you have exited the labelling mode, you can enter it by click on "Start Labelling"

**Train The Model**

Step 1: Verify the dataset (should be labelled and bounded by a box). After review click on “Train model”.

Step 2: Choose the training dataset > choose the existing dataset for testing > specify the test set > click on “Train”

You can find the model under training. The training time will be nearly an hour. After training is done your model will be ready to use.

**Credentials**

Step 1: Navigate back to Vocareum labs page > click on Account Details

Step 2: When you click on show button, you can find the credentials. Copy the credentials.

Step 3: Navigate to .aws folder in your PC, which you’ll be getting after the CLI installation. Usually the folder is located in C:/users/username/.aws . Open the “credentials” file using notepad. Replace the data present with the credentials copied from the Vocareum page.

**Start The Model**

Step 1: After the model is trained, in model details, under API Code copy the command specified under start model.

Step 2: Paste the code in the command prompt by removing the 3 slashes in between

The model will start after the command is executed.

**Create A Table In DynamoDB**

Step 1: Under services in AWS Console, search for “DynamoDB” and open it > click on “Create table”.

Step 2: Give the table name (count-without-masks) and the primary/partition key as date. We will be storing the count of violations each day. After clicking on create the table will be created.

**Create A Lambda Function**

Step 1: Under services in AWS Console, search for “lambda” and open it > click on “Create function”.

Step 2: Choose “Author from Scratch” > name the function > choose “Python 3.6” > click on create function

Create a new role with basic lambda permissions

Step 3: Scroll down and write the below code in the function.



**Attach The Policies**

Attach the policies to the role created: To access the DynamoDB using lambda function the permissions are to be given. To attach the policy open IAM from services.

Step 1: Choose roles

Step 2: Choose the role which was created while creating the lambda function. (The role name will be similar to the lambda function name)

Step 3: Click on Attach policies > search for “dynamodb” > choose the fullaccess policy > click on Attach policy

**Test The Lambda Function**

Step 1: Open the lambda function created > click on Test (you can find it on the top right corner or in the middle of the page).

Step 2: Give the event name / test case name. Give the sample input > click on create

Step 3: After the test event is saved, click on test to test the Lambda function

The result should be “Successful insertion” if the function is executed and the data is pushed to the database.

Step 4: Open the database to check if the data is inserted.

**Create An API For The Lambda Function**

Step 1: Open “API Gateway” from the services from AWS Console.

Step 2: Click on “Create API” > choose “REST API” > click on Build under Rest API

Step 3: Choose the REST Protocol > choose “New API” > give a name to it (maskcount) > the endpoint type should be “Regional” > click on Create API

Step 4: After the API is created, we need to create a resource under which a method will be created. Click on Actions dropdown and choose “Create Resource”.

Step 5: Give the resource name (maskcount) > click on “Create Resource”

Step 6: Under the resource create a method by clicking the actions dropdown > choose GET method > tick the tick symbol

Step 7: We shall connect the lambda function that is created earlier to the method. Under the Lambda Function input field choose the lambda function created > change the default timeout to 1000ms > click on save.

Step 8: For GET method the query parameters are to be set to take the specified inputs through URL. Click on Method Request (first box)

Step 10: The URL need to accept the data and send the data in json format. To change the response format, click on Integration request (second box)

Step 11: Scroll down to find the “Mapping templates” > in “Content-type” type application/json and click on tick to get the text box > type the below code > click on Save

Code: {

“date”:”$input.params(‘date’)”,

“count”:”$input.params(‘count’)”  
}

Step 12: Go back and click on Test

Step 13: Give the inputs in {maskcount} to the method to get the output in the format date=2020-08-30&count=30 > click on Test > the output will be displayed towards right.

**Deploy The API**

Step 1: Under action choose “Deploy API”

Step 2: Under Deployment stage choose [New Stage], name the stage (countmask) > click on deploy. You’ll be redirected to stages.

Step 3: When you choose the method under resource you can find the API on screen, which we will be using in future.

**Flask Application**

app.py has the main flask code, which on execution, the application will run.   
camera.py has the code which will capture the frame, pass it the rekognition model, operates on count and updates the database by hitting the API.   
countfile.txt stores the count value locally.  
xml file locates a face on screen.   
templates folder will the html page.

Step 1: Replace the user credentials. Open camera.py file using any editor > replace the existing credentials with your credentials.

Step 2: Replace the ARN (Amazon Resource Name) > copy the ARN which you can find in the model description of your rekognition model.

Step 3: Replace it with the existing ARN in the camera.py file.

Step 4: Replace the API which was created for the lambda function. You can get the API from API Gateway > click on API name > choose Stages > click on “Get” under the resource > copy the URL.

Step 5: Replace the existing URL with your copied URL (only till ?) in camera.py

Step 6: Save the file

**Run The Application**

Step 1: Open anaconda navigator if you are using anaconda environment. Open command prompt if you are using other environments like IDLE, pycharm.

Step 2: Navigate to the folder in which you have the file (set the path in prompt)

Step 3: Execute main.py using python command

Command: python main.py

Step 4: While file is running, open the browser and launch localhost:5000 to view the application.