

EC500 J1 Final Project Report

By

Muhammad Zuhayr Raghib

1) Introduction

Skin cancer is the most commonly diagnosed cancer in the United States. Over 100,000 of these cases involve melanoma, the deadliest form of skin cancer. Fortunately, if diagnosed in its earliest stages, there are treatments available that can produce survival rates of over 98%.

One common method of diagnosing skin cancer in general is to examine skin lesions using an imaging technique called Dermatoscopy (or Dermoscopy). A skin lesion is a part of the skin that has an abnormal growth or appearance compared to the skin around it.

This project seeks to deliver a system that can take input photos of skin lesions using a raspberry pi camera unit, transfer them to an online database, and use IBM Watson Cloud Computing services to distinguish the photos from being benign or malignant (cancerous).

2) System Overview - Planned

The system was originally planned to comprise of 2 hardware units:

- 1) A **Raspberry pi unit**, that would take photos using a camera and send the images to an S3 bucket via Amazon IoT MQTT messages
- 2) A **laptop unit** running Ubuntu 14.04 to download the images from an S3 bucket and classify if by calling the IBM Watson Visual Recognition services using Golang and use a Google search API to print information about the form of cancer that was detected.

The system described above would have required a dataset of a number of types of skin cancer. The dataset that was used, however, was focused on a single form of cancer, melanoma. Furthermore, while a script to send MQTT messages from the Raspberry pi unit to an AWS S3 bucket was prepared, it was clear that with a limited payload message size, the image would have to be encoded to base 64, and split to multiple MQTT messages, and downloaded in the laptop unit, in the same order, and decoded from base 64 to reconstruct the image correctly which did not seem as an ideal implementation.

3) System Overview - Implemented

During the progress meeting for the project, it was decided that the system would be redesigned, considering the dataset and the message types. By that time, as the Watson classifier was already prepared using the Python SDK for fast prototyping, it was decided to add a lambda function to the project, rather than converting the code to Golang. This 'Phase 2' system (fig 1) design of the project comprises of a **single Raspberry Pi hardware unit** and performs the following functions:

- 1) Python script on Raspberry Pi takes an input image using a camera connected via USB and uploads the image to an AWS S3 bucket.
- 2) The uploaded S3 image object triggers a Lambda Function that downloads the image to a temporary file and calls IBM Watson Cloud Services APIs to classify the image using a custom classifier.
- 3) The classifier returns the likelihood score for the image being classified as Melanoma, from 0 - 1, which is sent via an MQTT message back to the pi, where it is printed on a terminal.

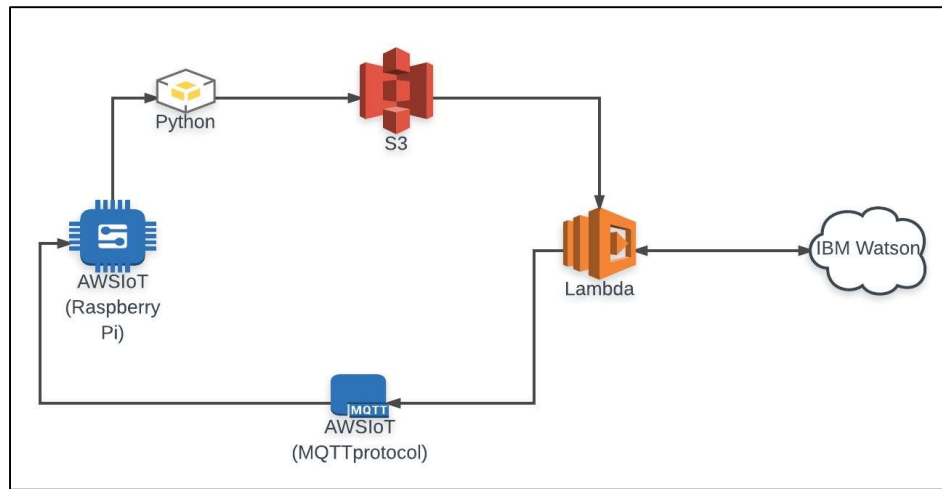


Fig 1 – Phase 2 System Diagram

4) Implemented System Design Approach

Creating a Custom Classifier

I began the design process by creating a custom image classifier using IBM Watson’s Visual Recognition Service APIs. This was done by first downloading a dataset of labelled skin lesion images. The dataset the was used for the project was taken from the ISIC 2017 website [1]. This included positive and negative Dermoscopy images of melanoma.

Using the IBM Watson services [2] Python SDK, I was then able to train a custom classifier (see `Create_custom_classifier.py` on GitHub page)

Uploading images from Raspberry Pi to AWS S3

Using the AWS Python SDK ‘Boto3’ on the Raspberry Pi, I wrote a python script (see `cam_to_S3.py` on Github) to take a photo using a USB webcam, and upload it to an S3 bucket with a specified name and ‘Key’ (object name for the photo inside the bucket).

To achieve the functionality of the code, the Python SDK had to first be configured by setting up IAM credentials for authentication. The access and security access keys for an AWS IAM user were downloaded and saved to default directories instead of being hard coded inside the code. The IAM user used in this case had administrator permissions.

Also, the S3 bucket used was given read and write permissions for ‘Everybody’, therefore making the bucket available to anyone.

Lambda Function

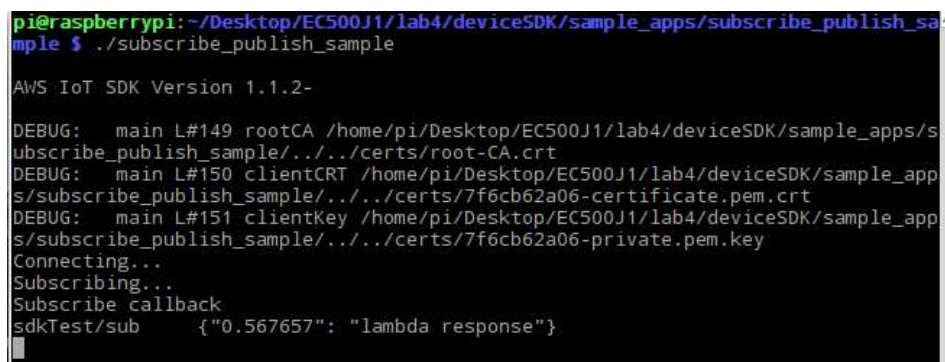
I began creating the lambda function by selecting AWS S3 as a trigger in the setup page on the AWS website. This automatically set the permissions of the lambda function to receive S3 objects and provided a base python handler function to work on. I also increased the timeout from 3 seconds to 10 seconds, to accommodate the time it would take the handler function to perform all its functions. To save time, I made a new role and gave it administrator permissions and kept all regions as the default ‘east - 1’.

After developing the handler function (see `Lambda_fn.py` on Github), I prepared a .zip file that contained the handler code as well as the IBM Watson SDK libraries which were installed using pip. This file was uploaded as the code of the lambda function.

When an image is uploaded on the specified s3 bucket, the triggered handler function is called. This first downloads the image onto a temporary file location, then classifies the images by calling the custom classifier developed at the start of the project. The API key for authentication with the Watson Cloud Services and custom classifier I.D. are hard coded in the handler function for fast prototyping. The custom classifiers return a result in JSON format containing information including the score for the specified class 'melanoma' which is extracted to a temporary variable. Using Boto3, the score is then published on the 'sdkTest/sub' topic via MQTT.

Receiving Score on the Pi

'subscribe_publish_sample.c' is a sample program that is provided while installing the embedded C SDK for AWS. This program was used to subscribe to the 'sdkTest/sub' topic and print the score on a terminal on the Raspberry Pi (Fig 2). The embedded C SDK was used instead of Boto3, as this functionality was prepared from an earlier lab (lab4). To use the sample program, the same MQTT client ID and shadow name were used from lab4, however, a new X.509 certificate and associated keys were used for the project as the one used for lab 4 seemed to have had expired.



```
pi@raspberrypi:~/Desktop/EC500J1/lab4/deviceSDK/sample_apps/subscribe_publish_sample
$ ./subscribe_publish_sample

AWS IoT SDK Version 1.1.2-

DEBUG:  main L#149 rootCA /home/pi/Desktop/EC500J1/lab4/deviceSDK/sample_apps/s
ubscribe_publish_sample/../../certs/root-CA.crt
DEBUG:  main L#150 clientCRT /home/pi/Desktop/EC500J1/lab4/deviceSDK/sample_app
s/subscribe_publish_sample/../../certs/7f6cb62a06-certificate.pem.crt
DEBUG:  main L#151 clientKey /home/pi/Desktop/EC500J1/lab4/deviceSDK/sample_app
s/subscribe_publish_sample/../../certs/7f6cb62a06-private.pem.key
Connecting...
Subscribing...
Subscribe callback
sdkTest/sub {"0.567657": "lambda response"}
```

Fig 2 – Response from Lambda Function

The score "0.567657" is received along with a standard message

5) Lessons Learned

Difficulties in Training a Custom Classifier

The score received from the classifier is a number from 0 to 1 that is a measure of confidence that the image in question can be classified as melanoma. The particular result in Fig 2 was obtained when the camera was pointing down at the floor. It can be seen from Fig 2, that with a score of 0.567657, this was given an unusually large score. In fact, when an image of a lesion that was positive for melanoma was uploaded, scores never reached beyond 0.2. This was because of a few reasons mentioned below.

The IBM Watson Visual Recognition Service has a few limitations as far as training is concerned. For example, there is a limit of 250 MB of data that can be used to train a classifier. This limits the number of images that can be used for training. After compressing the images by 80% using a jpeg compressor [2], approximately 400 images could be uploaded for training. For this reason, I selected 200 positive and 200 negative images of melanoma for the dataset.

In addition, the dataset used for the project was not ideal as each image was not standardized in how it was taken or prepared [1]. Using this dataset would require complex image processing techniques that was not in the scope of this project. To display the low scores, I set the threshold parameter when calling the custom classifier in the handler function to 0.

6) References

1. https://challenge.kitware.com/#challenge/n/ISIC_2017%3ASkin_Lesion_Analysis_Towards_Melanoma_Detection
2. <https://jpeg-compressor.en.softonic.com/>
3. <https://github.com/watson-developer-cloud/python-sdk>
4. <https://github.com/boto/boto3>