CSE 546 — Project2 Report

Sai Charan Papani - 1222323895 - spapani@asu.edu
Jayanth Kumar Tumuluri - 1221727325 - jtumulur@asu.edu
Lakshmi Venu Raghu Ram Chowdary Makkapati - 1222296439 - lmakkapa@asu.edu
Sarath Chandra Mahamkali - 1222281736 - smahamka@asu.edu

1. Problem statement

The main aim of the project is to create a distributed application to conduct real time face recognition on videos captured by edge devices in real time. We have utilized Raspberry Pi for the IoT part which includes a python script to record the videos and AWS Lambda for the Platform as a Service part which includes the scripts to extract the frames from the raspberry pi, perform face recognition recognition and send the results from DynamoDB to Raspberry Pi. We have also used the Amazon Elastic container registry to store the container image. Lambda is the most extensively used Function as a service iteration of the PaaS. And also Raspberry Pi is by far the most famous IoT development platform.

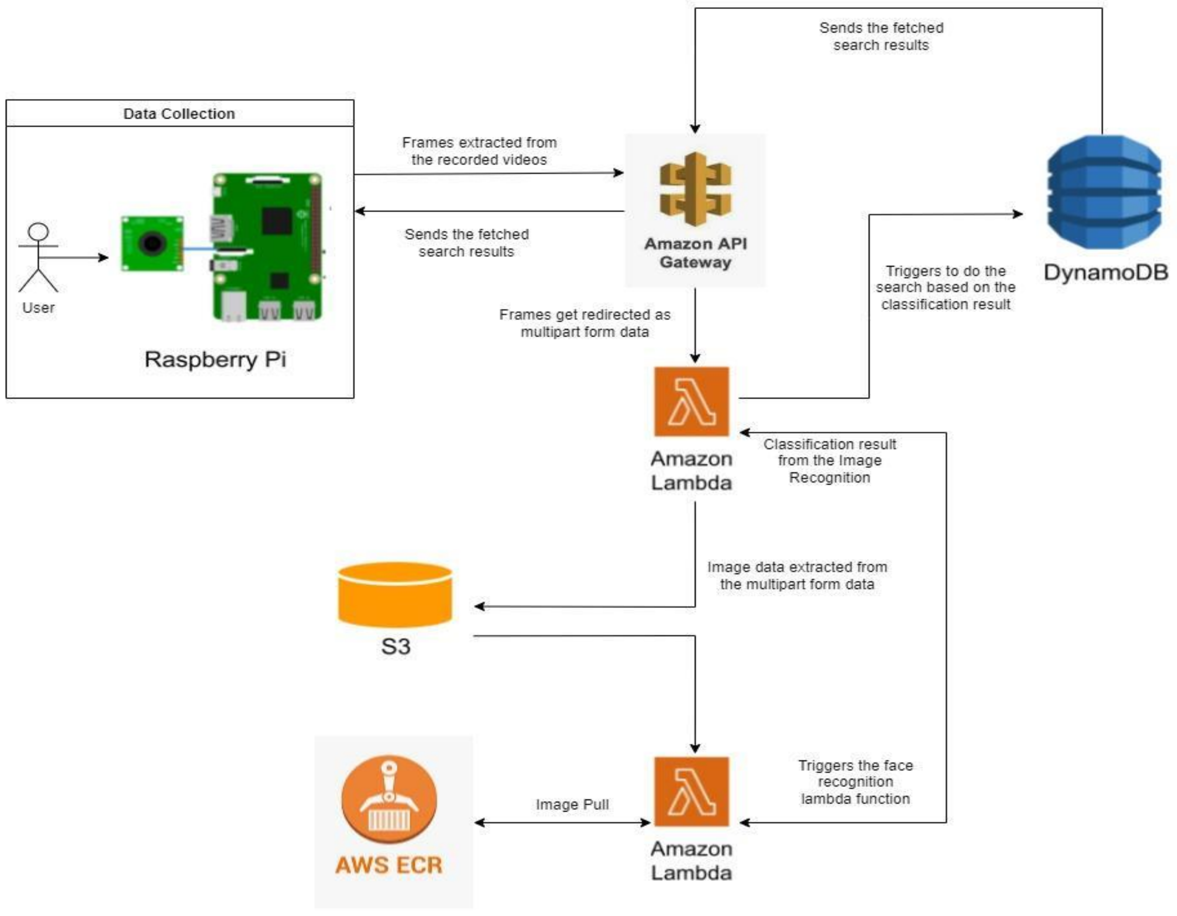
2. Design and implementation

2.1 Architecture

A video is recorded in the raspberry pi continuously for around 5 minutes. A multi threaded python script inside the raspberry pi is written where these threads performs the following tasks:

- Uploads the frames from the video to the input S3 bucket and
- Continuously records the video for 5 minutes.

The frames from the recorded video are uploaded to the S3 bucket via the API Gateway. In the AWS Lambda we created two functions one to perform the face recognition task on the container image and the other to get the results from DynamoDB and send those results to the raspberry pi. The upload_videos lambda is triggered by the API Gateway whenever it receives the frames from the raspberry pi. This lambda function will extract the data received from the API Gateway as a multipart form data and upload this data to the S3 so that the frames are backed up. A container image is created in the ECR and fetched by the Face_recognition Lambda function for the face recognition task. The upload_videos lambda function will invoke the face recognition lambda function and send the input frames from the S3 bucket. Once the recognition is done this function invokes the DynamoDB to search the details of the person which the face recognition function recognizes and then sends the dynamo search results back to the raspberry pi via the API Gateway.



Sends the fetched search results
Data Collection
Frames extracted from the recorded videos
Sends the fetched search results
User
Raspberry Pi
Amazon API Gateway
DynamoDB
Triggers to do the search based on the classification result
Frames get redirected as multipart form data
Amazon Lambda
Classification result from the Image Recognition
Image data extracted from the multipart form data
S3
Image Pull
AWS ECR
Amazon Lambda
Triggers the face recognition lambda function

2.2 Concurrency and Latency

Concurrency:
Concurrency is achieved using multi-threading. Refer App Tier of the code section for more info.

Latency:
Latency is reduced by performing the frame extract process inside the raspberry pi instead of doing it in a separate lambda function.

3. Testing and evaluation

We have tested our code from point to point and also tested it extensively after the final integration. We have checked whether the videos are being sent to the S3 from the Pi using lambda. We have also tested to make sure that the facial recognition lambda functions are being created correctly. According to our methodology, frames will be uploaded to S3 every 0.5 seconds, so we have also checked whether this functionality is working as intended and storing the frames in S3 from the Pi. We have also made sure that the information is being stored correctly in DynamoDB. Before using the model, we trained it with the images of the team mates and achieved a validation accuracy of 89 percent. We have then tested the application with the input video being the one with our faces and tested the accuracy. We have also fine tuned the neural network model by using various optimizers and different batch sizes. At last, we also tested and made sure that the end-to-end latency is not high.

4. Code

This could be considered as multi-tiered architecture. And it has the following abstractions.

- **App tier:** The raspberry pi python program which is running on the raspberry pi operating system.
- **Web tier:** This is the AWS itself and it includes and utilizes various services like API Gateway, S3, Lambdas and DynamoDB.

App Tier:

The python script inside the Raspberry Pi performs the following functionalities:

It continuously records video for the given number of seconds (5*60) which is 5 minutes according to the problem statement and for each 0.5 second duration a video is generated and stored in a specific location. Each video is now passed on to a method (getFaceRecognitionResult) which is run as a thread.

Example: We will get a total of 5*60*2 videos from the 5 minutes and each 0.5 second video is passed on to a thread. So the total number of threads that gets generated and executed are 5*60*2. These are executed independently (multi-threading).

getFaceRecognitionResult perform the following operations:

1. Extract frames from the given video after converting it into mp4 using MP4box library.
2. Upload the video into the S3 bucket.
3. Send the image for a face recognition task. Image will also be uploaded into S3 via lambda function before performing face recognition.

Web Tier:

As said previously, It utilizes some AWS services to cater the scalable and low latent face recognition service.

We utilized the API Gateway to expose an API to App Tier and serve it. And this API internally calls the UploadVideos lambda function. This lambda function performs the necessary operations and returns the response to the Raspberry Pi.

Lambda functions:

UploadVideos functionalities:

1. Extract image file and fileName from the multipart form data.
2. Put the image file into S3 bucket. And calls the lambda Face_Recognition by passing the file key name in the S3 bucket as a parameter.
3. Face_Recognition performs a face recognition task and returns the result back say *fr*.
4. Now the response *fr* is passed into another lambda function getResultsFromDynamoDB as parameter.
5. getResultsFromDynamoDB returns the complete student information by performing the scan operation using the fr (which is the student name).
6. Result from the getResultsFromDynamoDB is returned back to the Raspberry Pi as response.

Face_Recognition functionalities:

1. A docker image is tagged to the lambda from the ECR repository to perform face recognition.
2. This image contains the code snippets and the trained face recognition model.
3. Perform a face recognition task and return the result.

getResultsFromDynamoDB functionalities:

1. It gets the student name as an input parameter.
2. Do a scan operation using the student name and get the complete student information and return it.

Steps to run the code:

- ➔ Connect to the raspberry pi using any of the connection methods (SSH/ HDMI/ Ethernet cable).
- ➔ Run the convertVid.py script inside the raspberry pi.
- ➔ Once the script execution is completed the results are seen in the console.

5. Individual contributions

Saicharan Papani (1222323895):

➔ **Design:**

- The whole design and architecture were progressively made based on all the team members' ideas. I have majorly contributed to the raspberry PI video recording and frame extraction, implementation of the code snippets which fit properly in multi-threading code, implementation of various lambda functions and API gateway setup, etc.

➔ **Implementation:**

- Complete face recognition service is developed using raspberry pi IoT device and Paas service of AWS. I have designed and implemented the pi camera video stream capturing and frame extraction in the PI and handled the multi-threaded code in a way that the thread extracts frames from video after converting the input video into mp4 using the MP4Box library. After extracting the frames from the video, the video gets uploaded to s3, and extracted image frames are sent to the lambda function using API. I have configured the API gateway in such a way that it takes fileName and file as multipart/form-data and inputs the requestBody to lambda functions for further processing.

- Apart from this, I have worked on a couple of lambda functions that deal with face recognition and dynamo DB search. I have created the complete docker image with the trained face recognition model and a couple of other python code snippets and deployed it into the lambda function. The handler function in the docker image is used as a lambda function and has functionalities such as getting an image from the S3 bucket based on the input key, performing face recognition using the trained model, etc. I have also worked on the dynamo DB scan lambda function which extracts complete student information based on the given student name. These lambda functions will be called by the main lambda function which is tagged to the API gateway.

- I have worked together with my teammates on integrating the various lambda functions, API gateway, and raspberry PI client code. Furthermore, worked on reducing the latency or the response time of the API, multi-threading implementation, etc.

➔ **Testing:**

- I have tested the code in various ways like unit testing the components, integration testing (on the complete application), latency testing, performance testing, etc. Apart from this, tested the Pi camera code with multiple scenarios like changing the record durations, and estimating the number of threads that get created (Analysis of how the threads are being worked and evaluating the correction on the execution of the threads), etc.

Jayanth Kumar Tumuluri (1221727325):

➔ **Design:**

- As discussed with the team members, according to the project architecture, I’ve implemented the following:
  I’ve written a multithreaded python function which records the video from the pi camera and uploads the frames extracted from the video to the Amazon S3 bucket via the Amazon API Gateway in the App Tier part. I’ve also written a lambda function which handles the extraction of data, uploading the data to S3, and invocation of other lambda functions in the Web Tier Part.

➔ **Implementation:**

- The multi threaded python script which is present inside the raspberry pi records video for the given duration which in our case is 5 minutes according to the problem statement and for each 0.5 second duration a video is generated and stored in a specific location. Each video is now passed on to a method (getFaceRecognitionResult) which runs as a thread and this thread also uploads the frames to the S3 bucket. For example, we will get a total of 5*60*2 videos from the 5 minutes duration and each 0.5 second video is passed on to a thread. So the total number of threads that are generated and executed are 5*60*2. These are executed independently (multi-threading). I’ve implemented a part of getFaceRecognitionResult which is the uploadImages function. This function implements the API call via the Amazon API gateway to the Lambda function.

- The lambda function extracts the received multipart form data and puts the frames into the S3 Bucket. Also, this function invokes the other lambda functions which performs the face recognition task and once the result is received, the result is sent to the DynamoDB to search the details of the person. Once the results are received, they are sent back to the raspberry pi via the API gateway.

- Furthermore, I’ve also worked with my teammates to improve the accuracy of the given face recognition model by fine tuning and also contributed to improve the latency of the API.

➔ **Testing:**

- For testing the application, in the beginning I checked whether the extracted frames were hitting the API correctly with the Postman app and checked for a response code while debugging. In order to improve the accuracy of the face recognition model I've fine tuned the model by training the model with multiple types of frames which have multiple poses, different lightings and different distance of the subject from the pi camera. I’ve contributed in testing the latency, changed the duration of the video and checked whether the dynamo search is resulting in the correct results according to the classification result and whether these results are properly returned to the raspberry pi.