An Automatic Face Attendance Checking System using Deep Facial Recognition Technique

Thuy Nguyen-Chinh, Thien Do-Tieu, Sy Nguyen-Tan, Phuong Le-Van-Hoang, Qui Nguyen-Van, Phu Nguyen-Tan

***Abstract*—Nowadays, as computers are powerful enough for implementing complex algorithms, there are numerous applications that people utilize computers to run. In which, facial recognition is one of the most active fields of applications. In fact, computers can not only automatically identify who a person is, but also operate 24/7, which human beings cannot endure. This leads to the replacement of people by computers in some repetitive and real-time applications.**

In this work, we apply the facial recognition into an attendance checking system that uses faces of registered people to check their attendance. This system has a GUI which allows easy user-to-system interaction. The core of the system is a deep facial recognition technique, which has four stages (e.g., removing motion-blur frames, detecting faces, removing non-frontal-view faces, and recognizing). Particularly, in the recognition phase, we consider this stage as an open-set facial recognition problem, so the system is able to detect people who have not registered in the database before. Also, we boost the performance of the system by utilizing hardware resources of users' computers. Although the system is designed to run with a low-resolution webcam, its performance is reasonably accurate on our private dataset.

***Index Terms*—Face Attendance Checking, Facial Recognition, Deep Learning**

I. Introduction

Face recognition systems are being applied widely in real life such as tracking, managing employees, finding information about celebrities, and so forth. There are many approaches to design a face recognition system, but such these systems are frequently affected by light, non-frontal faces, resolution of cameras, etc. Thus, each method has specific challenges. Overall, a face recognition has two main stages which are face detection and face recognition, yet we want to create constraints on blur-clean and frontal-view faces of users that leads our system to have four stages: blur detection, face detection, landmark detection, and face recognition.

A. Face detection

Before face recognition stage, we need to detect faces in a image and bound the high-level regions to eliminate effects, such as hair, background. The face detector proposed by Viola and Jones [1] used Haar-Like features and AdaBoost algorithm to train cascaded classifiers, which achieve good performance with real-time efficiency. However, a few works [2],[3],[4] show that Haar-Like feature may degrade considerably in real-world applications with larger visual variations of human faces even with more advanced features and more training images. Besides the cascade model, [5],[6],[7] introduce deformable part models (DPM) for face detection and achieve remarkable performance. However, they need a high computational expense and may usually require expensive annotation in the training stage. With the rise of data-model, convolutional neural networks (CNN) achieve considerable accuracy in the number of computer vision tasks, especially face detection task. Li et al. [8] use cascaded CNN for face detection, but it requires bounding box calibration from face detection with an extra computational expense and ignores the correlation between facial landmarks localization and bounding box regression. Face alignment also attracts extensive interests. Regression-based methods [9],[10],[11] and template fitting methods [7],[12],[13] are two popular approaches.

However, most of the available face detection and face alignment methods ignore the correlation between these two tasks. Though there exist several works attempt to jointly solve them, there are still limitations in these works. For example, Chen et al. [14] jointly show alignment and detection with random forest using features of pixel value difference. But, the handcraft features used limits its performance. From those previous experiments, we choose a new approach which integrates these two tasks using unified cascaded CNN by multi-task learning called Multi-task Convolutional Network which is also face detector used in FaceNet model in section III-D.

B. Landmark detection

The locations of the facial landmark points around facial components (nose, eye centers, etc) capture facial deformations due to head movements and facial expressions, so we will control non-frontal faces for better accuracy. They are hence important for various facial analysis tasks. Many facial landmark detection algorithms have been developed to automatically detect those key points over the years. In this paper, we use Dlib library which is a powerfull opened source for face and facial landmark detection. We will discuss our implement in detail in section III-C.

C. Face recognition

After face detection and alignment, those regions of a face are extracted to get feature vectors. Conventionally, one of the

This work is a final project in the course "Artificial Intelligence in Control Engineering" (Aug-Dec 2018) guided by Dr. Cuong Pham-Viet (email: pvcuong@hcmut.edu.vn), Faculty of Electrical and Electronics Engineering, HoChiMinh city University of Technology.

Authors are senior students in the Faculty of Electrical and Electronics Engineering, University of Technology, HoChiMinh city Vietnam National University (e-mail: {thuy.ng.ch, dotieuthien9997, tansyab1, hpcqt97, nvqui97, tanphu97.nguyen}@gmail.com). The software is open source and can be found in https://github.com/AntiAegis/Face-Attendance-System.

most popular features for face recognition is Gabor feature. Tudor Barbu [15] used Gabor transform to extract features, and then applied K-Nearest Neighbour (K-NN) based on clustering features to predict identity of a face. This implementation achieved a quite impressed performance with an accuracy of 90% on the Yale Face Database B [22]. OpenCV [16], which is an open-source library focusing on Computer Vision algorithms, introduced a method called Local Binary Patterns (LBP) based on Haar-Like feature. In term of speed, LBP has a remarkably real-time efficiency, whereas it is not stable in term of accuracy. In particular, this method cannot face noise which is a reason why LBP and Haar-Like feature are rarely applied in practical systems. Because of limitations of conventional features, approaches exploiting deep learning have gradually developed instead. Yi Sun, Xiaogang Wang, and Xiaoou Tang [17] built a deep model named Deep hidden IDentity (DeepID) which used Convolutional Neural Network (CNN) to extract face features. The advantage of this method was using a small dataset for training that was considerably relevant for applications which cannot collect a large dataset of users. Nevertheless, to reach a high accuracy, DeepID model became really complex with many neural network branches for each person. There were 10 patches of a face, which contain interesting information, chosen from each image, then they were scaled with three figures in RGB and gray channel. Totally, the model had 60 different networks to extract features for an image, then feedforwarded extracted features into a classifier using Joint Bayesian. DeepID achieved an excellent accuracy of 97.45% on Labeled Faces in the Wild (LFW) [23]. In 2014, the authors of DeepID introduced DeepID2 [18] which was an improved version of its predecessor. In the new version, interesting regions were built to eliminate useless patches which cannot provide high-level features. That work really helpfully affected the accuracy, specifically there was an increase in accuracy at 99.63%. In 2015, Google Inc. [19] exploited a deep CNN Inception [20] and a triplet loss function to form the FaceNet model, which was to extract features. Their outstanding point was using a hard triplet loss to separate features for each person, so FaceNet feature is robust in both face verification and face recognition. The accuracy of 99.63% on LFW and 95.12% on Youtube Faces DB [24] were high enough to represent the perfection of the model.

II. Proposed system

In this paper, we apply deep facial recognition techniques to the problem of face attendance checking. A system is built in order to manage appearances of students in a class, which is revealed in Fig. 1. Normally, a facial recognition system is organized as a pipeline of typical stages such as face detection, landmark detection, and face recognition. However, to ensure input frames for underlying algorithms are high quality, we append an early filter (the Blur detection stage) that are able to discard blur frames, which are caught by motions of people in front of a webcam. Then, clean frames are passed through the Face detection block for counting the number of faces existing inside these frames. In our specific constraint, there is only one face per frame allowed to be processed, so frames

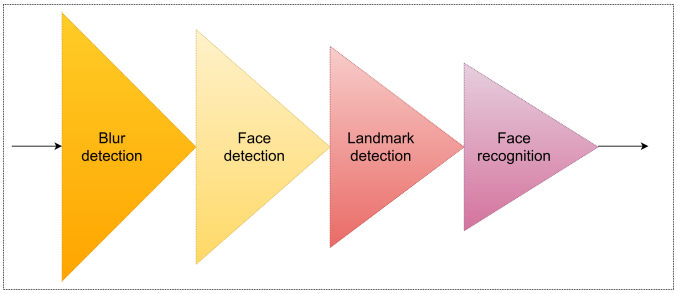


Fig. 1: The system pipeline. There are four stages, namely blur detection, face detection, landmark detection, and face recognition. These four blocks are in the descending order of size in the direction from input to output. This points out that our system is tougher to input frames from the camera when such frames passed through the system. Therefore, best frames are likely to be processed, which may improve the final recognition accuracy.

that contain more than one faces are rejected. Afterward, the Landmark detection is to localize statistic-salient points in the face in order to verify whether the face is in the frontal view of the camera. This frontal-view check will help improve the accuracy of the face recognition algorithm. Finally, the Face recognition uses blur-clean, one-face, and frontal-view frames from previous stages to extract relevant features and then perform a classification task to indicate which category the input most likely belong to.

In addition, to leverage the ease of use, we design a friendly graphic user interface (GUI) so that people who want to use the system to manage (teachers) or check (students) attendance can interact with the application without any specialized knowledge. To make the system more robust, we carefully analyze the distribution outlier of features representing for registered accounts. Therefore, the algorithm has ability to detect people who have not registered in the application before, which is equivalent to the open-set problem in face recognition.

Our work is organized as follows. In section III, stages of the proposed system are described clearly, including motion-blur detection, face detection, frontal-view detection, and face recognition. Then, section IV is for reporting some experimental results.

III. Implementation

A. Motion-blur detection

The first stage of this system is detecting blurred image and rejecting them out of next stage. We know that the blurred image means each pixel in the source image gets spread out and mixed into surrounding neighbour pixels. For our attendance checking system, the motion blur happens when an object (namely face or webcam) moves during the exposure. So as to detect whether an image is blurred, we use the 2D-FFT (2D-Fast Fourier Transform) method.

We will review about Fourier Transform of Images. To compute the Fourier transform of an image, you need to:

- Compute DFT of each row, in place.
- Compute DFT of each column, in place.

When a signal is discrete and periodic, we use the discrete Fourier transform, or DFT. Suppose our signal is a_n for $n =$

$0 \dots N-1$, and $a_n = a_{n+jN}$ for all n and j. The spectrum of a is:

$$A_k = \sum_{n=0}^{N-1} W_N^{kn} a_n \tag{1}$$

where

$$W_N = e^{-i\frac{2\pi}{N}}$$

and W_N^k for $k = 0 \dots N-1$ are called the *Nth roots of unity*. The sequence A_k is the discrete Fourier transform of the sequence a_n. Each is a sequence of N complex numbers.

The FFT is a fast algorithm for computing the DFT. If we take the 2-point DFT and 4-point DFT and generalize them to 8-point, 16-point, ..., 2^n-point (n is an integer), we get the FFT algorithm.

There are several ways to write an FFT. For instance, let m be an integer and let $N = 2^m$. Suppose that $x = [x_0, \dots, x_{N-1}]$ is an N dimensional complex vector. Let $\omega = \exp(\frac{-2\pi i}{N})$. Then the DFT, $c = F_N(x)$ is given by

$$c_k = \frac{1}{N} \sum_{j=0}^{j=N-1} x_j \omega^{jk}. \tag{2}$$

Let $n = N/2$, let u and v be n dimensional vectors defined by

$$u_j = x_j + x_{j+n}, \ j = 0, \dots, n-1 \tag{3}$$

$$v_j = (x_j - x_{j+n})\omega^j, \ j = 0, \dots, n-1. \tag{4}$$

Then

$$c_{2j} = \frac{1}{2}(F_n(u))_j, \ j = 0, \dots, n-1 \tag{5}$$

$$c_{2j+1} = \frac{1}{2}(F_n(v))_j, \ j = 0, \dots, n-1. \tag{6}$$

To compute the DFT of an N-point sequence using equation (1) would take (N^2) multiplies and adds. The FFT algorithm computes the DFT using $(N \log N)$ multiplies and adds.

Practical issues: We translate the picture so that pixel (0,0), which now contains frequency $(\omega_x, \omega_y) = (0, 0)$, moves to the center of the image. Then, we display pixel values proportional to log(magnitude) of each complex number. For color images, do the above to each of the three channels (R, G, and B) independently.

Apply to our system, firstly, we calculate FFT of image. Secondly, we will compute mean amplitude spectrum value of entire pixel in image and. Finally, the result of this operation is compared to an optimal threshold which distinguishes blurred and non-blurred image as accurate as possible. The image is called non-blurred if and only if its average value greater than the threshold value, and vice versa. After that, non-blurred images are applied to face detection stage of system.

B. Face detection

In this paper, we have used Histogram of Oriented Gradients method for extracting features of the face and Linear Support Vector Machine (SVM) method for face detections.
The implementation of this method using sliding window technique with the different sizes of the windows. Using the sliding window technique we could complete the calculation of HOG features, applied to detect and differentiate the face and the false face recognition using the SVM technique.
All of the pre-processing steps are automatically implemented before using Dlib library with the input of being given facial images and the output of localization the identified faces.

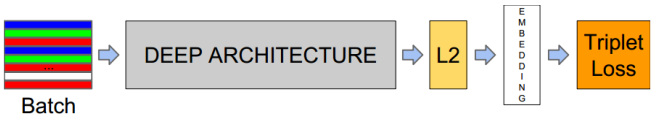


Fig. 2: Model structure. Our network consists of a batch input layer and a deep CNN followed by L2 normalization, which results in the face embedding. This is followed by the triplet loss during training.

C. Frontal-view detection

To check whether the shape of the faces has to be frontal, we implement these 3 following steps: Step 1.

1) Focusing on the center of the image. Only accept these faces that locate in the most central of the images. $((120-360), (90, 360))$
2) Identify the skew of the image: calculate the coordinate of these eyes and the angle deviation between two eyes in the horizontal direction. If the angle deviation is larger than 10 degree, the image will be ignored.
3) Identify the rotation of the image: choose the point which is the midpoint of the right and the left eye. If the nose which is deviated from the selected point is greater than 10 pixels in the horizontal direction, the image is ignored.

These steps are implemented based on 5-point facial landmark technique with Dlib library instead of 68-point facial landmark in order to improve performance. If the image satisfies the condition, it will be accepted.

D. Face recognition

In this stage, faces in raw images are detected and aligned by Multi-task CNN, we use convenient pre-trained FaceNet model to extract feature (in Figure 2) and then feedforward it to a SVM classifier for regconition.

1) Multi-task Convolution Network: The overall pipeline Multi-task CNN is shown in Figure 3. An image is initially resized to different scales to build an image pyramid, which is the input of the following three-stage cascaded framework with CNN architectures in Figure 4:

Stage 1: A fully convolutional network is exploited, called Proposal Network (P-Net), to obtain the proposed windows and their bounding box regression vectors. Then using the estimated bounding box regression vectors to calibrate the candidates. After that, employing non-maximum suppression (NMS) to merge highly overlapped candidates.

For each candidate window, P-CNN predict the offset between it and the nearest ground truth (i.e.,the bounding boxes left top, height, and width). The learning objective is

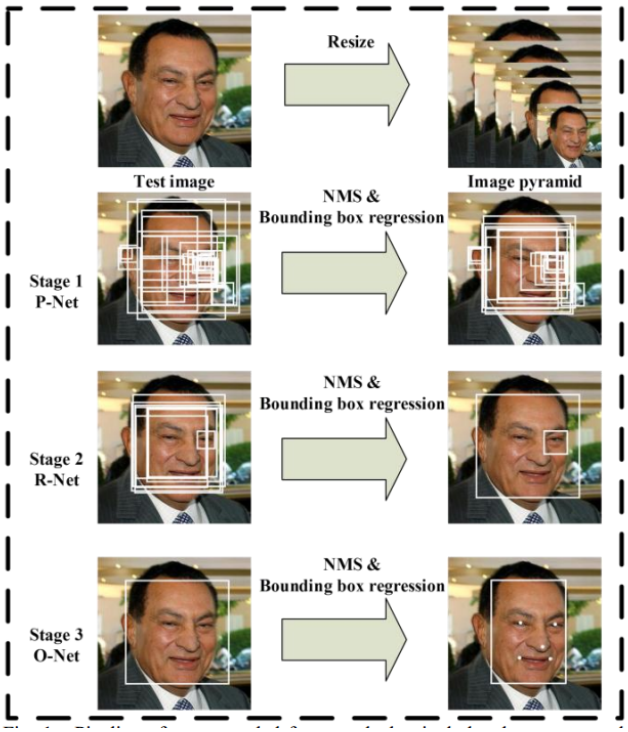


Fig. 3: Pipeline of our cascaded framework that includes three-stage multi-task deep convolutional networks. Firstly, candidate windows are produced through a fast Proposal Network (P-Net). After that, we refine these candidates in the next stage through a Refinement Network (R-Net). In the third stage, The Output Network (O-Net) produces final bounding box and facial landmarks position.

formulated as a regression problem, and the Euclidean loss is employed for each sample x_i:

$$L_i^{box} = \|y_i^{prediction} - y_i^{truth}\|_2^2 \tag{7}$$

Stage 2: All candidates are fed to following CNN, called Refine Network (R-Net), which further rejects a large number of false candidates, performs calibration with bounding box regression, and NMS candidate merge.

Stage 3: his stage is similar to the second stage, but in this stage we aim to describe the face in more details. In particular, the network will output five facial landmarks positions.

Similar to the bounding box regression task, facial landmark detection is formulated as a regression problem:

$$L_i^{landmark} = \|y_i^{prediction} - y_i^{truth}\|_2^2 \tag{8}$$

2) FaceNet model: This model use Inception-ResNet v1 architecture [20] and triplet loss to extract feature. Inception-ResNet v1 (in Figure 6) is a very deep convolutional network which combine ResNet network and Inception network with a complex structure. This deep network affords to extract high-level feature for object recognition, and combination with triplet loss gets better and better.

The embedding is represented by $f(x) \in R^d$. It embeds an image x into a d-dimensional Euclidean space. Here we want to ensure that an image x_i^a (anchor) of a specific person is closer to all other images x_i^p (positive) of the same person

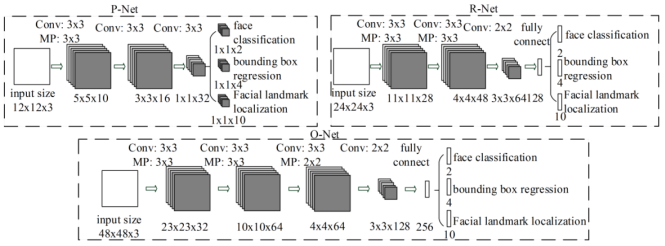


Fig. 4: The architectures of P-Net, R-Net, and O-Net, where MP means max pooling and Conv means convolution. The step size in convolution and pooling is 1 and 2, respectively

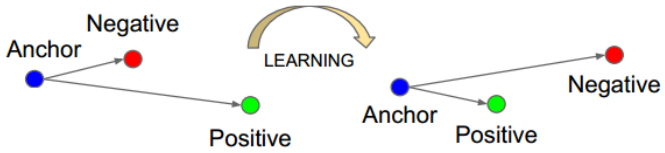


Fig. 5: The Triplet Loss minimizes the distance between an anchor and a positive, both of which have the same identity, and maximizes the distance between the anchor and a negative of a different identity.

than it is to any image x_i^n (negative) of any other person. This is visualized in Figure 5. Thus we want,

$$\|f(x_i^a) - f(x_i^p)\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2 \tag{9}$$

where α is a margin that is enforced between positive and negative pairs. The loss that is being minimized is then:

$$L = \sum_i^N (\|f(x_i^a) - f(x_i^p)\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha) \tag{10}$$

In this implement, FaceNet model is trained on VGGFace2 dataset which over 9000 identities and over 3.3 million faces. VGGFace2 is a large-scale face recognition dataset from Google Image Search and have large variations in pose, age, illumination, ethnicity and profession. Therefore, embeddings is specifical for each person.

3) Training: To apply Multi-task model and FaceNet model into Face Attandence system, we feedforward raw images of students into Multi-task CNN to get face patches, then extract feature of each patch with 512 dimension vector by pre-trained FaceNet model [21]. After that, we split dataset of students into 3 subsets: training, validating and testing. Each person in training, validating, testing subset contains 30 images, 10 images and 30 images respectively. We decide to collect only 30 images for training subset, because we want to reduce time of training and time for collecting images. Especially, we also choose 4069 people who have only 1 image in LFW dataset [23] for testing subset, because we want to evaluate the final accuracy of the whole algorithm. Next, we use SVM classifier with linear kernel to train on trainning subset and validate on validating subset. The formula of linear kernel:

$$probability = AX + b = W^T X^{bar} \tag{11}$$

where W is weight matrix, $X^{bar} = [1, x_1, x_2, ...x_n]$. To solve multi-class classification problem, loss funtion of multi-class SVM is different from binary SVM. The output of trained-classifier is probability vector whose each element represent

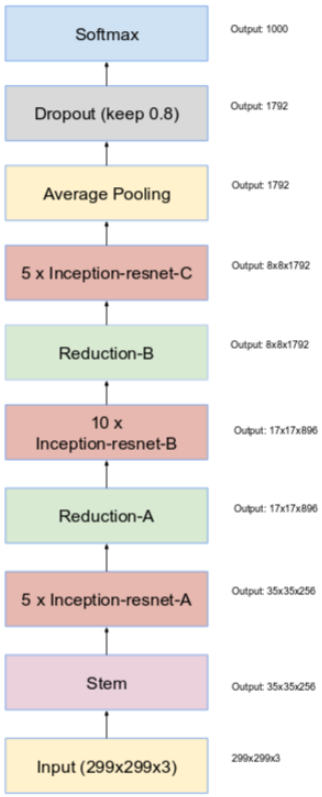


Fig. 6: Schema for Inception-ResNet-v1 and Inception-ResNet-v2 networks. This schema applies to both networks but the underlying components differ.

ability of an identity who anchor belong to. We assume x_i is embedding i^{th}, z_i is probability score of true label of x_i, z_j $(j \neq i)$ is probability scores of other classes of x_i and Δ is safe distance to ensure z_j is not much close with z_i. Loss value of x_i with m classes is defined:

$$loss = \sum_{j=1}^{m}(max(0, z_j - (z_i - \Delta)) \tag{12}$$

In dataset which has n embeddings, total loss value is:

$$loss^{total} = \sum_{i=1}^{n}\sum_{j=1}^{m}(max(0, z_j - (z_i - \Delta)) \tag{13}$$

This loss function will make z_i is maximum. Thus, anchor is determined by choose which identity has maximum probability, that suffer from mistakes when faces of strangers appear. To solve open-set problem, we combine both two subsets: training and validating to training threshold. This threshold helps us to eliminate "unknow" people, additionally ensure that the result of system achieves higher accuracy.

E. Graphic User Interface

The sections that follow describe how to create GUIs with PyQt5. This includes laying out the components, programming them to do specific things in response to user actions, and saving and launching the GUI; in other words, the mechanics of creating GUIs.

PyQt5 implements GUIs as figure windows containing various styles of uicontrol objects. You must program each object to perform the intended action when activated by the user of the GUI. In addition, you must be able to save graphical user interface development environment.

GUI Development Environment

The process of implementing a GUI involves two basic tasks:

- Laying out the GUI components
- Programming the GUI components

GUIDE primarily is a set of layout tools. However, you must create a PY-file that contains code to handle the initialization and launching of the GUI. This PY-file provides a framework for the implementation of the callbacks - the functions that execute and launch your GUI.

The Implementation of A GUI

Laying out the GUI components

While it is possible to write an PY-file that contains all the commands to lay out a GUI, it is easier to use GUIDE to lay out the components interactively and to generate one file that save the GUI:

UI-file -contains a complete description of the GUI figure and all of its children (uicontrols and axes), as well as the values of all object properties.

Programming the GUI components

Once you have the UI-file, you will use PyQt5 to extract the UI-file to PY-file to execute the program.PY-file -contains the functions that launch and control the GUI and the callbacks, which are defined as sub functions. This PY-file is referred to as the application PY-file in this documentation. Note that the application PY-file does not contain the code that lays out the uicontrols; this information is saved in the UI-file. The following diagram illustrates the parts of a GUI implementation.

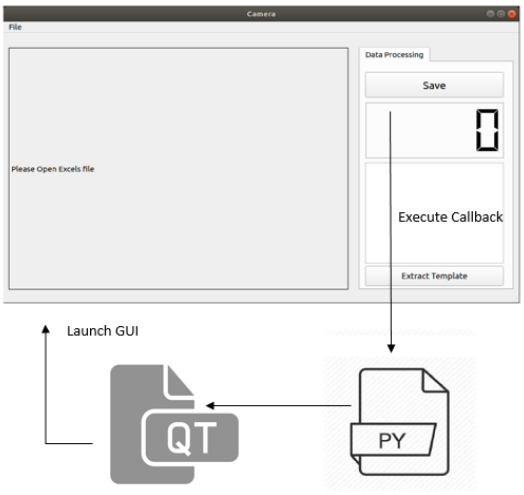


Fig. 7: Parts of GUI Implementation.

The GUI are available from the QtDesigner shown in the figure below. The design is described briefly below. Subsequent sections show you how to use them.

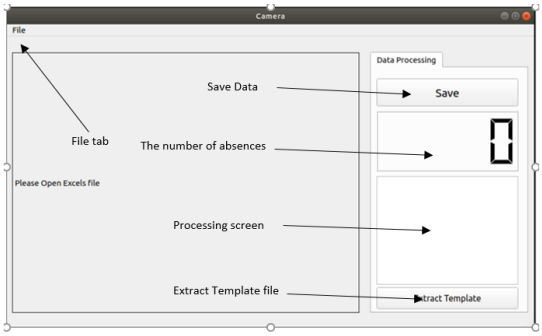


Fig. 8: GUI layout.

uicontrols	function
Open File Option	We use this option to open the existing file Open option appears and you can choose to open the file.
Start Camera	Open camera to start checking process
Stop Camera	Stop camera
Exit	We use this option to close the program. we can follow the steps: Click on File tab >Exit, active file will be closed. When we close the file, we get the confirmation message to save the file or not or cancel the command.
Save	Store completely-checked student IDs on the document. Once program exits, this process will be automatically started so that you wont lose your work that has been completed if there is a power interruption or other system malfunction, the first time you Save, it will take you to the Save File dialog box.
Extract Template	We use this option to export the template in XLSX document. To Extract the file, we can follow the steps: Click Extract Template . And then we can export it as per our requirement.

TABLE I. USER GUIDE

F. Attendance management

This is a module in the GUI. It was designated to mark the presence of one resulted from our algorithm in a file of excel format, namely xlsx extension. To be used by the system, the excel file must meet a stringent format made up of essential contents and be generated by the GUI.

Fig. 9a depicts a new standard empty excel table generated by our GUI. After obtaining a new file, we should fill in the table with the desired data (Fig. 9b). The most special things in this table are column ID and Total. ID is considered a primary key because the algorithm will mark the presence of a specific person via his ID. To help the host in easy attendance management, we designed the column Total with a view to showing the number of absences in all.

Fig. 9c depicts an excel file's content after a checking progress finished. The GUI will automatically insert the only one new day column between Group and Total ones and in the tail of previous checked day. Letter 1 will be marked as presence in a cell of this column accordant to an ID. After attendance checking process is completed, the Total column will display the number of absences of previous days and the current one. Smartly can it display as we specially assigned a size-dynamic sum function to each cell of this column.

STUDENT LIST
YOUR COURSE/SUBJECT/TITLE
1 = prensent
blank = absent

ID	Last Name	First Name	Group	Total

(a) New standard excel form

STUDENT LIST
Artificial Intelligence
1 = present
0 = absent

ID	Last Name	First Name	Group	Total
1511844	Lương Hữu Phú	Lộc	1	
1512221	Phạm Ngọc Khôi	Nguyên	1	
1512396	Bùi Tấn	Phát	1	
1512534	Nguyễn Trọng	Phúc	1	

(b) Excel form contain pre-inputed data

STUDENT LIST
Artificial Intelligence
1 = present
0 = absent

ID	Last Name	First Name	Group	11/18/2018	Total
1511844	Lương Hữu Phú	Lộc	1		1
1512221	Phạm Ngọc Khôi	Nguyên	1	1	0
1512396	Bùi Tấn	Phát	1		1
1512534	Nguyễn Trọng	Phúc	1	1	0

(c) Form is under checking

Fig. 9: Attendance Management module.

IV. Experimental result

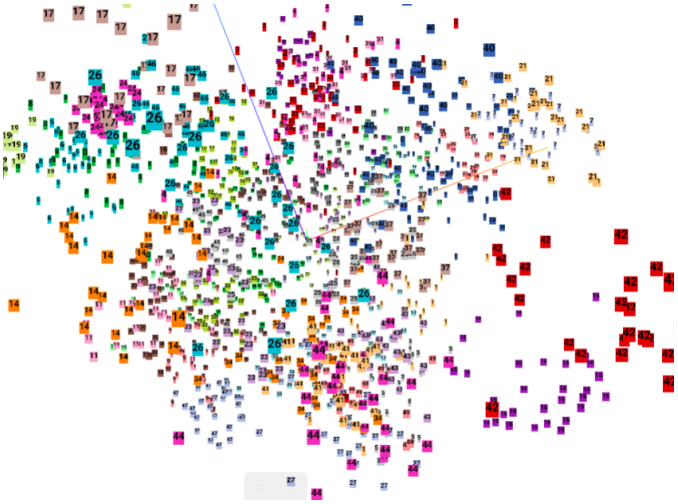
In this section, we first evaluate the effectiveness of the features extracted from FaceNet. Then, we conduct experiments to point out accuracies of the model on datasets (e.g., training, validating, and testing). Regarding the test dataset, there are two subsets taken into account, namely known and unknown ones. The former is from our private data, while the latter is collected from the LFW dataset [23]. By using only the former, we evaluate the model on a closed set (only containing registered identities), but by combining both two ones, we have an open set (containing not only registered identities but also unregistered identities).

TABLE II. DATASET SUMMARY

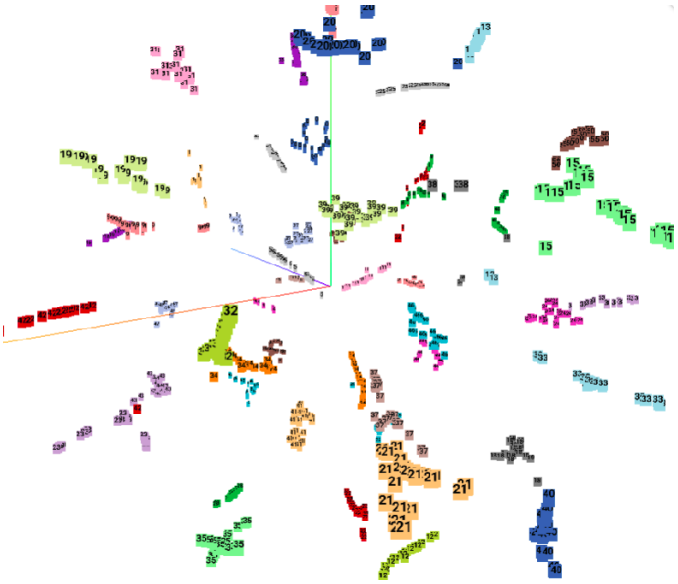
Dataset	Subset	#identities	#images per identity	#images totally
Training	-	52	30	1560
Validating	-	52	10	520
Testing	Known (1)	52	20	1040
	Unknown (2)	4069	1	4069

A. Feature embeddings

As mentioned above, features representing for faces play a crucial role in a robust face recognition algorithm. A good kind of feature is a way of representing input data so that data points belonging to a same category are close together, whereas, those ones which are different from their categories need to be as far as possible. Therefore, before training the model, we start evaluating how separate samples of different classes stand in a vector space.



(a) Visualization using PCA.



(b) Visualization using t-SNE.

Fig. 10: 128-dimensional embeddings of the training set are projected to the 3D space using PCA (a) and t-SNE (b) techniques. In general, although samples of different classes are not totally separated, the PCA method partly reveals the separation of classes. Meanwhile, the t-SNE method provides a better view of classes as distinguished clusters. Best viewed in color.

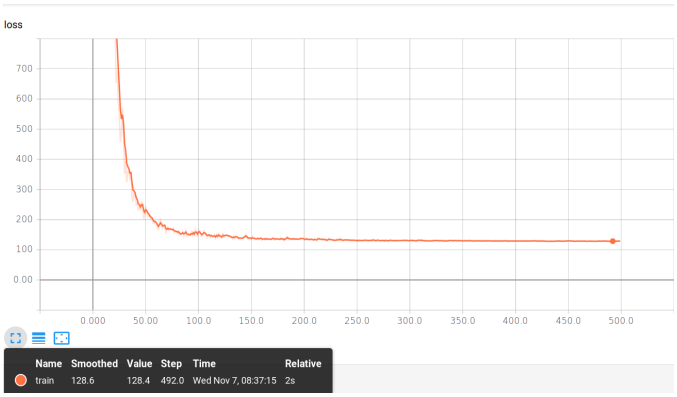
In this work, we utilize the pre-trained model FaceNet because of its spectacular demonstration over the recent period. Embedding or the feature vector of FaceNet has size of 128. Therefore, in a 128-dimensional space, embeddings of samples should form clusters, in which, a cluster is a set of samples in a same class. Nevertheless, the 128-dimensional space is beyond human's vision and awareness. Therefore, to have a prior look on these embeddings, we use visualization methods so that embeddings are converted into a new vector space, which human can understand.

Here we exploit the Principal Component Analysis (PCA) and t-SNE [25] to complete this task. The two methods project embeddings from the original size into the 3D space. The result of projection is depicted in Fig. 10. In particular, PCA visualization selects eigenvectors corresponding to the highest eigenvalues as basics for constructing a new 3D space. Thus, its result in Fig. 10a, although, illustrates that data points are slightly separated between distinct classes, this separation is still not clear. The reason is non-selected (residual) eigenvectors still convey a significant amount of information, so by removing them, there is more spatial information is lost. Meanwhile, in Fig. 10b, the separation illustration is picturesque. Inter-class samples are closed while intra-class samples are far from each other. This is because the t-SNE method includes a clustering stage, so embeddings totally belong to their classes. If embeddings of FaceNet do not contain high-level specification, dimension reduction algorithms cannot show or cluster embeddings properly. Consequently, embeddings of the pre-trained FaceNet are sufficient for us to train a classifier for indicating the most likely class that an input belongs to, so we do not conduct a further finetuning stage to adapt the pre-trained FaceNet model in order to be compatible to our dataset.

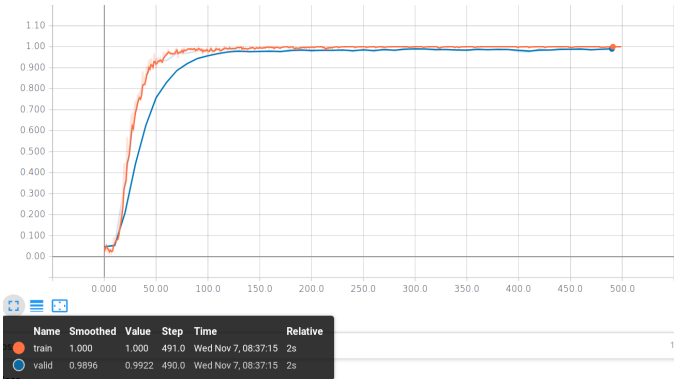
B. Model evaluation

Training data is carefully collected with different views from $-70°$ to $70°$. This work can improve accuracy in pratical system ,because it is difficult for users to keep their faces in a correct position. Training data include 52 identities and 1560 images totally. The training accuracy and the validating accuracy of SVM classifier is 100% and 99.22% respectively (in Figure 11b). In Figure 11a, the convergence of loss can indicate that this model achieves a good performance without overfitting. Training data is carefully selected with different views from $-70°$ to $70°$. This work can improve accuracy in practical system, because it is difficult for users to keep their faces in a correct position. Therefore, by choosing faces with view variations, the model learns to generalize from training data. The training set includes 52 identities (people), in which, each identity consists of 30 images with various poses. This results in the training set of 1560 images totally. The training accuracy and validating accuracy of SVM classifier is 100% and 99.22% respectively (in Figure 11b). In Figure 11a, the convergence of loss can indicate that this model achieves a good performance without overfitting.

We try to implement multi-class SVM by using API of sklearn library, the training accuracy of 100% and the validating accuracy of 99.36% are slightly the same in comparison with the above SVM model. After training classifier, we train to get the best threshold to determine "unknown" people. We divide threshold in range $[0; 1]$ and then select a maximum value which make model achieves the best accuracy. Particularly, the model ignores people who have the scores being lower than threshold, and we combine the training subset and the validating subset for training threshold. Our dataset of students is collected once a semester, so training threshold with only training subset cause overfitting and difficult recognizing for reality system. As a result, threshold for 52 identities is 0.18825 in Figure 12. Testing accuracy with threshold achieves 96.48% on testing subset, which is shown in Table III. In practical environment, we test on 32 identities, there



(a) Loss



(b) Accuracy

Fig. 11: Loss and accuracy during the training phase.

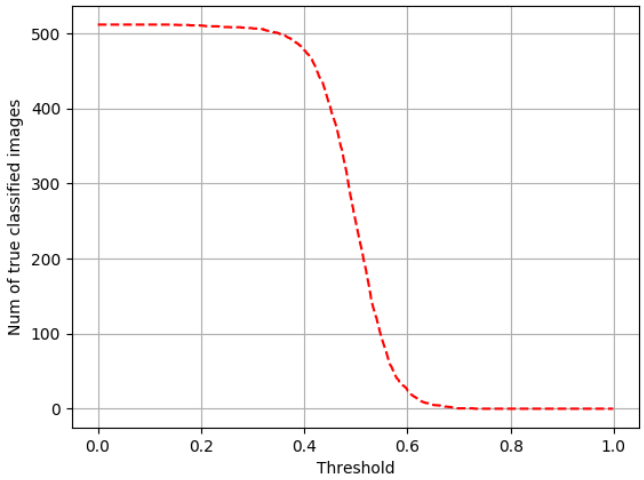
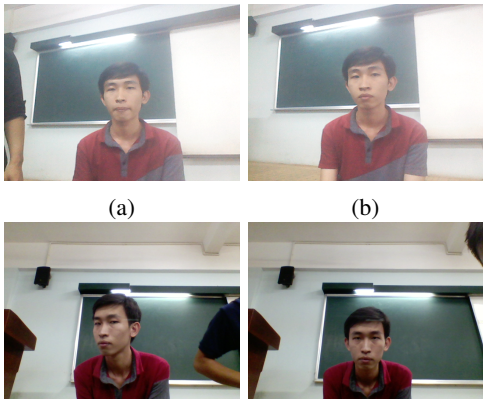


Fig. 12: Training threshold stage, the final threshold is 0.18825

are 29 easily recognized identites and 3 identities who are not recognized continuously. In Figure 13, the (a),(b) are training images and (c),(b) are testing images, the effect of different ilumination lead the probabilities of testing anchor are lower than threshold, so training data have to cover many real-life cases to create the best classifier.



(a) (b)

(c) (d)

Fig. 13: Unrecognized identity, (a),(b) are training images, and (c),(d) are testing images in pratical condition.

TABLE III. ACCURACIES AMONG DATASETS

Dataset	Training	Validating	Testing (1) (Closed set)	Testing (1+2) (Open set)
#images	1560	520	1040	5109
#identities	52	52	52	4121
Accuracy	100%	99.36%	98.85%	96.48%

V. Conclusion

In this work, we applied the deep facial recognition techniques to solve the problem of face attendance checking. The system has a pipeline with four stages (e.g., motion-blur detection, face detection, landmark detection, and face recognition). Besides, the system is also integrated a friendly GUI, which allows users both teachers and students interact with it in an easy way. On our private dataset, the application performs accurately despite the low-resolution webcam on typical laptops. This demonstrates that our underlying algorithm is effective to deal with this poor-quality input problem.

In the future, we will target to widen our dataset so that the dataset can be asymptotic to real applications. In addition, more algorithms will be considered to improve the ability of the algorithm to discriminate feature distributions of output classes.

Acknowledgment

The authors would like to acknowledge Dr. Cuong Pham-Viet for providing documents as well as promoting the chance for us to do this work. Also, the authors would like to thank volunteer students in the course EE3063 (Aug-Dec 2018, HoChiMinh city University of Technology), who donated data for us to conduct this work.

References

[1] B. Yang, J. Yan, Z. Lei, and S. Z. Li, *"Aggregate channel eatures for multi-view face detection"*, IEEE International Joint Conference on Biometrics, pp. 1-8, 2014.

[2] P. Viola and M. J. Jones, *"Robust real-time face detection. International journal of computer vision"*, vol. 57, no. 2, pp. 137-154, 2004.

[3] M. T. Pham, Y. Gao, V. D. D. Hoang, and T. J. Cham, *"Fast polygonal integration and its application in extending haar-like features to improve object detection"*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 942-949, 2010.

[4] Q. Zhu, M. C. Yeh, K. T. Cheng, and S. Avidan, *"Fast human detection using a cascade of histograms of oriented gradients"*, IEEE Computer Conference on Computer Vision and Pattern Recognition, pp. 1491-1498, 2006.

[5] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, *"Face detection without bells and whistles"*, European Conference on Computer Vision, pp. 720-735, 2014.

[6] J. Yan, Z. Lei, L. Wen, and S. Li, *"The fastest deformable part model for object detection"*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 2497-2504, 2014.

[7] X. Zhu, and D. Ramanan, *"Face detection, pose estimation, and landmark localization in the wild"*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 2879-2886, 2012.

[8] H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, *"A convolutional neural network cascade for face detection"*, IEEE Conference on Computer Vision and Pattern Recognition, pp. 5325-5334, 2015.

[9] X. P. Burgos-Artizzu, P. Perona, and P. Dollar, *"Robust face landmark estimation under occlusion"*, IEEE International Conference on Computer Vision, pp. 1513-1520, 2013.

[10] X. Cao, Y. Wei, F. Wen, and J. Sun, *"Face alignment by explicit shape regression"*, International Journal of Computer Vision, vol 107, no. 2, pp. 177-190, 2012.

[11] J. Zhang, S. Shan, M. Kan, and X. Chen, *"Coarse-to-fine auto-encoder networks (CFAN) for real-time face alignment"*, European Conference on Computer Vision, pp. 1-16, 2014.

[12] T. F. Cootes, G. J. Edwards, and C. J. Taylor, *"Active appearance models"*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 6, pp. 681-685, 2001.

[13] X. Yu, J. Huang, S. Zhang, W. Yan, and D. Metaxas, *"ose-free facial landmark fitting via optimized part mixtures and cascaded deformable shape model"*, IEEE International Conference on Computer Vision, pp. 1944-1951, 2013.

[14] D. Chen, S. Ren, Y. Wei, X. Cao, and J. Sun, *"Joint cascade face detection and alignment"*, European Conference on Computer Vision, pp. 109-122, 2014.

[15] T. Barbu, *"Gabor filter-based face recognition techniquet"*, http://www.acad.ro/sectii2002/proceedings/doc2010-3/12-Barbu.pdf, 2008.

[16] Opencv library, Haar-Like Cascade, https://docs.opencv.org/3.4.1/d7/d8b/tutorial_py_face_detection.html.

[17] Y. Sun, X. Wang, X Tang, *"Deep Learning Face Representation from Predicting 10,000 Classes"*, http://mmlab.ie.cuhk.edu.hk/pdf/YiSun_CVPR14.pdf.

[18] Y. Sun, X. Wang, X. Tang, *"Deep Learning Face Representation by Joint Identification-Verification"*, https://arxiv.org/abs/1406.4773, 2014.

[19] F. Schroff, D. Kalenichenko, J. Philbin, *"FaceNet: A Unified Embedding for Face Recognition and Clustering"*, https://arxiv.org/pdf/1503.03832.pdf, 2015.

[20] C. Szegedy, S. Ioffe, V. Vanhoucke, *"Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning"*, https://arxiv.org/pdf/1602.07261v1.pdf, 2016.

[21] *"Pre-trained model FaceNet*, https://github.com/davidsandberg/facenet, 2018.

[22] A.S. Georghiades, P.N. Belhumeur, and D.J. Kriegman, *"From Few to Many: Illumination Cone Models for Face Recognition under Variable Lighting and Pose"*, IEEE Transaction on Pattern Analysis and Machine Intelligence, vol. 23, no. 6, pp. 643-660, 2001.

[23] E.L. Miller, G.B. Huang, A. RoyChowdhury, H. Li, and G. Hua, *"Labeled Faces in the Wild: A Survey"*, In Advances in Face Detection and Facial Image Analysis, Springer, pp. 189-248, 2016.

[24] L. Wolf, T. Hassner, and I. Maoz, *"Face Recognition in Unconstrained Videos with Matched Background Similarity"*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2011.

[25] Laurens van der Maaten, Geoffrey Hinton, *"Visualizing Data using t-SNE"*, Journal of Machine Learning Research 9 (2008) 2579-260.