CSE 546 — Project 2 Report

Shubham Chawla - 1227415724

Anurag Banerjee - 1225497506

Vedant Munjaji Pople - 1225453164

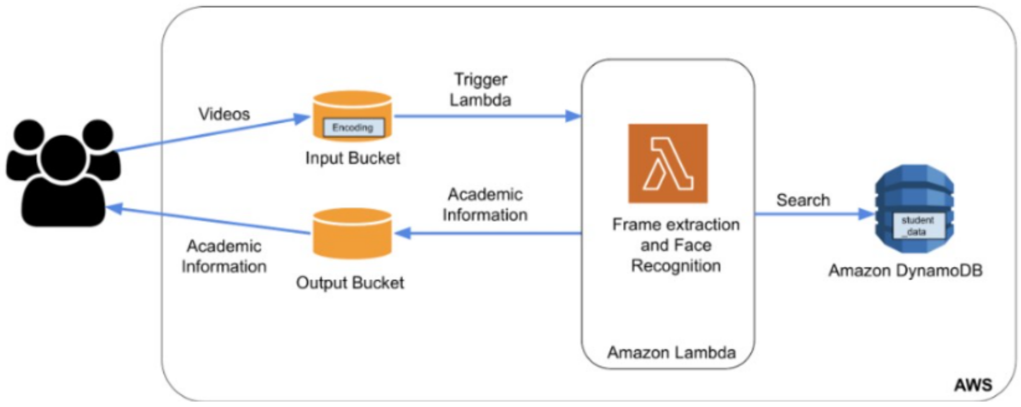
1. Problem statement

Our project implements a smart classroom assistant for educators. This app takes in videos from the user's classroom and performs face recognition on them. The results of face recognition are looked up in the database. These results are used for returning the relevant academic information of each student back to the user.

2. Design and implementation

2.1 Architecture

The application takes videos from the user in the input S3 bucket. Adding a video in the bucket triggers the Lambda function. The Lambda function breaks the video into frames using the Python face_recognition library to recognize faces from the frames. The application only classifies the first face detected in the video. The recognition result is then used to find the details of the classified faces from the Dynamo DB. The Dynamo DB has all the details of the faces classified, like name, major, and year. The academic information extracted from Dynamo DB is put into the output S3 bucket. The output S3 bucket stores the academic information for the input videos.



2.2 Autoscaling

Auto-scaling of this application is looked after by AWS Lambda. AWS Lambda gets many in-flight requests, for each concurrent request, Lambda provisions a separate instance of the execution environment. As the function receives more requests, the Lambda function handles scaling by increasing the number of execution environments until the maximum concurrency limit of execution environments is reached. By default, there are 1000 concurrent execution environments for an account in a region. [1]

2.3 Member Tasks

Shubham Chawla - 1227415724

Shubham's contribution to the project involved creating a Docker image and verifying the AWS Lambda script's functionality by manually uploading videos to the Amazon S3 bucket. This task allowed him to understand better how to streamline cloud deployment using Docker. At the same time, he used the AWS APIs to mimic Lambda triggers on the local development environment to ensure the functionality of the project. He was also responsible for validating the smooth functioning of the application on the cloud. Collaborating with his peers, Shubham's efforts were critical to ensuring the project's overall success. One of his peers wrote the "Handler" python script to recognize faces from video frames exported by the "ffmpeg" command line program. At the same time, another ensured that Lambda triggers and DynamoDB worked correctly on AWS. Together, they delivered a functional product that worked seamlessly on the cloud.

Anurag Banerjee - 1225497506

In this project, Anurag created a Python script that is triggered every time a video is uploaded to the video upload bucket in S3. The first part of the script involves obtaining the file name and other pertinent details from the event that are passed to the handler. Subsequently, the video file is downloaded from S3 and stored in a temporary directory. After this, the “ffmpeg” tool is utilized to extract the frames from the video as JPEG images.

Furthermore, the known face names and their encodings are loaded from the encoding file provided. Next, a for loop is employed to traverse through each of the images in the temporary directory. For each image, the face_recognition library is used to extract face locations and encodings. The face_recognition library is then used to find matches between the detected faces in the image and the known faces. Once a match is found, the for loop is terminated.

Subsequently, the details for the recognized name are fetched from dynamoDB and uploaded to the output bucket in S3. The overall process aims to recognize the faces in the uploaded video and extract their corresponding information from the database. This is achieved through the integration of various libraries and tools in the script, including ffmpeg, face_recognition, S3, and dynamoDB.

Vedant Munjaji Pople - 1225453164

In this project, Vedant primarily worked on setting up and managing the AWS services part of the project. He started working on setting up the S3 buckets. There are 2 AWS S3 buckets. One of them is the Input bucket that gets the input from the workload generator. The other S3 bucket is used to store the outputs of the recognized images. He configured Amazon Lambda to recognize the images from the input bucket. The lambda function uses a docker container image to recognize images from the input. The recognized results are stored in the output bucket. The recognized results are being searched for in the DynamoDB, where Vedant entered the tables containing the academic information of the students. The results along with the academic information about students are stored in the output bucket and then presented to the user.

3. Testing and evaluation

The code has been tested extensively on the local machine. The handler function was tested along with the student_data json file locally. The handler function could use the manual input from the test cases provided. Videos from these inputs were parsed using the ffmpeg library. The still frames obtained were recognized using the recognition function. The recognition function was then tested with a variety of images generated. Iterative changes were made to the function to improve its performance. The ability to fetch data from the json was also tested along with recognition.

4. Code

Requirements.txt:

The requirements.txt file contains a list of python packages that are required to run the program.

Mapping:

The Mapping file contains the mapping of output for a particular input. A single line in the mapping file contains the name of all the videos, followed by the output i.e. the major and year of the student from the database.

Entry.sh:

The entry.sh file is the entry point for the execution of the docker file in the virtual environment. This file contains commands to run the lambda function in both cases, once if the runtime API is present, or when the API is not present.

Handler.py:

The handler.py file is the most important file in this project. This file contains the logic for the classification of the images formed by the ffmpeg library. First of all, all the required packages are imported. This includes packages like boto3, face_recognition, pickle, os, numpy, and urllib. Then the constants like the Input bucket, Output bucket, the path to the files, and image extensions for the generated images are defined. A method, open_encoding is made, this opens the file, and the pickle file for the model is loaded in it. The file is then closed and the pickled model is returned. The next method generates a response from the entered student data json file. The next method, face_recognition_handler is responsible for the recognition of faces. The method gets the video name and makes a session using Amazon credentials. The method then forms the directories for taking in videos. The images list is created and all the image names are added to it. If the image name is valid, the face_recognition library works and recognizes it. The recognized name is then searched for in the DynamoDB. The student_data table uses recognized name as the key and searches for details in the table. These details are then put into the Output bucket.

5. References

1] Lambda function scaling by AWS documentation
https://docs.aws.amazon.com/lambda/latest/dg/lambda-concurrency.html

MCS Portfolio report - Shubham Chawla - 1227415724

Shubham's main role in the project was to create a Docker image and verify the AWS Lambda script's functionality. The creation of the Docker image was crucial in ensuring that the Lambda function could execute seamlessly on the cloud. Shubham worked closely with his peers to develop a functional application that could successfully process videos and extract academic information from them. The educational information was then stored in an output S3 bucket, ensuring efficient and effective processing of user videos.

To accomplish this, Shubham first set up a local development environment for AWS Lambdas to work and trigger execution by video uploading to the S3 bucket. This involved configuring the Lambda function to trigger when new videos were uploaded to the input S3 bucket. Shubham worked tirelessly to ensure that the Lambda function was triggered correctly and that the output was saved to the correct output S3 bucket.

Shubham's contribution to the project was critical in ensuring that the application was functional and could process videos seamlessly. Alongside his peers, who contributed by writing the Python script to recognize faces and ensuring that Lambda triggers and Dynamo DB worked correctly on AWS, Shubham delivered a functional application that could successfully process videos and extract academic information from them.

In conclusion, Shubham's contribution to the project was instrumental in ensuring that the application was deployed correctly on the cloud. His efforts in creating a Docker image and verifying the AWS Lambda script's functionality were critical in streamlining the cloud deployment process. Shubham worked tirelessly alongside his peers to develop a functional application that could successfully process videos and extract academic information from them. The project demonstrated the use of cloud computing in an intelligent classroom setting by allowing face recognition and fetching academic information from Dynamo DB. Overall, Shubham's contributions helped to ensure the success of the project and highlighted the importance of effective collaboration in cloud computing projects.

MCS Portfolio report - Anurag Banerjee

The project aimed to implement a smart classroom assistant for educators. The application's purpose was to collect videos from the educator's classroom, perform face recognition on the collected videos, look up the recognized students in the database, and return the relevant academic information to the educator. The application workflow consisted of uploading a video to an upload bucket in S3, which would trigger the lambda function. The lambda function would then download the video from S3, separate it into its frames, recognize the faces, fetch the students' academic information from DynamoDB, and write it to the output bucket in S3.

The project was divided into three major parts: setting up services and other data in AWS, developing the python script for the handler, and dockerizing the script and validating its functionality. Within the group, I was responsible for the python script for the handler, which was invoked as soon as a video was uploaded into the upload bucket in S3. The handler received an event argument containing the necessary details of the uploaded image.

To begin, the script created a temporary directory with the prefix as the video filename, using the tempfile library in Python. The video from the triggering event was then downloaded into this temporary folder, and the ffmpeg tool was utilized to extract frames from the video. These images were saved in the same temporary directory created above, and the list of image names present in the temporary directory was stored in a list.

Furthermore, the encoding file provided to us was utilized to extract the known face names and known face encodings using the numpy library. A flag was then created with a value of false, which helped to stop the code's execution when a face was recognized. For each image in the temporary directory which was stored in the list, the face_recognition library was utilized to load the image file, extract the face locations, and extract the face encodings from these locations. The encodings were then matched with the known encodings, and once a match was found, the name was stored in a variable, and the flag was set to true, so the loop would not execute further.

Finally, the details of the recognized face were fetched from DynamoDB, and on successful extraction, the name, major, and year were stored as a comma-separated values (CSV) file in the output bucket with the name of the file as the name of the video that was uploaded. The python script for the handler played a crucial role in the application's workflow by recognizing the faces in the uploaded videos and extracting their corresponding academic information from the database.

Vedant Munjaji Pople - 1225453164

As a member of the project team, Vedant's primary role was to set up and manage the AWS services used in the project. Specifically, his focus was on setting up the S3 buckets and configuring Amazon Lambda to recognize images from the input bucket. This task involved working with multiple AWS services, including S3, Lambda, DynamoDB, and Docker.

To begin, Vedant created two S3 buckets in AWS. The first bucket, known as the Input bucket, was designed to receive input from the workload generator. The other S3 bucket was used to store the outputs of the recognized images. These two buckets were essential components of the image recognition process, which relied on Lambda functions to detect and recognize images.

To enable the image recognition process, Vedant configured Amazon Lambda to recognize images from the input bucket. This involved creating a lambda function that used a docker container image to recognize images from the input. The docker container image contained the necessary code and dependencies to recognize images from the input, which allowed for the recognition process to occur quickly and efficiently. The Lambda function was tweaked with the Timeout function and available RAM to work well.

Once the lambda function had recognized the images, the results were stored in the output bucket. Vedant then set up a DynamoDB database to store information about the students, such as their academic information. The recognized results were searched for in the DynamoDB database, where he entered the tables containing the academic information of the students. This allowed for the results to be linked to the students who were responsible for the images being recognized.

Finally, the results along with the academic information about students were stored in the output bucket and then presented to the user. This was achieved by setting up a user interface that displayed the recognized results and the academic information about the students in a clear and concise manner. This enabled the user to easily understand the results and their relevance to the students.

Overall, Vedant's role in the project involved setting up and managing the AWS services used in the image recognition process. This involved working with multiple AWS services, including S3, Lambda, and DynamoDB. By setting up the necessary infrastructure and configuring the services to work together seamlessly, he was able to enable the image recognition process to occur quickly and efficiently. This helped to improve the accuracy and efficiency of the overall project, while also enabling the user to easily understand and interpret the results.