Automated Classroom Attendance System (ACAS)

By
Adam Fenton
April 2019
# Table of Contents

Abstract.................................................................................................................................................. 1
Introduction................................................................................................................................................ 1
Project Requirements............................................................................................................................... 2
Implementation.......................................................................................................................................... 3
Results, Evaluation, and Reflection......................................................................................................... 9
Future Work............................................................................................................................................... 10
Bibliography............................................................................................................................................. 11
Abstract

Recording classroom attendance can be a time-consuming process for teachers and can add up to hours of lost productivity. The goal of this project is to improve the attendance taking process in a classroom environment. The Automated Classroom Attendance System (ACAS) is a cloud-based solution built using Amazon Web Services (AWS) for automatically recording and tracking classroom attendance using facial recognition. AWS was chosen based on its wide range of services that allow for scalability, reliability, and ease of administration. ACAS uses Amazon’s Rekognition technology to identify students and record them as present or absent. ACAS has an easy to use web application that allows teachers to create students, add classes and retrieve attendance history. The only component of ACAS that does not run in AWS is the classroom cameras, which are responsible for uploading images to AWS for processing. By implementing ACAS in the classroom, manual attendance taking is not necessary, allowing for more teaching time and increased reporting accuracy.

Introduction

Manual attendance recording can be a very time-consuming process for teachers; it can add up to hours of lost productivity over the course of a year. Manual attendance recording typically takes place by calling student name or by visually checking to see if a student is present. Until now there has been no easy way to automate the attendance taking process. An automated attendance system knows all the students in a class and takes appropriate action based on if the student is present or absent. This type of system has the potential to benefit other environments beyond classrooms such as churches and workplaces. By implementing the Automated Classroom Attendance System (ACAS), time would be saved and accuracy improved.
The primary goal of this project is to develop a cloud-based solution that will automate the attendance taking process in a self-contained classroom environment. The project will consist of three main components. The facial recognition component will be responsible for recognizing students. The web application component will allow teachers to create students, add classes and view attendance history. The camera component that will be responsible for uploading classroom images to AWS for processing.

**Project Requirements**

The first phase of the project was to outline the requirements necessary to be successful. The following criteria are broken down into the major components that together will create a comprehensive solution.

1. Facial Recognition
   a. Third-party software that is easy to integrate into the project framework. This project is not about creating facial recognition software from scratch, which is a very complicated process. There are many open source and commercial products available.
   b. The software needs to have a very high success rate of recognizing faces to improve accuracy.
   c. Using more images of the same student will increase the recognition success rate, but it is time-consuming to gather and upload multiple images of the same student. The ideal solution is to find a balance between using the lowest number of images necessary, while also achieving a high success rate.
   d. The process of recognizing students needs to be quick and reliable to handle hundreds of requests within a short time period.

2. Capturing/Storing Classroom Images
   a. Cameras will be used for capturing classroom images, and those images will be uploaded to a location that is accessible to the facial recognition software for centralized processing.
b. A centralized storage location with the ability to scale in size while still maintaining quick response time.

3. Web Application User Interface
   a. The web application must have an intuitive interface that allows teachers and administrators to complete the following tasks:
      i. Create teachers, students and classes
      ii. View teacher and student homepages
      iii. Enroll students in classes
      iv. View attendance history by class or student

4. Database
   a. The database must be able to support the web application backend and store the attendance results.

5. Cloud
   a. Cloud-based solution that is developed using AWS, Microsoft Azure or Google Cloud.

Implementation

Facial recognition is one of the most critical components of the project and that makes it a logical place to start the development process. After doing some extensive research, the first tool tested was OpenCV. OpenCV (Open Source Computer Vision Library) is an open source computer vision library with machine learning capabilities [1]. The two features that made OpenCV appealing was its ability to use a camera to capture video, and its built-in facial recognition library.

OpenCV can be programmed using C++, Java or Python. Python was used to program OpenCV for this project because of its prebuilt libraries and detailed documentation. The image capturing feature of OpenCV was successful in capturing images and will be used for processing classroom images as part of the solution. The Facial recognition feature in OpenCV uses the Local Binary Patterns Histograms LBPH algorithm [2], but it required multiple images of each student to achieve a high success rate in recognition. The documentation
recommended at least eight images of each student, however that number appeared to be much higher because even after increasing it to 30 images the success rate was still in the 40 percent range. That range does not meet the requirements. Since OpenCV did not meet the success rate, it was necessary to explore other tools.

After exploring other recognition tools, Amazon Web Services (AWS) was chosen because their services met all the requirements. The services that were used include S3 (Simple Storage Service), Rekognition, Lambda, RDS (Relational Database Services), EC2 (Amazon Elastic Compute Cloud) and CloudWatch. All programming was done using the Python programming language because OpenCV and AWS supported that language. Instead of using full video, only classroom images were used for facial recognition processing because it reduced the amount of bandwidth needed to the upload images and reduced cost.

**S3**

S3 is an object storage service that offers scalability, availability, security, and performance [3]. S3 will store student profile photos and classroom images for facial recognition processing. S3 uses the term “bucket” to describe a folder that contains data.

**Rekognition**

Rekognition is an API that can quickly analyze image or video files that are stored in Amazon S3. It can be used for object detection, facial detection, facial recognition, facial comparison and facial analysis. [4]. Since Rekognition can only utilized S3 buckets, two buckets will need to be created. The student bucket will store the profile picture of the student and the classroom bucket will store the images from the classroom camera. When a student is created using the web application their profile picture will be uploaded to the student bucket.

The first step in performing facial recognition is done by creating a collection. A collection is a searchable index of face feature vectors [5]. The
collection will contain each student’s indexed face that can be used to perform facial recognition matching. The next step is to process each student’s picture and reference the IndexFaces API to add the student to the collection. When all the students have been added to the collection, images being uploaded from the camera into the classroom bucket will be used to compare and find matches in the collection. The SearchFacesByImage API detects faces from the classroom camera images, but it only analyzes on the largest face in the image. This is a downfall because if there are three students in an image, it will only detect the largest face in the image and therefore only process one student. To resolve this issue the DetectFaces API must first be called. The DetectFaces returns the bounding box for each face in the image. The bounding box coordinates are used to extract each face from the image and process them independently using the SearchFacesByImage [6]. If a match is found, the student identification number and confidence score will be returned.

**Lambda**

Lambda is a serverless environment that allows code to run without the need to provision or manage servers [7]. The code can also be set up to trigger based on specific actions. In this project, there are two Lambda functions. The recognitionaddtocollection function automatically adds a student to the collection when a new student is created and their picture is added to the student S3 bucket. The recognitionengine function performs all the facial recognition API calls when images are uploaded to the classroom S3 bucket from the camera. It also updates the MySQL database with the results that will be used in the web application. Lambdas save time by not having to set up servers and each function can run independently which allows for increased scalability.

**Amazon RDS**

Amazon Relational Database Service (Amazon RDS) is a managed service that makes it easy to set up, operate, and scale a relational database in the cloud [8]. RDS allows for all different types of databases to be
created with within minutes. MySQL is used as backend database for the web application and for storing the facial recognition results.

**Amazon EC2**

Amazon Elastic Compute Cloud (Amazon EC2) are easy to spin up virtual servers that provide resizable compute capacity [9]. The web application is hosted on a Red Hat EC2 server.

**CloudWatch**

The CloudWatch service allows for collection and storage of logs from resources, applications, and services in near real-time [10]. The Lambda functions store execution logs in CloudWatch for troubleshooting issues and tracking performance statistics.

**Django Web Application Framework**

The web application was developed using Django, which is a Python web framework. It is hosted in EC2 and uses a MySQL backend database that is managed by RDS. The application provides an easy to use interface for the user to perform all the tasks that are required for automating the attendance process. When accessing the site, the user is presented a list of all the classes that have been created. Figure 1 displays the layout of the homepage. The user can create a class by entering a teacher name, grade and year. All those fields must be unique before the class can be created. (Figure 2)

![Figure 1: class list](image)

<table>
<thead>
<tr>
<th>Teacher</th>
<th>Grade</th>
<th>Year</th>
<th>Class id</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fenton</td>
<td>K</td>
<td>2019</td>
<td>1</td>
</tr>
<tr>
<td>Smith</td>
<td>1</td>
<td>2019</td>
<td>2</td>
</tr>
</tbody>
</table>
After selecting a class, all students enrolled are displayed with the ability to add a new student to the class. (Figure 3) When creating a new student, the student id must be unique and a picture of the student will be uploaded to the student S3 bucket. That picture is used to perform facial recognition matching. (Figure 4)
When viewing class attendance, the status will either be present (green) or absent (red). The first date is the time the student was first recognized and the last date is the most recent time the student was recognized for the day. The confidence score is between 0 and 100, and it determines the confidence of the facial recognition match. (Figure 5) The student homepage displays the student’s picture and provides a date range to be selected for historical attendance reporting. (Figure 6)

The diagram in figure 6 displays how each component is connected and how the data flows from each stage to the next.
Results, Evaluation, and Reflection

Overall the project met the requirements that I set out to achieve. The Automated Classroom Attendance System works well in using facial recognition to track attendance. The web application allows teachers to create classes, and students, enroll students in classes, and in real time view classroom attendance.

When I first started this project, I envisioned a completely different solution. I wasn’t even considering using AWS as an option since I was going down the path of using OpenCV and their facial recognition libraries. When those libraries failed to provide high success rates I knew it was time to look at other options. Most third-party software provided no evaluations and required an upfront charge to test out the software. I started to shift my research toward cloud providers like Amazon, Google and Microsoft to see what they had to offer. I already had an AWS account from previous class projects, so I started evaluating Amazon Rekognition. What I liked most about Rekognition was the extensive documentation and detailed examples on how to use the APIs. Their Python examples allowed me to be up and running very quickly and after a short time, I was performing facial recognition.
What transformed my project into a complete Cloud solution was using Lambda functions. It allowed me to take the code that I would be running locally on my machine and move it into a serverless Cloud environment.

Overall, I was able to immerse myself in the different serves that AWS provided which helped me build out a complete solution. The services allowed me to spend more time developing rather than trying to get systems installed and configure. I can see the benefits of why companies are shifting their development towards Cloud.

**Future Work**

This project met the requirements, but has areas for future improvements. Unfortunately, I was unable to test out the solution in a real classroom environment due to privacy concerns. I want to implement the solution in a classroom with students to verify if it can process images simultaneously and scale to multiple classrooms. Currently, ACAS only uses classroom images, but I would like to add the ability to process streaming video. There are also improvements that can be made in the web application that pertain to security and viewing attendance history. A login page would allow users to authenticate to gain access to the application. Currently, the user can only view class attendance for a specific date; a date range would allow the user to view class attendance over a longer period of time.
Bibliography

[1] https://opencv.org/about.html