iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 9 **Issue:** VIII **Month of publication:** August 2021

DOI: https://doi.org/10.22214/ijraset.2021.37499

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VIII Aug 2021- Available at www.ijraset.com

Face Recognition Attendance System using Raspberry Pi

Tresnor Menezes[1], Samuel Furtado[2], Ruhi Morajkar[3], Pushpak Verlekar[4], Shreedatta Sawant[5], Saurabh Naik[6]
[1, 2, 3, 4, 5]*Department of Computer Engineering Agnel Institute of Technology and Design Assagao, Goa*
[6, 7]*Assistant Professor, Department of Computer Engineering Agnel Institute of Technology and Design Assagao, Goa*

Abstract: Face recognition is used for security, authentication, identification, and has got many advantages over conventional methods. It is being used in many sectors since it is contactless and non-invasive.
Billions of images have been uploaded on social media networks and are crawled by search engines over many years. These images may include many different faces. The increase in computing capability and collected data has helped in creating more powerful neural network models. [1]
This project thesis aims to create an attendance system which uses face recognition biometric authentication as the currently used manual attendance system is cumbersome to maintain and time consuming. Face recognition prohibits the chance of students marking attendance for their peers (proxy attendance).
Keywords: Face Recognition, Face embeddings, Face Detection, Image Processing, Raspberry Pi automation.

I. INTRODUCTION

With the continuous development of technology, the demand for safety and security is sky-rocketing. Face recognition is used on a day-to-day basis in our life, especially in information security, security systems, human-computer interaction.
Researchers are working towards improving the response speed and accuracy of the face recognition system. The technology has improved a lot by the introduction of deep learning. There is still a long way to go to improve the recognition accuracy of face recognition methods in real scenarios.
Face recognition has the merits of low intrusiveness and high accuracy. Over many decades, many researchers have put forward different face recognition methods, due to the increased number of real-world applications requiring the technology.
Face recognition is a vital application of Image Processing and is used in many different sectors. Identification of students in a class for attendance is one application of face recognition. Monitoring and maintenance of attendance records plays an important part in the analysis of performance of any organization. The main purpose is to digitise the traditional process of taking attendance. Automated Attendance Management performs the daily tasks of marking attendance and reports with little or no human assistance or intervention.

II. EXISTING SYSTEM

In the existing system, taking attendance wastes a lot of time and is also quite inefficient. Also, students can give attendance for their friends (proxy attendance). The system that we are building will be fool proof and automated.

III. PROPOSED SYSTEM

The objectives of the project are detection and recognition of unique faces in a classroom, effective recognition of unique faces in a crowd (classroom) and automated updates to the database.
The main objective of this project is to create a system that simplifies and automates the process of recording and tracking students' attendance through face recognition technology using Raspberry Pi.
We aimed to reduce errors introduced in manual process of attendance. Increase security where student cannot present themselves or their friend while they are not present in class. Flexibility, where lecturers are capable of editing attendance records, calculate percentage of absenteeism and send emails to students. The task of the proposed system is to click images of students in a class and recognise the faces and mark attendance in the database.

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 9 Issue VIII Aug 2021- Available at www.ijraset.com

IV. OVERVIEW

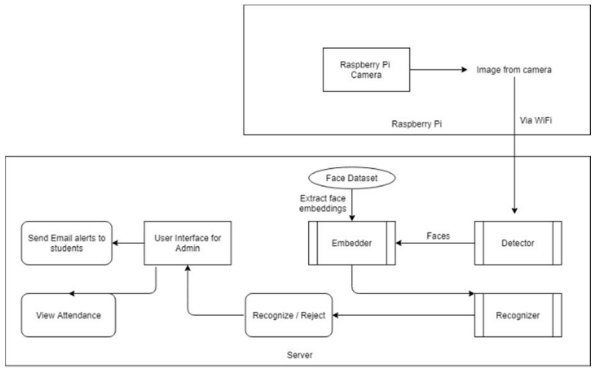


Fig 1. Block Diagram

The flow of the project:

1) We have a GUI that allows the admin to register students. Details like their name, roll number, email, photograph are taken and saved into the database.

2) After successful registration of all students of a class, the admin has to click on 'save embeddings' to generate embeddings for every student in the database and save the embeddings into a pickle file for later use.

3) Now, the raspberry pi automatically clicks images every 5 minutes and saves it along with the time stamp of when the image was taken (this helps with marking attendance for the lecture). At the end of every day, the admin has to click a button on the VNC viewer to transfer the folder of all the clicked images from the Pi to the Server.

4) Once these images are transferred, the admin has to click 'Mark attendance for the day'. This runs a code that automatically recognises the students that were present throughout the day and marks attendance for the subjects that they were present for using the help of the time stamps that we spoke of earlier. The attendance is updated in the database.

5) At the end of the month, the admin has to click on 'Report for month' that automatically calculates the total attendance for every student for each subject along with their overall attendance and sends an email to all the students with the data.

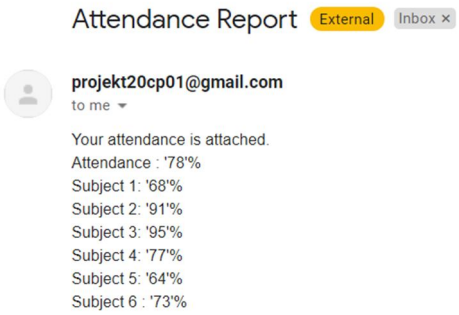


Fig 2. Email with attendance report

A. Face Detection

The Face detection algorithm is used to detect the faces present in the images and extract it.

In this project we use the MTCNN method to detect faces.

MTCNN: This method works for faces having various orientations in images and can detect faces across various scales and even handle occlusions. [3]

The objectives during this step are as follows:

1) Retrieve images stored in database.

2) Read images through matplotlib's imread() function.

3) Detect faces through the MTCNN algorithm.

4) Extract faces from a picture.

B. Face Embeddings

Facenet DNN is responsible for the quantification of each face in an image and belongs to the OpenFace project.

FaceNet has trained its output to be 128-D embedding using a triplet-based loss function. The triplets use two matching faces and a non-matching face and the main objective is to increase the distance between the positive pair and the negative pair.

We want an embedding f(x), from a picture x, such that the squared distance between all faces, without regard to the various conditions, of the same person is little, whereas the squared distance between a pair of faces from different people is large.

Hence, we wish to make sure that a picture x_i^a (anchor) of a particular person is closer to any or all other images (positive) of the identical person than it's to any image x_i^n (negative) of the other person. [2]

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2$$

Where α is a margin that is enforced between positive and negative pairs. [2]

The training process comprises of:

1) The input data to the network.

2) The triplet loss function.

Each input group of data consists of three images:

a) Anchor.

b) Positive image.

c) Negative image.

Our face is the anchor whose embeddings we want and has identity A.

The second image(positive) contains a face of person A. The negative image has a different identity, a person B or C such that the anchor and positive image belong to the identical person while the negative image doesn't has a different face.

The 128-d embeddings are calculated by the neural network and the weights are tweaked using the triplet loss function such that:

- The 128-d embeddings of the anchor and positive image are similar.
- The embeddings for the negative image are moved farther away.

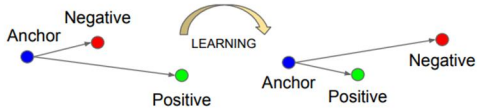


Fig 3. Triplet loss function

C. Face Recognizer

A VGGFace2 model will be used for face verification. [9]

This involves calculating a face embedding for a new given face and comparing the embedding to the embedding for the only example of the face known to the system.

Cosine distances are calculated between two embeddings and faces are said to match or verify if the distance is below a predefined threshold, often tuned for a specific dataset or application. [9]

This model can then be used to make a prediction, which will return a face embedding for one or more faces provided as input. [9]

Verification can be performed by calculating the Cosine distance between the embedding for the known identity and the embeddings of candidate faces. This can be achieved using the cosine() SciPy function. The maximum distance between two embeddings is a score of 1.0, whereas the minimum distance is 0.0. A common cut-off value used for face identity is between 0.4 and 0.6, such as 0.5, although this should be tuned for an application. [8]

V. IMPLEMENTATION

A. Clicking images on Raspberry Pi

On the pi, we created a bash file that automatically clicks an image using the raspistill command line tool and save it into the camera directory. The filename is saved using the TIME variable that automatically takes the time, the format is HM, so if the time is 10:30, it saves it as 1030. Now, in order to automate that bash script to run every 5 minutes, we use crontab.

At the end of the day, we transfer the images in the ‘camera’ directory to the server using VNC Viewer which has an inbuilt feature to easily transfer data from the pi to the server.

B. Face Detection

The first step is to load a picture as a NumPy array, which we will achieve using the Matplotlib imread() function.
We use the MTCNN library to make a face detector and extract faces for our use with the VGGFace face detector models in subsequent sections.
Next, we create an MTCNN face detector class and use it to detect all faces.
We use the PIL library to resize the image of the face to the specified size; the model that we use expects square input faces with the form 224×224. [8]
The function extract_face() will load an image from the loaded filename and return the extracted face.

C. Face Embeddings

A VGGFace model will be created using the VGGFace() constructor and specifying the sort of model to make via the 'model' argument.
We define a function that when given a list of filenames for photos containing a face, will extract the face from each photo via the extract_face() function developed in a prior section, then predict a face embedding for every face.
We create a connection to the database.
The sql_select query selects all the roll numbers from the database and saves it in the list names[].
We then save all the filenames ie the images of each student into the list filenames[].
Finally we call the function get_embeddings() with the filenames as a parameter which returns all the embeddings; we save this in the list embeddings[].
Now we create a variable data that saves key value pairs of names with corresponding embeddings and we dump it in a pickle file for later use.

D. Face Recognition and Attendance

In this module, we import the same extract_faces() and get_embeddings() functions.
We have another function is_match() that takes in 3 inputs: known_embeddings, candidate_embeddings and threshold.
This function determines the cosine distance between the known_embeddings and the candidate _embeddings and if it is less than the threshold, returns true ie, a match.
Now, we load the data key value pair variable that we dumped earlier, we use this for the main face recognition.
The embeddings are stored in embeddings[] and names in names[].
file_list[] is a list that contains all the filenames in the directory 'camera' where all the images clicked by the Raspberry Pi are stored. Earlier we discussed that all the filenames are actually the timestamp of each image.
Now, we send the file_list to get_embeddings() and save all the embeddings of every photo into embeddings[] list.
We run for loops across both the embeddings files to check if they match, if they do, then we insert that filename ie timestamp along with the student recognised into the database in a table 'Records'.
We now have the roll number of the student who is recognised along with the time that he was recognised.
We retrieve data from the database.
Now, we run a for loop in times[]. If the time matches in any if statement, we connect the database and increment the attendance for that particular subject only if chk for that subject is 0. If it is 1, it means that we have already given attendance for that student in that subject, this is to prevent giving attendance multiple times.

E. Report and Sending Automated Emails

We have two functions that send emails to students. The contents of the email are as follows:

1) Total Attendance
2) Subject 1 attendance
3) Subject 2 attendance
4) Subject 3 attendance
5) Subject 4 attendance
6) Subject 5 attendance
7) Subject 6 attendance

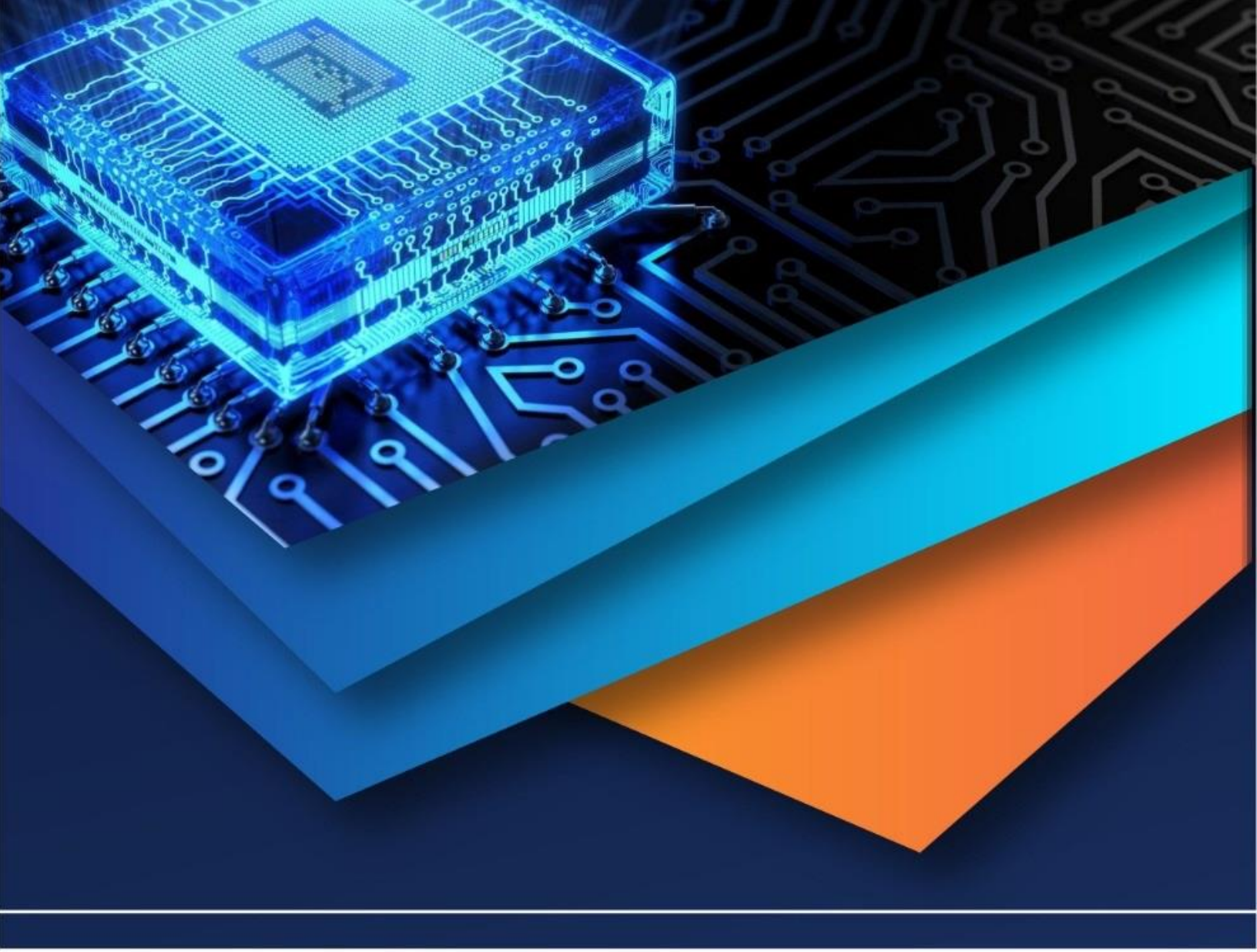
If the student has a total attendance of 75%, we use the second function that also tells the student that their attendance is low.
We assume that we have 6 Subjects for this class. Every day, each subject is taken once.
Each subject is taken 5 times a week (Saturday and Sunday is off).
Assuming a month contains 30 days, we have 22 working days. So, each subject has 22 lectures a month.
We calculate attendance for each subject as **x = (s / 22) * 100** where s is attendance for the subject.
The final total of all subject's attendance is calculated as:
TOTAL = ((Subject1[i] + Subject2[i] + Subject3[i] + Subject4[i] + Subject5[i] + Subject6[i]) / 132) * 100
All this data is then stored into the database and emails are sent to each student.

VI. CONCLUSION

At the completion of the project, we have a Face Recognition Attendance System that automatically captures the attendance of students in a classroom and automatically updates their attendance and sends reports via email monthly.

REFERENCES

[1] Kevin Santoso, Gede Putra Kusuma "Face Recognition Using Modified OpenFace" Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia, 11480
[2] Florian Schroff, Dmitry Kalenichenko, James Philbin "FaceNet: A Unified Embedding for Face Recognition and Clustering" arXiv:1503.03832v3 [cs.CV] 17 Jun 2015
[3] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, Senior Member, IEEE, and Yu Qiao, Senior Member, IEEE "Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks"
[4] Qiong Cao, Li Shen, Weidi Xie, Omkar M. Parkhi and Andrew Zisserman "VGGFace2: A dataset for recognising faces across pose and age" Visual Geometry Group, Department of Engineering Science, University of Oxford {qiong,lishen,weidi,omkar,az}@robots.ox.ac.uk
[5] "Scheduling tasks with Cron" https://www.raspberrypi.org/documentation/linux/usage/cron.md
[6] "raspistill to click images on raspberry pi" https://www.raspberrypi.org/documentation/usage/camera/raspicam/raspistill.md
[7] "Extract Data from Database using MySQL-Connector and XAMPP in Python" https://www.geeksforgeeks.org/extract-data-from-database-using-mysql-connector-and-xampp-in-python/
[8] Jason Brownlee "How to Develop a Face Recognition System Using FaceNet in Keras" https://machinelearningmastery.com/how-to-develop-a-face-recognition-system-using-facenet-in-keras-and-an-svm-classifier/
[9] Jason Brownlee "How to Perform Face Recognition With VGGFace2 in Keras" https://machinelearningmastery.com/how-to-perform-face-recognition-with-vggface2-convolutional-neural-network-in-keras/



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY