



Monitoring Employees Entering and Leaving the Office with Deep Learning Algorithms

Viet Tran Hoang, Khoi Tran Minh, Nghia Dang Hieu and Viet Nguyen Hoang

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

October 19, 2021

Monitoring Employees Entering and Leaving the Office with Deep Learning Algorithms

Tran Hoang Viet¹, Tran Minh Khoi², Dang Hieu Nghia³, and Nguyen Hoang Viet⁴
Can Tho University, Vietnam

¹thviet@ctu.edu.vn

²khoim3721005@gstudent.ctu.edu.vn

³dhnghia@ctu.edu.vn

⁴nhviet@ctu.edu.vn

Abstract. This study attempts to create a system to monitor employees entering and leaving the office using face recognition. In addition, the system also signals by LED when recognizing a staff who has clearance to enter or notifies those who do not in the area. Events of entering and leaving from staff are written into a log file for management purposes. The face-detection and image preprocessing utilize Multi-task Cascaded Convolutional Network. Feature data is then extracted from the processed images by FaceNet, which is classified by the Support Vector Machine algorithm into a model. Information of employees and logs are saved in MySQL database, which is also used in a web application using Python and Django web framework.

Keywords: Face, face recognition, feature extraction, training, learning (artificial intelligence), face detection, detectors, convolutional neural nets, databases, cameras, real-time systems, facenet, python, support vector machine, mysql, django, raspberry pi.

1 Introduction

As a part of facial image processing applications, face recognition systems' significance as a research area are increasing recently. They are usually applied and preferred for people and security cameras in modern life. These systems can be used for person verification, video surveillance, crime prevention, and other similar security activities.

Nowadays, with the non-stop growing quality of hardware like cameras, CPUs, or especially GPUs, the computing time for training process in machine learning, has reduced significantly. Face recognition, which is a case of machine learning, uses some algorithms to extract features from the input face, converting them into mathematical vectors to classify them.

Face recognition system is a complicated image-processing problem with many affecting factors like pose, angle, facial expression, illumination, occlusion, imaging conditions, and time delay (for recognition). It is a combination of face detection and recognition applications in image analysis. Detection techniques are used to find the position of the faces in a given image, while recognition techniques are used to classify said faces base on extracted facial feature components.

There are many methods to develop a face recognition application with many training algorithms and models. The applications related to this technology haven't been very popular in our city yet. Therefore, more attention should be spent on learning and practicing face recognition technology. Thus, in this work we developed a system utilizing the technology to apply in practice.

In the rest of the introduction we briefly review the related work and the main topics necessary to understand our system discussed in Section 2. Section 3 details the results of our system before drawing some conclusions and discussing further directions in Section 4.

1.1 Related Work

An attempt to tackle the task of checking attendance automatically has been proposed in [1], where the idea is to use face recognition technique, with Eigenface values, Principle Component Analysis (PCA), and Convolutional Neural Network (CNN) to implement an automated attendance management system for students of a class.

Multiple approaches have been proposed in order to overcome some of the critical challenges of face recognition under difficult conditions. A system for real-time video-based face recognition has been introduced in [2], which tackled three key challenges: computational complexity, in the wild recognition, and multi-person recognition.

Since face recognition in real-time has been a rapidly growing challenge, another work [3] has proposed the PCA facial recognition system, where the PCA is a statistical method under the broad heading of factor analysis, which aims to truncate the substantial quantity of data storage to the size of the feature space that is required to represent the data economically. However, differently from [1], this work's system has been built with OpenCV, Haar Cascade, Fisher Face, Local Binary Pattern Histogram (LBPH), and Python.

After taking notes from the mentioned works and considering among many new algorithms as well as technologies, we introduce our own approach to develop a system in this work, using multi-task Cascaded Convolutional Network (MTCNN) and Facenet.

1.2 Convolutional Neural Network

A CNN is a Deep Learning algorithm that takes an input image, assigns parameters (weights and biases) to several aspects in it, and can recognize one from the rest. The pre-processing required in a CNN is lower than other classification algorithms. With enough training, CNNs have the ability to learn these characteristics.

The architecture of a CNN is analogous to that of the connectivity pattern of Neurons within the Human Brain and was inspired by biological processes in which the connectivity pattern between neurons imitates the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field called the Receptive Field. The receptive fields of various neurons partially overlap so that they cover the whole visual area.

Thanks to the reduction in the number of parameters involved and the reusability of weights, CNN's architecture performs a better fitting to the image dataset. In other words, the network can be trained to grasp the sophistication of the image better.

1.3 Multi-task Cascaded Convolutional Networks

The MTCNN model consists of three separate convolutional networks: the P-Net, the R-Net, and the O-Net:

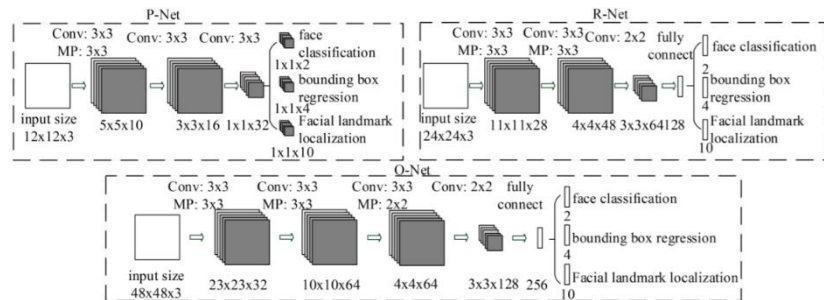


Fig. 1. MTCNN structure (Source: [5])

For every image passed in, the network creates an image pyramid, to detect faces of all different sizes. In other words, it creates multiple different copies of the same image in different sizes to search for different sized faces within the image.

For each scaled copy, a 12x12 kernel will go through every part of the image, scanning for a face. It starts in the top left corner, a section of the image from (0,0) to (12,12). This portion of the image is passed to Proposal Network (P-Net), which returns the coordinates of a bounding box if it notices a face. Then, it will repeat that process after shifting the 12x12 kernel sideways (or downwards), and it continues doing that until it has gone through the entire image. How many pixels the kernel moves by every time is known as the stride.

Each kernel would be smaller relative to a large image, so it would be able to find smaller faces in the larger-scaled image. Similarly, the kernel would be bigger relative to a smaller image, so it would be able to find bigger faces in the smaller-scaled image.

With each of these 12x12 kernels, 3 convolutions are run through with 3x3 kernels. After every convolution layer, a PReLU layer is implemented (when you multiply every negative pixel with a certain number α , which is to be determined through training). In addition, a Maxpool layer is put in after the first PReLU layer (Maxpool takes out every other pixel, leaving only the largest one in the vicinity).

After the third convolution layer, the network splits into two layers. The activations from the third layer are passed to two separate convolution layers, and a Softmax layer after one of those convolution layers (Softmax assigns decimal probabilities to every result, which all add up to 1. In this case, it outputs two probabilities: the probability that there is a face in the area and the probability that there isn't a face).

Refinement Network (R-Net) has a similar structure, but with even more layers. It takes the P-Net bounding boxes as its inputs and refines its coordinates.

Similarly, R-Net splits into two layers in the end, giving out two outputs: the coordinates of the new, more accurate bounding boxes, as well as the machine's confidence level of each of these bounding boxes.

Finally, Output Network (O-Net) takes the R-Net bounding boxes as inputs and marks down the coordinates of facial landmarks.

O-Net splits into three layers in the end, giving out three different outputs: the coordinates of the bounding box, the coordinates of the five facial landmarks (locations of the eyes, nose, and mouth), and the confidence level of each box.

1.4 FaceNet

FaceNet is a face recognition system introduced by Google in June 2015. The system directly learns a mapping from face images to a compact 128-dimension Euclidean space. The L2 distance (or Euclidian norm) between two faces embeddings directly corresponds to its similarity: faces of the same person have small distances and faces of distinct people have large distances, which is exactly like measuring the distance between two points in a line to know if they are close to each other. In an n -dimension Euclidean space, the L2 distance between two points can be expressed as

$$d_{L2}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Once this space has been produced, tasks such as face recognition, verification, and clustering can be easily implemented using standard techniques with FaceNet embeddings as feature vectors.

FaceNet system uses a deep convolutional network trained to directly optimize the embedding itself, instead of an intermediate bottleneck layer used by previous deep learning approaches, resulting in much higher efficiency. The system also introduces the concept of harmonic embeddings, and a harmonic triplet loss, which describes different versions of face embeddings (produced by different networks) that are compatible with each other and allow for direct comparison between each other.

1.5 Support Vector Machines

SVM is a class of machine learning algorithms, belongs to the area of supervised learning methods, which need labeled, known data to classify new unseen data. Using the idea of kernel substitution, it can deal with many tasks such as classification, regression, and novelty detection. Besides face recognition, SVM also has applications in handwritten characters recognition, text classification, bioinformatics, etc.

Its approach to classifying the data starts by trying to create a function that splits the data points into the corresponding labels with (a) the least possible amount of errors and (b) with the largest possible margin. This is so because larger empty spaces around the splitting function result in fewer errors because the labels are better distinguished from one another overall.

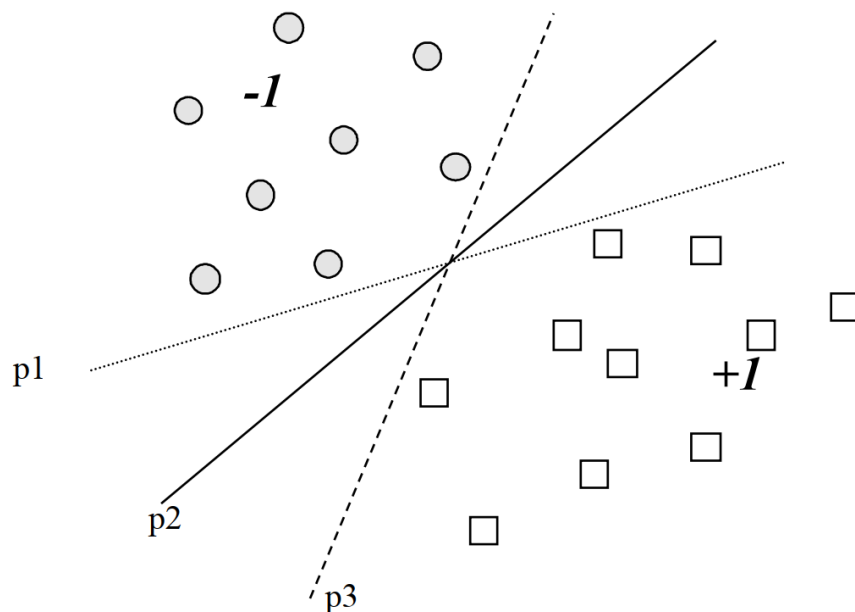


Fig. 2. Visualization of an SVM

Figure 2 shows that a data set may be separable by multiple functions without any error. Therefore, the margin around a separation function is used as an additional parameter to evaluate the quality of the separation. In this case, the separation p_2 is the better one, since it distinguishes the two classes in a more precise manner.

Formally, SVM tries to find one or multiple optimal hyperplanes in an n -dimensional space. The first attempt in the process of splitting the data is to try to linearly separate the data into the corresponding labels. For example, for a task of predicting if a day is rainy or sunny uses a data set with n data points, where each data point consists of a label $y \in \{rainy, sunny\}$ and an attribute vector \vec{x} containing the data values for the specific session. The SVM now tries to find a function that separates all the data points (\vec{x}, y) with $y = rainy$ from all the data points (\vec{x}, y) with $y = sunny$.

If the data is completely separable linearly, the separating function can be used to classify future events.

The data may not be well linearly separable or not linearly separable at all, which is often the case for real-world data. In the example given above, it can happen for example that two days have the exact same weather attributes such as temperature, humidity, etc. but it only rains on only one of them. This leads to inseparable data since the same attribute vector has different labels. This can be resolved by mapping the n -dimensional input data to a higher-dimensional space, where the data can be separated linearly.

In order to avoid overfitting in SVM, the data has to be preprocessed to identify noise and accept some misclassifications. Otherwise, the accuracy values of the SVM will be flawed and result in more erroneous classification for future events.

2 Work Details

In this section we present the design of our Monitoring employee entering and leaving the office system. As a first step, we need to identify what the users require from the system, which are the following tasks:

- Registration: new user can register to be recognized by the system in the future, this procedure must have the system taking photos of the user's face and letting the user input their personal information.
- Model training: the system processes and learns the face images from registered users to output a trained model that can recognize them.
- Face recognition and Door opening: the system can detect a person's face in front of the camera and recognize their identity, if they are a user who is authorized to enter, the system will automatically open the door and save the event to log. An announcement can be played by voice as an optional step.
- Web application: the system also provides a website for both users and admin to access, view information, and perform tasks from their own device, as long as it is connected to the same network as the server's.

With these requirements in mind, after discussing many possible approaches, we have decided to come up with the system design as follows:

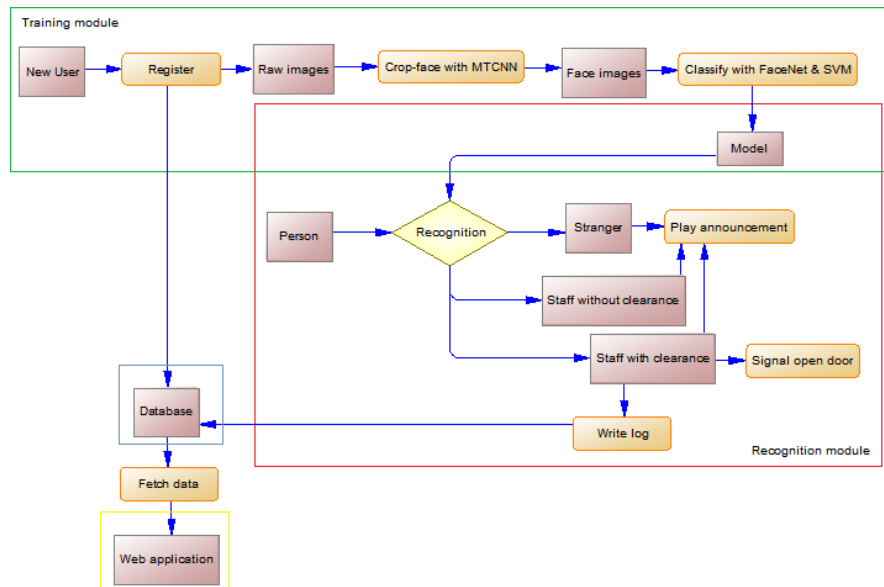


Fig. 3. System design

This system design consists of 4 main parts:

- ***Training module:*** This part handles new user registration, taking raw images, preprocessing them which outputs face images to train into the recognition model.
- ***Recognition module:*** Using the trained model, this part detects and recognizes faces from live-acquired images then performs accordingly
- ***Database:*** Storing personal data from users, as well as other necessary information for the Django web application to run.
- ***Web application:*** Providing users and admin a place to view their personal information and logs from the database, as well as performing some other tasks depending on permission.

In the rest of this section, we will go into detail about each part, explaining the components involved in them, how they operate together to reach their own goal, and the general workflow when the system is running.

2.1 Training module



Fig. 4. Training module design

The training module is the first part of the system, which is usually installed in the administrator's office. Here new employees can register into the system by inputting their ID, name, and other required information, which are input into the database and used to create a class name for the classification model later.

The next step is to take raw face images from staff. By default, the system will take up to 50 images, one shot every 5 frames (by default), unless terminated prematurely by the administrator. During this process, the staff is advised to move their face around in front of the camera so it can take pictures from multiple angles. The 5 skipped frames by default is to give time for the staff to move their face, preventing multiple similar images. The more various face angles captured in the images, the better the data is. The images then will be saved into a created folder with a name generated from the staff's name and ID.

The administrator can repeat this registration step for multiple staff members. Until everyone is registered, the image preprocessing and model training can begin, using the technologies that are discussed in the previous chapter. This results in a model, which is uploaded to the server.

In the training process, however, with new data and a previously trained model, instead of having to retrain the entire dataset over again, a tweak was made to the original SVM to reuse the classified data in the model so it can reduce the training time for the new data. An option to retrain the entire model is still available, just in case.

2.2 Recognition module

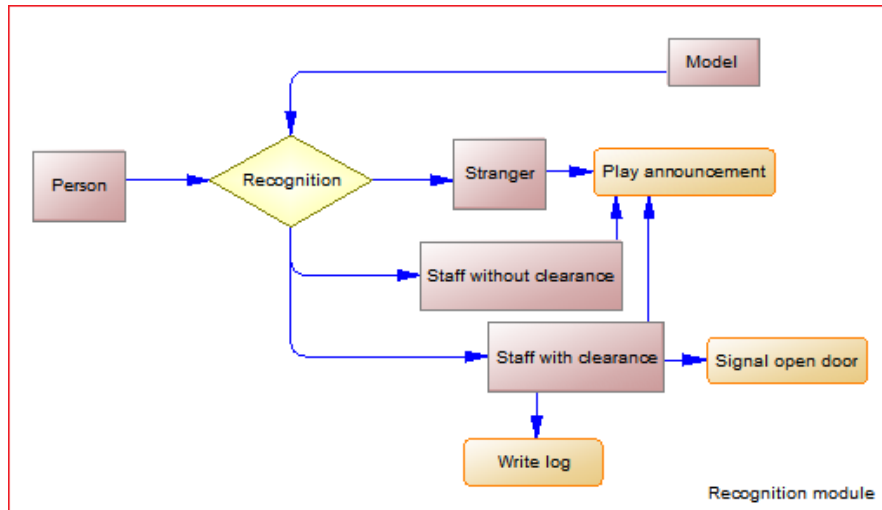


Fig. 5. Recognition module design

This module is installed on every door of the system. It consists of the following components: an LED, a camera, and a speaker, all connected to a Raspberry Pi, which has to connect to the same network as the server's.

When the system is running, the camera will open and look for faces in front of it. The live acquired images are constantly streamed to the server, where the face detection and recognition computations take place. This process uses the model which is the result of the training module mentioned in the previous section. Once at least a face is detected in the frame, if the person is a staff and has clearance to the area, a log is automatically written into the database with the staff's identity, current timestamp, in/out status, and door's identity.

At the same time, the log's data will also be written down into a local text file, which will be read by the API view and returns the data in form of JSON. After reading and returning the JSON to API view, the local text file will be reset.

The Raspberry Pi is programmed to constantly access the API view, if it reads a log data, the speaker will announce their name and the LED will be lit if they have clearance to enter.

2.3 Database

In order to support running the system on multiple doors at once, besides the two tables Staff and Log to save data mentioned previously, a table Gate is also created to store every gate's name, IP address, and voice option.

In addition to that, several tables are also generated automatically by the Django framework to support its web application, which will be discussed later.

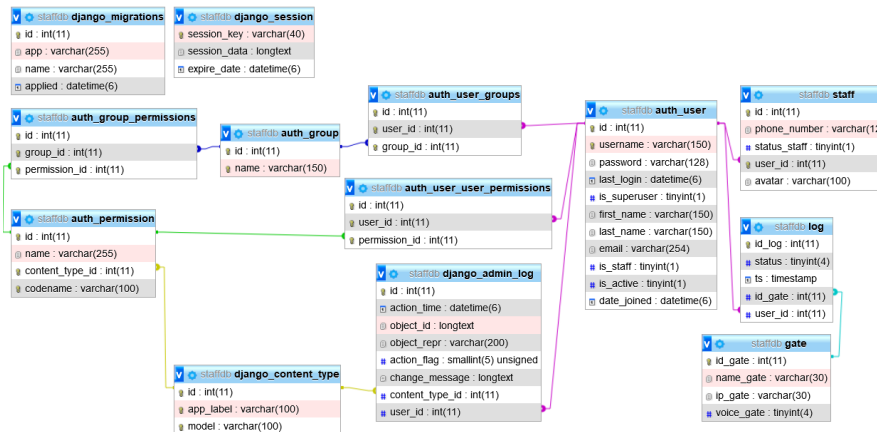


Fig. 6. System database diagram

In total, our system has 13 tables and 12 relational references. MySQL is used to store all the data, and the database's design can be seen in Figure 6 above.

2.4 Web application

Once the previous three modules work, a requirement from the admin to access the database for management and administrative tasks has arisen, as well as staff users also want access the database to view their own entering/leaving logs and personal information. To solve this requirement, a web application is created so that admin and users can access from their device's browser directly without downloading any extra applications.

When a staff registers into the system (the process in section 3.1), an account of this web application is also created with their ID and a default password, which can be changed later after login. An admin account was already created when developing the application, extra admin accounts can be created if the need arises.

Because Django's authentication is used, the Staff table described in Figure 3.4 has a one-to-one relationship with Django's generated Auth_user table. Its purpose is to extend Django's default User model so more related information like phone number, clearance, and avatar can be stored.

By fetching data from the database described in section 3.3, the web application can render and display plenty of useful information as well as providing the user with some functions they can use.

For admin, this application allows them to manage all users and gates, including create, update, view, delete. They can also view and delete log records from everyone, but not editing them. New admin accounts can also be added.

For users, this application allows them to view and edit their personal information except for their ID, change their account password, and only view their own log records. They can also upload an avatar that will be displayed when they log in, the image will be saved in folder *media*.

3 Results

Following the design as discussed in section 2, with the technology and knowledge introduced in section 1, we have succeeded in developing and implementing a complete hardware and software system in Cantho University Software Center. The developed system has been tested for many live acquired images and the results are satisfactory for prototype work, with requirements that are set out in the starting of section 2. Testing conditions and results are discussed in this section.

The computer is the brain of the system, which processes acquired images, analyzes images, and determines the person's name. It also acts as a server for the web application. The computer used in the test is a typical laptop with the following specifications:

- Operating System: Windows 10 Pro 64-bit (10.0, Build 18363)
- Processor: Intel® Core™ i7-6500U CPU @ 2.50GHz (4CPUs), ~2.6GHz
- Memory: 8192MB RAM

3.1 Training module

The registration procedure starts by letting the user enter their name, id, and other information. This data is saved into the database and used to create a class name for the classification model later.

The next step is taking raw images. By default, the system takes 50 raw images in size 640x480, which are saved in folder *DataSet\FaceData\raw\<fullname>_<id>* with *<fullname>* and *<id>* taken from user input. For each staff, this step takes about 1 minute at max, which can be quickened by reducing the number of frames it skips in between shots (as explained in section 2.1).

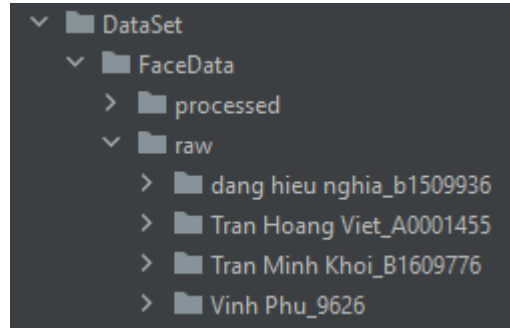


Fig. 7. List of folders for each user's raw images

This registration step can happen at any point of time and multiple times, even when the system is running live since it only uploads information from user to database and saves taken raw images in the server's folder.

Once all the staffs' raw images are taken, the admin can decide to preprocess and train the data at any time, preferably when the system is not running.

Image preprocessing is handled by MTCNN, it detects the face area and crops it, resizing to 160x160 and save in folder *DataSet\FaceData\processed\<fullname>_<id>* (similar structure as Figure 7, but for folder processed instead of raw). For an example of 50 raw images, this step takes about 30 seconds.

3.2 Recognition module

The recognition procedure starts by loading the model and opening the camera(s). This process takes about 30 seconds to a minute, depending on the computer's CPU, GPU (unavailable in this testing computer), and memory. Once opened, the camera(s) will stream live acquired images back to the computer, which will be presented on screen in the separated window(s).

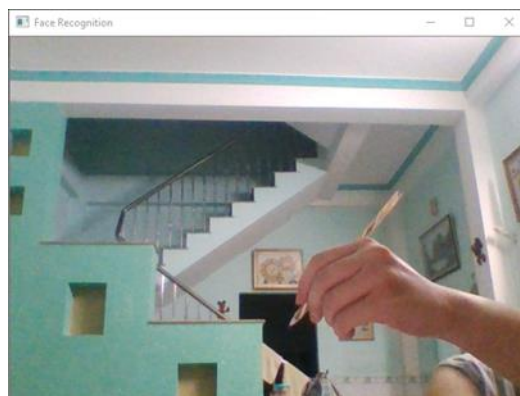


Fig. 8. Live acquired image from the webcam without a face

It only takes 2 seconds max for the system to recognize a face when it's on-screen, but it will only announce the result when the same face is recognized at least 5 times within 50 frames, in order to avoid false recognition. Due to the small number of data collected, it is difficult to fully test the system's accuracy, but it is estimated to be around 80% on the first try for the system to recognize the correct person.



Fig. 9. Live acquired image from the webcam with a recognized face

Once the user's face is recognized like in Figure 9, one of the following three cases happen:

- If they are staff and have clearance to enter, a log record is created with the user's ID, the gate they're seen at, current timestamp, and in/out status depending on the latest log record of the user in the database. Then, this new log record is then inserted into the database while the system writes its content down to a local file for the API view to read, which later will be accessed by the Raspberry Pi (as explained in section 2.2)
- If they are staff but don't have clearance to enter, no log record is created while the system writes down the staff's ID and name to a local file for the API view.
- If they are strangers, no log record is created while the system writes down 'stranger' to the local file for the API view.

Depending on the three previous cases, the API view will read the content of the local file and the Raspberry Pi will act accordingly once accessing it:

- If they are staff and have clearance to enter, the Raspberry Pi will greet/say goodbye (according to the record's in/out status) to the staff's name and light up the LED light.
- If they are staff but don't have clearance to enter, the Raspberry Pi will greet the staff's name and tell them they are not allowed to enter the area.
- If they are strangers, the Raspberry Pi will greet them in general and tell them they are not allowed to enter the area.

3.3 Web application

For users, after logging in the web application, the first page user sees is their logs, with the timestamp sorted in descending order.

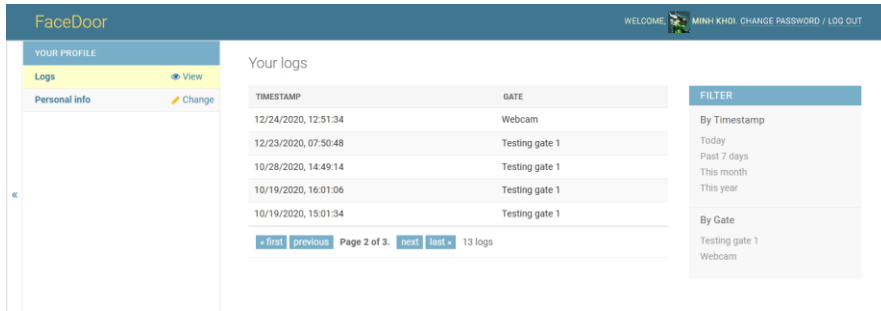


Fig. 10. User Logs view

By clicking on the link on the left side, user can access their personal info page, which displays the user's first name, last name, email, phone number, and avatar that can be changed if they want.

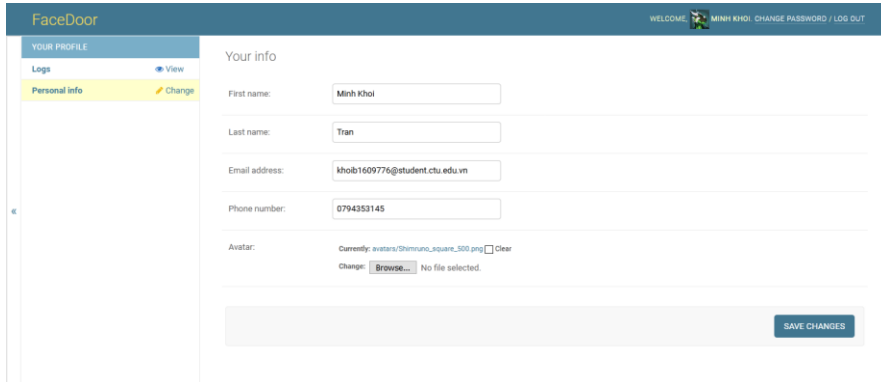


Fig. 11. User Info view

For admin, they can log in the web application to do a variety of management tasks that will be described in the following.

On the Users page, the admin can view the list of current users with summarized details and filters as in Figure 11.

The screenshot shows the 'FaceDoor Administration' interface for user management. At the top, there's a navigation bar with 'Home - Authentication and Authorization - Users'. Below this, there's a search bar and a table of users. The table has columns for Username, Email Address, First Name, Last Name, and Staff Status. There are 5 users listed. To the right of the table, there are filter options for 'By staff status' and 'By superuser status', each with 'All', 'Yes', and 'No' options. At the bottom left, it says '5 users'.

USERNAME	EMAIL ADDRESS	FIRST NAME	LAST NAME	STAFF STATUS
9626	example1@email.com	Vinh Phu		○
A0001455	example@email.com	Hoang Viet	Tran	○
admin	tmkhoi98@yahoo.com			●
b1509936	nghiab1509936@student.ctu.edu.vn	Hieu Nghia	Dang	○
B1609776	khoib1609776@student.ctu.edu.vn	Minh Khoi	Tran	○

Fig. 12. Admin users management

When clicking on a user, the admin can view more information about them in details, and edit each of them, including their account's password.

The screenshot shows the 'FaceDoor Administration' interface for editing a user. The page title is 'Change user'. On the left, there's a sidebar with 'AUTHENTICATION AND AUTHORIZATION' and 'FACEDOOR' sections. The main content area has a 'Change user' form. The form includes fields for Username (B1609776), Password (algorithm: pbkdf2_sha256 iterations: 216000 salt: ESAmLw***** hash: VhYpJ*****), Personal info (First name: Minh Khoi, Last name: Tran, Email address: khoib1609776@student.ctu.edu.vn), and Additional info (Phone number: 0794353145). There are also checkboxes for 'Allowed in?' and 'Avatar'.

Fig. 13. Admin user edit

On the Gates page, since each Gate has less information than each User, its listing table is also simpler.

<input type="checkbox"/>	GATE NAME	GATE IP	VOICE
<input type="checkbox"/>	Webcam	0	⊘
<input type="checkbox"/>	Testing gate 2	http://192.168.10.67:8081	⊘
<input type="checkbox"/>	Testing gate 1	http://192.168.10.69:8081	✔

Fig. 14. Admin gates management

When clicking on a gate, the admin can view and edit its information.

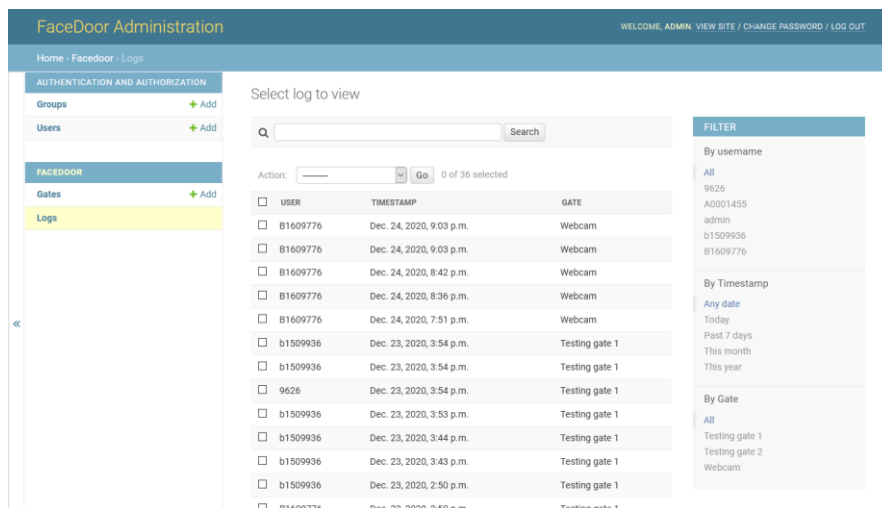
Gate name:

Gate IP:

Voice

Fig. 15. Admin gate edit

On the Logs page, the admin can view all the log records available and filter them base on many options on the right side. Notice that with log records admin can only view or delete, editing them is not an option.



The screenshot shows the 'FaceDoor Administration' interface. The main content area is titled 'Select log to view' and contains a search bar and a table of log records. The table has three columns: USER, TIMESTAMP, and GATE. The log records are as follows:

USER	TIMESTAMP	GATE
<input type="checkbox"/> B1609776	Dec. 24, 2020, 9:03 p.m.	Webcam
<input type="checkbox"/> B1609776	Dec. 24, 2020, 9:03 p.m.	Webcam
<input type="checkbox"/> B1609776	Dec. 24, 2020, 8:42 p.m.	Webcam
<input type="checkbox"/> B1609776	Dec. 24, 2020, 8:36 p.m.	Webcam
<input type="checkbox"/> B1609776	Dec. 24, 2020, 7:51 p.m.	Webcam
<input type="checkbox"/> b1509936	Dec. 23, 2020, 3:54 p.m.	Testing gate 1
<input type="checkbox"/> b1509936	Dec. 23, 2020, 3:54 p.m.	Testing gate 1
<input type="checkbox"/> 9626	Dec. 23, 2020, 3:54 p.m.	Testing gate 1
<input type="checkbox"/> b1509936	Dec. 23, 2020, 3:53 p.m.	Testing gate 1
<input type="checkbox"/> b1509936	Dec. 23, 2020, 3:44 p.m.	Testing gate 1
<input type="checkbox"/> b1509936	Dec. 23, 2020, 3:43 p.m.	Testing gate 1
<input type="checkbox"/> b1509936	Dec. 23, 2020, 2:50 p.m.	Testing gate 1
<input type="checkbox"/> B1609776	Dec. 23, 2020, 2:50 a.m.	Testino oaste 1

On the right side, there is a 'FILTER' section with three categories: 'By username' (All, 9626, A0001455, admin, b1509936, B1609776), 'By Timestamp' (Any date, Today, Past 7 days, This month, This year), and 'By Gate' (All, Testing gate 1, Testing gate 2, Webcam).

Fig. 16. Admin logs management

4 Conclusions

The main theme of this study is to provide solutions to address some of the challenges that arise in developing and deploying a system to monitor employees entering and leaving the office using deep learning. In this section, we describe the main contributions and discuss some further research directions.

4.1 Summary of Contributions

After researching deep learning in general as well as face recognition in particular, referencing many other articles and examples that had tackled this problem before, we have managed to build a system to monitor employees entering and leaving the office in my own set of conditions. The system's benefits are described as following:

- With the currently complicated Covid-19 situation, practicing social distancing is crucial, with this system, employees can check attendance without directly interacting with other people or touching any surface.
- The system can take images continuously for multiple employees without interruption in between any two.

- The system can detect people's faces from a distance, ideally within three meters from the camera for better image quality, which will provide better recognition results.
- The system can perform tasks automatically like playing announcements accordingly, opening doors, and logging events without any further input.
- User can access their own log history from anywhere, as long as they are connected to the server's network, they can view when and where they entered or left the area.
- Admin can manage various information on the system, especially controlling the users' permission to enter the area and enabling a door's announcement.
- Admin can use the log information from the system to monitor staff's performance at work as a reference to evaluate their KPI.
- The logs that are saved in the database can be accessed at any point in the future if a situation requires them.

4.2 Future Directions

The system is designed, implemented, and tested. Test results show that the system has acceptable performance. On the other hand, the system has some future works for improvements and extended functionality.

Using another model implementation. Facenet using Keras instead of Tensorflow and Similarity learning supports training only new user data without retraining the entire dataset again or interrupting the system, which reduces the training time.

Extending to other field applications. This system can be applied to other applications like at parking lot where you can automatically check-in and out; or in the dormitory where you can monitor the students entering or leaving the area and some other management tasks.

Including body temperature check. By integrating this system with a thermal camera, it can be very useful in the current pandemic situation where it can check the person's body temperature in addition to their face before letting them enter.

Acknowledgements. We would like to express our sincere gratitude to the College of Information and Communications Technology for assisting us with technical expertise, as well as Can Tho University Software Center for facilitating our work with equipment and testing environment. Without their support, carrying out this research would not have been possible.

References

1. S. Sawhney, K. Kacker, S. Jain, S. N. Singh and R. Garg, "Real-Time Smart Attendance System using Face Recognition Techniques," *2019 9th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, pp. 522-525, doi: 10.1109/CONFLUENCE.2019.8776934, 2019.
2. S. W. Arachchilage and E. Izquierdo, "A Framework for Real-Time Face-Recognition," *2019 IEEE Visual Communications and Image Processing (VCIP)*, Sydney, Australia, pp. 1-4, doi: 10.1109/VCIP47243.2019.8965805, 2019.
3. M. Khan, S. Chakraborty, R. Astya and S. Khepra, "Face Detection and Recognition Using OpenCV," *2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, Greater Noida, India, pp. 116-119, doi: 10.1109/ICCCIS48478.2019.8974493, 2019.
4. S. Saha, "A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way," 16 December 2018. [Online]. Available: <https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53>. [Accessed 2 January 2021].
5. K. Zhang, Z. Zhang, Z. Li and Y. Qiao, "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499-1503, doi: 10.1109/LSP.2016.2603342, October 2016.
6. F. Schroff, D. Kalenichenko and J. Philbin, "FaceNet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, pp. 815-823, doi: 10.1109/CVPR.2015.7298682, 2015.
7. D. Sandberg, "Face recognition using Tensorflow," 17 April 2018. [Online]. Available: <https://github.com/davidsandberg/facenet>. [Accessed 2 January 2021].