|  |
| --- |
|  |
| import datetime |
|  | import requests |
|  | import cv2 |
|  | import ibm\_boto3 |
|  | from ibm\_botocore.client import Config, ClientError |
|  | import json |
|  | import ibmiotf.application |
|  | import ibmiotf.device |
|  | import random |
|  | import time |
|  |  |
|  | from cloudant.client import Cloudant |
|  | from cloudant.error import CloudantException |
|  | from cloudant.result import Result, ResultByKey |
|  | #Provide your IBM Watson Device Credentials |
|  | organization = "7x37hm" |
|  | deviceType = "facedetect" |
|  | deviceId = "1001" |
|  | authMethod = "token" |
|  | authToken = "1234567890" |
|  |  |
|  |  |
|  | url = "https://face-mask-detection.p.rapidapi.com/FaceMaskDetection" |
|  |  |
|  |  |
|  | # Constants for IBM COS values |
|  | COS\_ENDPOINT = "https://s3.jp-tok.cloud-object-storage.appdomain.cloud" # Current list avaiable at https://control.cloud-object-storage.cloud.ibm.com/v2/endpoints |
|  | COS\_API\_KEY\_ID = "QgMp-B0GKTnFu3FkdmGeo681oR1UA-RUdTidTkisG5a-" # eg "W00YixxxxxxxxxxMB-odB-2ySfTrFBIQQWanc--P3byk" |
|  | COS\_AUTH\_ENDPOINT = "https://iam.cloud.ibm.com/identity/token" |
|  | COS\_INSTANCE\_CRN = "crn:v1:bluemix:public:cloud-object-storage:global:a/cfdf24503963447ba9432271b7cbd1df:eec1510a-7347-40be-bdff-746117c6639e::" |
|  |  |
|  | client = Cloudant("apikey-v2-1t8pu8xkov65lsjznpkb1mbe1f0mb3v0g1slyih3sd7d", "bdff9d8bf839712fd47c65cb06ce00e1", url="https://apikey-v2-1t8pu8xkov65lsjznpkb1mbe1f0mb3v0g1slyih3sd7d:bdff9d8bf839712fd47c65cb06ce00e1@bd453ab0-a5bd-4358-b64d-83ca7948d553-bluemix.cloudantnosqldb.appdomain.cloud") |
|  | client.connect() |
|  | database\_name = "facedetect" |
|  |  |
|  |  |
|  | # Create resource |
|  | cos = ibm\_boto3.resource("s3", |
|  | ibm\_api\_key\_id=COS\_API\_KEY\_ID, |
|  | ibm\_service\_instance\_id=COS\_INSTANCE\_CRN, |
|  | config=Config(signature\_version="oauth"), |
|  | endpoint\_url=COS\_ENDPOINT |
|  | ) |
|  |  |
|  | bucket = "mdetect" |
|  | def multi\_part\_upload(bucket\_name, item\_name, file\_path): |
|  | try: |
|  | print("Starting file transfer for {0} to bucket: {1}\n".format(item\_name, bucket\_name)) |
|  | # set 5 MB chunks |
|  | part\_size = 1024 \* 1024 \* 5 |
|  |  |
|  | # set threadhold to 15 MB |
|  | file\_threshold = 1024 \* 1024 \* 15 |
|  |  |
|  | # set the transfer threshold and chunk size |
|  | transfer\_config = ibm\_boto3.s3.transfer.TransferConfig( |
|  | multipart\_threshold=file\_threshold, |
|  | multipart\_chunksize=part\_size |
|  | ) |
|  |  |
|  | # the upload\_fileobj method will automatically execute a multi-part upload |
|  | # in 5 MB chunks for all files over 15 MB |
|  | with open(file\_path, "rb") as file\_data: |
|  | cos.Object(bucket\_name, item\_name).upload\_fileobj( |
|  | Fileobj=file\_data, |
|  | Config=transfer\_config |
|  | ) |
|  |  |
|  | print("Transfer for {0} Complete!\n".format(item\_name)) |
|  | except ClientError as be: |
|  | print("CLIENT ERROR: {0}\n".format(be)) |
|  | except Exception as e: |
|  | print("Unable to complete multi-part upload: {0}".format(e)) |
|  |  |
|  | video\_capture = cv2.VideoCapture(0) |
|  | while True: |
|  | # Grab a single frame of video |
|  | ret, frame = video\_capture.read() |
|  |  |
|  | # Display the resulting image |
|  | cv2.imshow('Live', frame) |
|  | picname=datetime.datetime.now().strftime("%y-%m-%d-%H-%M") |
|  | picname=picname+".jpg" |
|  | pic=datetime.datetime.now().strftime("%y-%m-%d-%H-%M") |
|  | cv2.imwrite(picname,frame) |
|  | my\_database = client.create\_database(database\_name) |
|  | multi\_part\_upload(bucket, picname,pic+".jpg") |
|  | payload="linkfile=https://mdetect.s3.jp-tok.cloud-object-storage.appdomain.cloud/"+picname |
|  | if my\_database.exists(): |
|  | print("'{database\_name}' successfully created.") |
|  | json\_document = { |
|  | "link":"https://mdetect.s3.jp-tok.cloud-object-storage.appdomain.cloud/"+picname |
|  | } |
|  | time.sleep(1) |
|  | #print data |
|  | def myOnPublishCallback(): |
|  | print ("Published data to IBM Watson") |
|  | headers = { |
|  | 'content-type': "application/x-www-form-urlencoded", |
|  | 'x-rapidapi-key': "92bc4dfae7mshf453994a97ab7c0p168e9fjsn8fe751cd72a8", |
|  | 'x-rapidapi-host': "face-mask-detection.p.rapidapi.com" |
|  | } |
|  |  |
|  | response = requests.request("POST", url, data=payload, headers=headers) |
|  | print(response.text) |
|  |  |
|  | a=json.loads(response.text) |
|  | x0=a["data"][0]["x0"] |
|  | y0=a["data"][0]["y0"] |
|  | x1=a["data"][0]["x1"] |
|  | y1=a["data"][0]["y1"] |
|  | print(type(a)) |
|  | if(a["data"][0]["masked"]==0): |
|  | print("face detected with no mask") |
|  | img=cv2.putText(frame,'No Mask', (x0,(y0-10)), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2) |
|  | img=cv2.rectangle(img,(x0,y0),(x1,y1), (0, 0, 255), 2) |
|  | elif(a["data"][0]["masked"]==1): |
|  | print("face detected with mask") |
|  | img=cv2.putText(frame,'Mask', (x0,(y0-10)), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 0, 255), 2) |
|  | img=cv2.rectangle(img,(x0,y0),(x1,y1), (0, 0, 255), 2) |
|  |  |
|  | cv2.imshow('Processed',img) |
|  | # Hit 'q' on the keyboard to quit! |
|  | if cv2.waitKey(1) & 0xFF == ord('q'): |
|  | break |
|  |  |
|  | # Release handle to the webcam |
|  | video\_capture.release() |
|  | cv2.destroyAllWindows() |
|  |  |