SMART ATTENDANCE MANAGEMENT

SYSTEM USING FACE RECOGNITION ALGORITHM

Ruthvik Sai Kottala

December 6th, 2023

ABSTRACT

A traditional attendance system involves the teacher calling out the students’ name or ID in- order to mark the attendance. With the increase in the number of students the management of attendance has become a tedious job. But the attendance management is a crucial job as it helps in tracking and managing students’ attendance during lectures and exams.

The proposed project aims to improve the attendance of students. Initially the images of the students are taken as a reference for the attendance, then the face detection and recognition are useful to recognize the person against the images of the students taken. The recognized person data is stored. The face recognition is one of the most challenging problems up to date, and there is no technique that provides robust solution. The project aims to employ a best suitable technique of recognition such that it supports attendance management.

This project helps to reduce the tedious job of attendance management. It recognizes students from the camera present in a classroom feed thereby recognizes and marks their attendance.

1. INTRODUCTION

In traditional educational settings, the manual method of calling out names or IDs for attendance tracking has become impractical, primarily due to the increasing number of students. This project addresses the challenges associated with manual attendance management by introducing a Smart Attendance Management System. The key innovation lies in the integration of face recognition algorithms to automate the attendance process. The system employs computer vision techniques to capture reference images of students, detect and recognize faces in real-time, and subsequently record attendance data.

**Challenges with Traditional Attendance Methods:**

As educational institutions experience a surge in student enrollment, the conventional approach of calling out names or IDs during attendance has proven inefficient and time-consuming. This manual process becomes increasingly impractical, leading to potential errors and difficulties in maintaining accurate attendance records.

**Proposed Solution:**

The proposed Smart Attendance Management System offers a contemporary solution to these challenges. By leveraging face recognition algorithms, the system shifts from manual tracking to an automated, real-time attendance management process. The core functionalities include capturing reference images of students, employing face detection and recognition algorithms to identify individuals during lectures or examinations, and recording attendance data seamlessly.

**Key Features of the System:**

**Real-time Attendance Tracking:** The system operates in real-time, eliminating the need for manual intervention during attendance calls.

Face Recognition Technology: Utilizing face recognition algorithms, the system identifies and matches faces against a predefined database of student images.

Attendance Record with Timestamps: Each instance of recognized attendance is recorded in the database, complete with timestamps, providing a comprehensive record of when each student was marked present.

**Importance of Attendance Management:**

Managing attendance is crucial for monitoring student participation and engagement during lectures and examinations. The system not only simplifies the administrative burden on educators but also enhances the accuracy and efficiency of attendance tracking.

In summary, the proposed Smart Attendance Management System represents a transformative step in addressing the limitations of traditional attendance methods. By seamlessly integrating face recognition technology, the system offers a robust and automated solution for accurate attendance tracking in educational institutions.

**2. METHOD**

Developing an intelligent attendance management system, some steps need to be followed to achieve this Successful task. The steps are definable as follows:

* + - Database creation
    - Face detection
    - Feature extraction
    - Face recognition
    - Redundancy removal
    - Report generation

A diagram of a face detection process

Description automatically generated

## 2.1 Database Creation

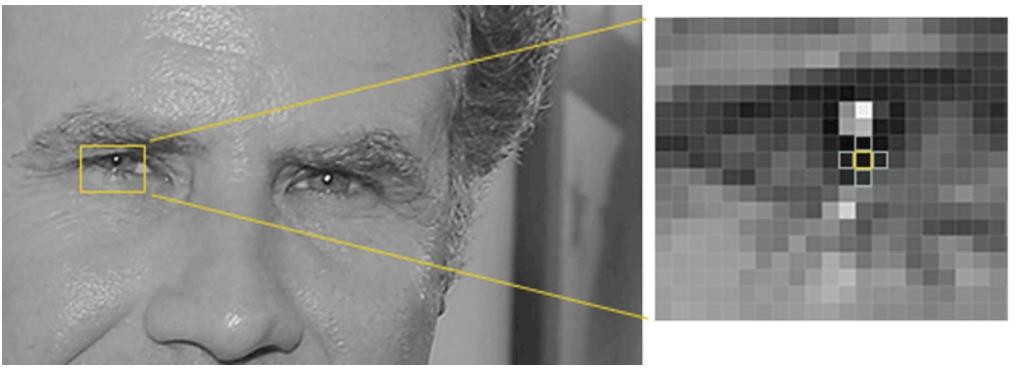
In the first step, the database will be created at the time of enrollment of students. The database will store generic information of students like names or identification numbers. Also, the images of the students are to be captured by the system for training of the proposed system (i.e., to recognize the student from the stored images).

**2.2 Face detection**

The second step in our process is face detection. We need to locate the faces in a photograph before we can try to recognize or identify them.

Face detection went mainstream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. We’re going to use a method invented in 2005 called Histogram of Oriented Gradients or just HOG for short.

To find faces in an image, we’ll start by making our image black and white because we don’t need color data to find faces. Then we’ll look at every single pixel in our image one at a time. For every single pixel, we want to look at the pixels that directly surrounding it:



*Figure 5.2 Comparison of every pixel with its surrounding pixels*

Our goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Then we want to draw an arrow showing in which direction the image is getting darker:

A black and white pixelated image

Description automatically generated

*Figure 5.3 Obtaining the gradient*

If we repeat that process for every single pixel in the image, we end up with every pixel being replaced by an arrow. These arrows are called gradients, and they show the flow from light to dark across the entire image:

A close-up of a person's eye

Description automatically generated

*Figure 5.4(a) Original Image*

A black and white background

Description automatically generated

*Figure 5.4 (b) Image converted to gradient*

This might seem like a random thing to do, but there’s a really good reason for replacing the pixels with gradients. If we analyze pixels directly, dark images and light images of the same person will have totally different pixel values. But by only considering the direction that brightness changes, both dark images and really bright images will end up with the same exact representation. That makes the problem a lot easier to solve.

But saving the gradient for every single pixel gives us way too much detail. We end up missing the forest for the trees. It would be better if we could just see the basic flow of lightness/darkness at a higher level so we could see the basic pattern of the image.

To do this, we’ll break up the image into small squares of 16x16 pixels each. In each square, we’ll count up how many gradients point in each major direction (how many points up, point up-right, point right, etc.…). Then we’ll replace that square in the image with the arrow directions that were the strongest.

A close up of a person's face

Description automatically generatedA black background with white lines

Description automatically generatedThe result is we turn the original image into a very simple representation that captures the basic structure of a face in a simple way

*Figure 5.5 (b) Image after obtaining gradients for 16x16 pixels*

A close-up of a face pattern

Description automatically generatedTo find faces in this HOG image, all we must do is find the part of our image that looks the most like a known HOG pattern that was extracted from a bunch of other training faces:

*Figure 5.6 Image after HOG representation*

Using this technique, we can now easily find faces in any image. The detected faces have a problem that faces turned different directions look totally different to a computer.

To account for this, we will try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps.

To do this, we are going to use an algorithm called face landmark estimation. There are lots of ways to do this, but we are going to use the approach invented in 2014 by Vahid Kazemi and Josephine Sullivan.

The basic idea is we will come up with 68 specific points (called landmarks) that exist on every face — the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face.

A dot to dot game

Description automatically generated

*Figure 5.7 68 face landmarks*

Now that we know where the eyes and mouth are, we’ll simply rotate, scale, and shear the image so that the eyes and mouth are centered as best as possible. We won’t do any fancy 3d warps because that would introduce distortions into the image. We are only going to use basic image transformations like rotation and scale that preserve parallel lines (called affine transformations):

A face with a drawing

Description automatically generated with medium confidence*Figure 5.8 Detected Face converted to perfectly centered face*

Now no matter how the face is turned, we are able to center the eyes and mouth in roughly the same position in the image. This will make our next step a lot more accurate.

**2.3 Feature Extraction (Encoding Faces):**

Now we are to the actual problem of telling faces apart. The simplest approach to face recognition is to directly compare the unknown face we found in the previous step.

With all the pictures we have of people in the database. But comparing each face in the database with the detected face will be a huge task, and if the database is large then it would take a large amount of time to recognize the exact match of the detected face.

What we need is a way to extract a few basic measurements from each face. Then we could measure our unknown face the same way and find the known face with the closest measurements. For example, we might measure the size of each ear, the spacing between the eyes, the length of the nose, etc. But we do not know the exact parameters and measurements which help us. It turns out that the measurements that seem obvious to us humans (like eye color) don’t really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network. But instead of training the network to recognize pictures objects like we did last time, we are going to train it to generate 128 measurements for each face.

The model structure used is.

A grey rectangular sign with black text

Description automatically generated

Figure 5.9 Model Structure

The network consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by triplet loss during training.

L2 normalization may be defined as the normalization technique that modifies the dataset values in a way that in each row the sum of the squares will always be up to 1. It is also called least squares.

The training process works by looking at 3 face images at a time:

* Load a training face image of a known person.
* Load another picture of the same known person.
* Load a picture of a totally different person.

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart. It is represented in the figure below (Figure 5.11).

A diagram of an anchor

Description automatically generated

*Figure 5.10 Triplet Loss training model*

The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity.

*Figure 5.11 Example of a single triplet training step*

A diagram of a person's face

Description automatically generated

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements. Machine learning people call the 128 measurements of each face an embedding.

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. But once the network has been trained, it can generate measurements for any face, even ones it has never seen before. So, this step only needs to be done once. There are several trained networks published by Open Face. So, all we need to do ourselves is run our face images through their pre- trained network to get the 128 measurements for each face.

## 

## 2.4 Face Recognition

Initially all the faces in the database are encoded. Then during real time operation the detected faces are encoded too. All we must do is find the person in our database of known people who has the closest measurements to our detected image.

We can do that by using any basic machine learning classification algorithm. We can use a simple linear SVM classifier, but lots of classification algorithms could work.

All we need to do is train a classifier that can take in the measurements from a new detected image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name or id of the person.

## 

## 2.5 Redundancy removal

A persons’ face may be repeatedly detected but noting down the person data every time the person is recognized leads to redundancy in the attendance and a waste of memory. So, while noting down the attendance we initially check whether the person’s name is noted down or not. If and only if the persons’ name is not present in the attendance the person’s name is added.

## 

## 2.6 Report generation

Trailing face recognition reports are generated by marking the present in front of the students’ name or enrollment number in database during a lecture.

3. EXPERIMENTS

**3.1 DATASETS**

In the course of our experiment, we conducted facial recognition-based attendance tracking using a proprietary dataset consisting of images of students enrolled in the class. Our team meticulously curated a dataset comprising facial images of students associated with the class. Below are the images we used for our dataset:

A person with a beard and mustache wearing a grey jacket

Description automatically generatedA person with a beard

Description automatically generated

A person with a beard

Description automatically generated

The system will use these images to compare to the faces it scans through the webcam and give the result by either updating the attendance records or stating the person as unknown.

**3.2 EVALUATION METRICS**

In the context of the Smart Attendance Management System utilizing face recognition, the evaluation metrics play a crucial role in quantifying and assessing the performance of the implemented algorithm. The following metrics are employed to comprehensively evaluate the system:

- Accuracy:

Accuracy represents the ratio of correctly identified students to the total number of students. It provides a general measure of the algorithm's correctness in recognizing individuals. The accuracy metric is particularly valuable for understanding the overall performance of the face recognition system.

- Precision:

Precision focuses on the accuracy of positive predictions, indicating the ratio of correctly identified students to the total number of identified individuals. Precision is relevant for assessing the system's ability to avoid false positives, ensuring that the recognized individuals are indeed present in the dataset.

- Recall:

Recall, also known as sensitivity or true positive rate, assesses the ability of the system to correctly identify all relevant instances. In the context of attendance management, recall measures the ratio of correctly identified students to the total number of students who are actually present. A high recall value indicates the system's effectiveness in capturing all instances of student attendance.

- F1 Score:

The F1 score is the harmonic mean of precision and recall, providing a balanced evaluation that considers both false positives and false negatives. It is particularly useful when there is an uneven class distribution. A higher F1 score indicates a better balance between precision and recall, showcasing the overall robustness of the face recognition algorithm.

These evaluation metrics collectively offer a comprehensive understanding of the face recognition system's performance. By considering accuracy, precision, recall, and F1 score, the assessment becomes nuanced, allowing for a more detailed analysis of the system's strengths and potential areas of improvement. The chosen metrics provide a well-rounded evaluation framework, essential for ensuring the reliability and efficacy of the Smart Attendance Management System.

**3.3 CODE SETUP (Importing libraries)**

Install dlib and face\_recognition (not always on the newest version):

pip install dlib and then pip install face\_recognition.

Requirments:

* Microsoft Visual Studio 2015 (or newer) with C/C++ Compiler installed. (Visual C++ 2015 Build Tools didn't work for me, and I got into problems in compiling dlib)
* Python3 (I used Python3.5 x64 but the other versions may work too)
* CMake for windows and add it to your system environment variables.
* (ONLY FOR older versions of dlib) Boost library version 1.63 or newer. Also, you can use precompiled binaries for specific MSVC you have, but I don't suggest.(This step is optional)

A person taking a selfie

Description automatically generated**3.4 RESULTS**

A person sitting on a couch

Description automatically generated

*Figure (a) Figure (b)*

A person taking a selfie

Description automatically generatedA person taking a selfie

Description automatically generated

*Figure (c) Figure (d)*

A screenshot of a computer

Description automatically generated

*Figure (e)*

**3.5 ANALYSIS AND DISCUSSION**

* When we start the program, we will be given a prompt to press “o” to start or “q” to quit.
* On hitting “o” the program execution will begin and all the images of the students that are stored in our dataset will get encoded.
* Then the program will proceed to take the image of the person in front of the camera(webcam) and will try to encode that image as well.
* In the next step it will compare this encoding with the encoding list we had created with the images in our dataset.
* After a thorough comparison the program with then decide whether there is a match for the person in our dataset or not. Based on this decision there will be two choices

1. If there is a match and the student has already marked their attendance, then it will simply update the timestamp as shown in figure e. If the student has not marked his attendance yet, then a new record will be created with the timestamp.
2. If there is no match found with the images from our dataset, then the program will flag that person as unknown and will not update any records in the database.

* When the match is successful the output will be as shown in figures a, b, c and e.
* When there is no match in our dataset then the output will be as shown in figure d.

An example for the image encodings are shown in the figures below:

A number of numbers on a white background

Description automatically generated

A person with a beard and mustache wearing a grey jacket

Description automatically generated

A person taking a selfie

Description automatically generatedA number of numbers on a white background

Description automatically generated

**4. CONCLUSION**

The system proposed has 5 steps involved. Dataset creation is initially done to extract the features which will help in the later stages for recognition purpose. The facial features are extracted using CNN (convolution neural networks). Later the actual process of attendance marking starts. The system gets video feed from camera. From this feed the faces of the students are detected using HOG accompanied by linear SVM classifier. Facial features from these detected faces are extracted using the same CNN. After the facial features are extracted the linear SVM helps us in recognizing the face which has the closest match with the detected face. The recognized face id is used in order to mark attendance in database.

The system requires only a single image of the person to be present in the dataset for the recognition. So, the dataset space used is quite less.

The proposed system meets the objective of achieving great deal of precision and less computational complexity. This system is cost-efficient and less manual work is needed.

**5. CONTRIBUTION**

**5.1 CODE:**

* My role included curating a bespoke dataset by collecting images of students associated with the class.
* Streamlining the project setup by providing clear instructions for installing and importing necessary libraries. This includes popular computer vision libraries such as OpenCV and face\_recognition, ensuring a smooth and accessible development environment.
* Leveraging the face\_recognition library, I have implemented a robust image encoding mechanism. This process converts facial images into unique encodings, creating a reference set for subsequent recognition tasks. The utilization of face\_recognition library contributes to the efficiency and accuracy of our facial recognition system.
* Utilizing the face\_recognition library, I have incorporated a face detection mechanism. The library provides a high-level interface for face detection, simplifying the implementation and improving efficiency.
* Throughout the development process, meticulous attention was given to ensure that the program ran without any issues. Rigorous testing and debugging were conducted to identify and address potential issues, resulting in a stable and reliable facial recognition-based attendance system.

**5.2 REPORT:**

* To get in-depth knowledge on face detection, elucidating its underlying mechanisms, functionality, and the process by which it identifies facial features. This educational component contributes to a deeper understanding of the technology employed in our facial recognition-based attendance system.
* Emphasizing how to perform face recognition using image encodings. I provided insights into the interpretation of facial features, the comparison of encodings, and the decision-making process that leads to successful face recognition.
* Aiding in common tasks such as clearly defining the problem statement, presenting results, and providing detailed analysis descriptions. These components are essential for effective communication and documentation, ensuring that users and stakeholders can comprehend the project's objectives, outcomes, and implications**.**

**6. REFERENCES**

https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face- recognition-with-deep-learning-c3cffc121d78