APPLICATIONS OF MACHINE LEARNING TECHNICAL FOR VERIFICATION OF STUDENT IDENTITY

Nguyen Thanh Dat[1], Ngo Dinh An, Pham Quoc Hung, Truong Trong Tien

[1]datntqe170110@fpt.edu.vn, FPT University, Quy Nhon

Abstract—*The application of technology in education is becoming an effective tool to improve the quality and efficiency in management and teaching. However, every development has certain positives and negatives, so it is necessary to have supporting tools to minimize the negatives. Currently, at the FPT Quy Nhon campus, each student is issued an ID card to verify his/her identity at studying and gain access to the exam room. Over time, ID cards can become faded or facial features can change, and even with current technology, it is not difficult to forge an ID card. So, it cannot also expel the impersonation of the exams (impersonation in exams is the act of one student taking an exam on behalf of another) and the need for 2 proctors to authenticate in the exam room. With the development of artificial intelligence, facial recognition technology for identity verification has made significant progress through the integration of multiple different techniques. Given the aforementioned issues, our team decided to have developed a web service model to support facial recognition for student identity verification using CNN networks and YOLOv8 models. The goal of the project is to prevent impersonation in exams at the FPT Quy Nhon campus by scanning faces through mobile devices. The project utilizes a data input approach that detects faces extracted from a mobile camera, and then extracts features, processes, matches, and identifies the faces. The output is displayed as a notification on the application screen.*

Keywords—Identification, Facial recognition, CNN, YOLOv8, FaceNet, MTCNN

I. INTRODUCTION

In the education system, it is crucial to ensure the integrity of examinations. The first aspect is verifying the identity of students in the exam room in order to prevent impersonation and maintain a fair academic environment. Traditional verification methods, such as identification cards' visual inspection have several drawbacks, such as being time-consuming, laborious, or not ensuring foolproof security. With the rapid development of data science and advancement in artificial intelligence, in which face recognition is a technology widely applied in people's daily life such as surveillance systems at buildings, airports, ATMs, time attendance systems, anti-theft cameras, identity authentication, etc.

Currently, face recognition methods are divided into many directions and methods according to different criteria to improve performance. However, we realize that more or less these methods are facing difficulties and challenges such as brightness, tilt orientation, image size, or the influence of environmental parameters. A solution our team proposed to support identity verification in the exam room is that we have built a system to automatically search candidates from the image database through acquisition from many surveillance cameras.

Our team has started to study previously developed techniques such as artificial neural networks, hidden Markov models for geometric feature matching and pattern matching. Artificial neural networks and pattern matching have high accuracy but require a relatively large database to train a model for an object,

so the results will not be high if our system does not have enough data about the object to be identified. Facial feature-based recognition is a method based on determining the geometrical features of facial details such as position, area, distance of objects on the face, and the relationship between them (*eg. distance between eyes*). Geometric feature-based methods involve the examination of facial attributes and their spatial arrangements to analyze the relationships between different facial components. This is also known as a feature-based method. In this way, it helps us to solve difficult problems in the context that the image details on the face have similar overall characteristics. But the disadvantage of this method is that the implementation of the algorithm is complicated due to the determination of the relationship between the features, and it will not be effective when it is not possible to distinguish the features when the image size is small. We consider the image as a vector in multi-dimensional space, then we build a new space for the image with a reduced number of dimensions without losing the features of the face image. When reducing the data dimension from D to K<D, we only keep the K most important elements, but the problem is that we do not know which components are more important. For example, in the worst case scenario where the amount of information each component is similar, removing any component will result in a large loss of information.

Through the research, we found that two methods are applied for this problem: Principal Components Analysis (*PCA*) and Linear Discriminant Analysis (*LDA*). PCA is a method to find a new basis system so that the information of the data is mainly concentrated in a few coordinates, the rest only carries a small amount of information. And for simplicity in calculation, PCA will find a orthonormal system as a new basis[1]. However, the PCA method is an unsupervised learning method, it only uses the vectors describing the data without using the labels of the data (*if any*). In fact, in classification problems (*the most typical form of supervised learning*), using labels will yield better classification results[2].

PCA is a method widely deployed in many projects with large labeled data sets and external conditions, guarantees the best support for photos and it gives quite high results. Typically the research team of Doan Hong Quan [1] applied the CNN model (*altered the parameters and added convolutional layers*) with test results ranging from 91.46% to 99.63%. Another approach is based on edge maps (*calculating the matching will be based on Hausdorff distance and compared with the Eigenface method*), but the results are better within lighting conditions, angles, and facial expressions [2]. The solutions such as reducing the size of the face compared to the original [5], and changing parameters with edge map techniques [4], provide significantly better output results than other techniques in error conditions such as lighting conditions, posture variation, and facial expression [6][10].

To solve problems where collecting enough good data (*quantity and quality of images*) is an impossible task, Siamese Neural Network (SNN) is an architecture targeted by researchers. SNN is a neural network architecture that contains two or more identical subnets to find similarities with input data by comparing their feature vectors. And the Facenet technique is developed based on SNN so that the smaller the distance

[1] https://vi.wikipedia.org/ (Internet access: 01/05/2023)

[2] https://vi.wikipedia.org/(Internet access: 10/05/2023)

between the embedding vectors is, the greater the similarity between them is. FaceNet[3] is a solution that depends less on objective factors and provides better results than the Eigenface method. Currently, FaceNet is interesting in the research community thanks to the superior ability of this technique.

This study focuses on exploring applications of machine learning techniques to verify student identity in the exam room. By utilizing the ability of advanced techniques in the machine learning field, especially deep learning, we seek to improve the precision, efficiency, and reliability of the verification process. Four problems we need to solve: image extraction from video, face detection, feature extraction, and face recognition.

In this paper, we present the obtained results from our research and application. The paper's structure includes the following main sections: Section 1 briefly discusses the problem of identity verification and the background of face recognition, as well as the reason why we chose this topic. Section 2 presents the research's purposes and objectives. Section 3 introduces more specifically the system description, the role of each system's block, and the algorithms applied. Section 4 presents the experiment and the obtained outcome. Section 5 is the desired results and the development orientation of the system. Section 6 summarizes the above content and concludes.

II. THE AIM AND RESEARCH OBJECTIVES

The purpose of the project is to build identity recognition for FPT university students in the Web environment. To complete this target, our team members focus on solving the following issues:

- Collect students' data with suitable quality (*number of images per subject, time for this task, and objective image quality*) to train the model.
- Building a system for face recognition.
- Deploy and test the application for identification with the small dataset.

III. SYSTEM DESCRIPTION

Face recognition is currently one of the closest applications of Computer Vision. To build a system, we propose a model with the input data of a video frame extracted directly from the camera, then self-match with the existing faces in the database (*or create a new database file for the object*), and the output is to verify the identity of the person in the video. The face recognition process takes place completely automatically without human intervention.

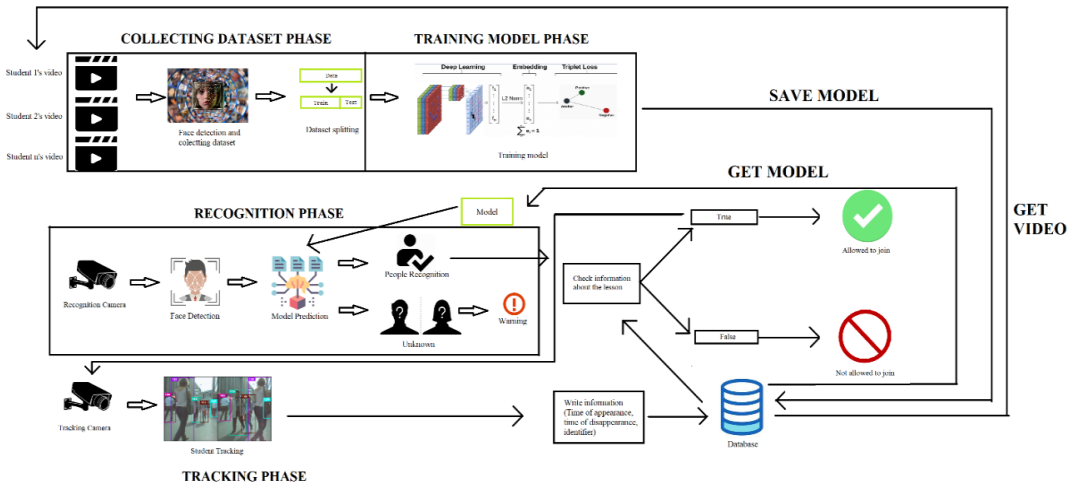


Fig. 1 Working principle of the face recognition system

This system helps us to proactively detect faces in a discreet manner by taking pictures of the faces of subjects entering a defined area from the surveillance camera without subject interaction and is performed quickly. Therefore, the target audience is completely unaware of this process. They do not feel "*under surveillance*" or their privacy is violated. This system we divide into the following parts:

3.1. Collecting dataset

In the data collection phase, we divide into 2 types of collection:

1) Exam objects were asked to provide a self-recorded video of their face for more than 30 seconds. These videos capture their faces with various expressions from multiple angles. (*This is our main data collection source*)

2) We extract the object faces from the camera when it appears in the observation area. (*This is our source of additional data as needed*)

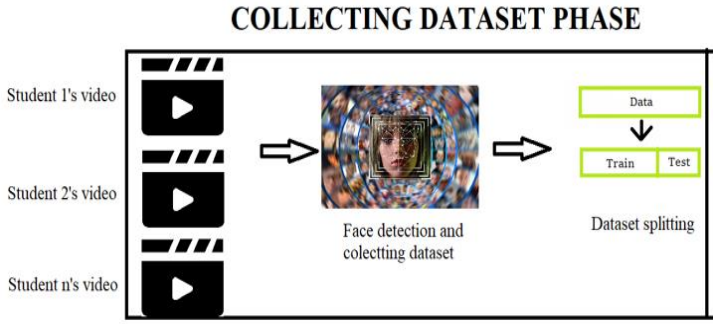


Fig. 2 The data collection process

The most common frame rate is 24-60 fps (*Frames Per Second*). This number of frames guarantees that the human eye will not perceive the video as a series of images but rather as a "*moving image*"[3]. Based on this parameter, we will determine the size of the time required for a video to extract the right number of frames for the model. The captured camera data is saved into video and then uploaded to the data analysis system. To capture a face image, we use the comparison of the position of the face in two consecutive frames. If a detection at time t is too close to a state already present at time t-1 then it will be considered a duplicated image. That is, we call the distance of 2 frames delta, and we compare 2 consecutive frames if the delta value has not been exceeded, it is called the same frame and vice versa.

To extract the faces images of the objects in the videos, we use the MTCNN algorithm to detect, align the face, crop the frame (*Crop the frame only when face accuracy is above 80%*), and resize it into the model's input size (160,160,3). And after recording it into an image (*.jpg). With each video, we took extracted 300 pictures and saved them in different folders whose labels represent the corresponding objects. The dataset often has a huge influence on the accuracy of the model, so we also pay attention to creating the dataset. Here are some requirements for the videos:

+ Moderate brightness (*not too bright or too dark*) and it should be enough to see the face clearly.
+ Each video corresponds to each object and only that object appears.

[3] https://vi.wikipedia.org/ (Internet access: 10/05/2023)

+ The resolution of each video is 1080p/60fps (*60fps frame rate means 60 frames per second of video*).

+ The length of each video is 30 seconds or more.

+ In the video, objects need to rotate their faces from many angles.

Then we use this dataset to split into Training data and Test data.

3.2. Using MTCNN to detect faces in frames

For face detection, the first step is to detect and identify the features of the face in the frame. In this paper, we propose to use a neural network MTCNN [11] to recognize and align faces on images. The MTCNN model includes three separate networks: the Proposal Network (P-Net), the Refine Network (R-Net), and the Output Network (O-Net).

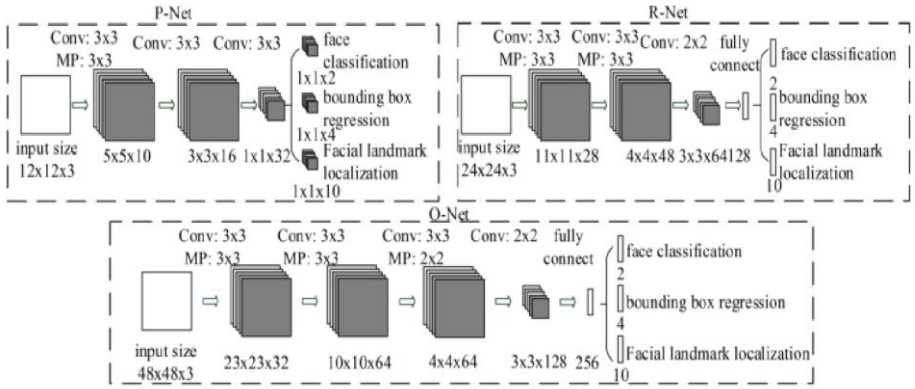
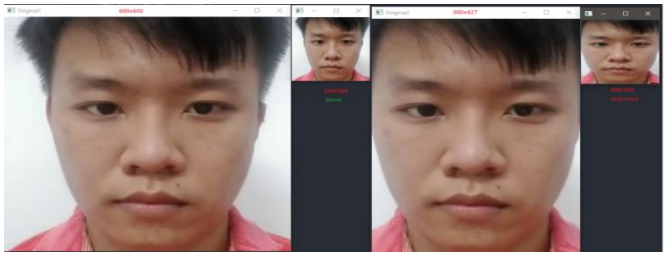


Fig. 3 Architecture of MTCNN[11]

In Fig 3, each CNN network in MTCNN has a different structure and role for face detection. The output data of the MTCNN network is a feature vector representing the location of the face identified in the image (*eyes, nose, mouth, ...*). The P-Net layer is responsible for finding the coordinates of the high-confidence bounding boxes and deleting the low-confidence bounding boxes. And in addition, there will be overlapping bounding boxes, further using NMS (*Non-Maximum Suppression*) technique to reduce this [11]. The R-Net layer uses the output of the P-Net to further refine the coordinates of the bounding boxes to remove areas that are not faces. The O-Net layer uses the output of R-Net to determine the face coordinates in more detail, especially since the network will output five locations of facial landmarks such as 2 eyes, nose, and corner of the mouth.

Once the face position is determined, the original image on the dataset with dimensions of x × y pixels will be made by cropping the face area and bringing it to 160 × 160 size. When using MTCNN to align faces, crop, and resize, a problem occurs that the face proportions are changed or in other words, the faces are distorted. The reason for this problem is that the size of each image of the face when aligned on different frames is different, leading to resizing to the same size makes the face distorted.

We recommend solutions to improve output image quality as follows: When the model detects a face and provides us with a bounding box (*rectangle or square*), if the shape of the image is originally a rectangle, scaling it to 160x160 can result in distortion. We provide a solution to this problem by adjusting the edges to 160 x 160 ratio (*increasing or decreasing the edges of the output image by e - known value in advance*).



After Before

Fig. 4 The output image fine-tuning result is distorted

```
for i, det in enumerate(det_arr):
    det = np.squeeze(det)
    bb = np.zeros(4, dtype=np.int32)
    bb[0] = np.maximum(det[0]-args.margin/2, 0)
    bb[1] = np.maximum(det[1]-args.margin/2, 0)
    bb[2] = np.minimum(det[2]+args.margin/2, img_size[1])
    bb[3] = np.minimum(det[3]+args.margin/2, img_size[0])
    dta1 = bb[3] - bb[1]
    dta2 = bb[2] - bb[0]
    e = abs(dta1 - dta2)
    if dta1 < dta2:
        bb[1] -= e
        bb[3] += e
    else:
        bb[0] -= e
        bb[2] += e
    cropped = img[bb[1]:bb[3] , bb[0]:bb[2], :]
```

3.3. Develop a Face Recognition System

3.3.1. Face Recognition

One of the most significant breakthroughs in facial recognition technology is the development of FaceNet, it effectively solves the hurdles in face detection and verification. In this part, we will explore the FaceNet Inception V3 - developed by Google researchers in 2015 [12]. Inception V3, a Siam network, is a well-known image recognition model that has demonstrated impressive performance, achieving over 78.1% accuracy on the challenging ImageNet dataset [18]. Inception V3 allows deeper networks while also keeping the number of parameters from growing too large (under 25 million parameters) to compute less, reduce errors and help the trained network faster [15]

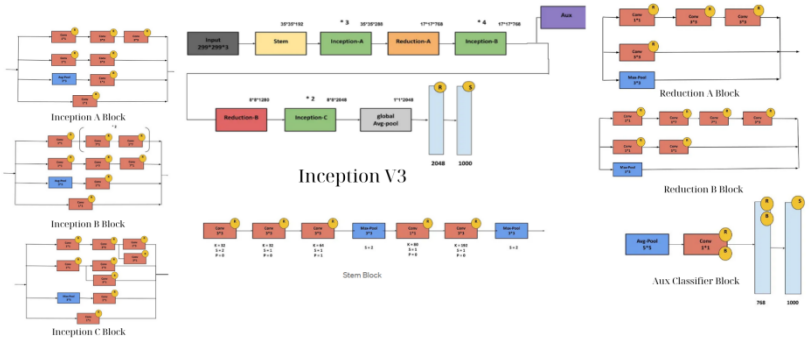


Fig. 5 The InceptionV3 architecture

Our training model with the input dataset (uses a 160x160x3 size face image as input) is to convert the face image into Euclidean space by CNN, then normalize with the output vectors with 128 dimensions for face features and finally use the triplet loss function to create the best embedding vector.

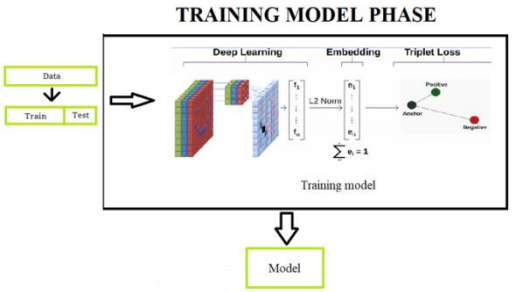


Fig. 6 Training FaceNet Model

The primary breakthrough of FaceNet lies in its adoption of a triplet loss function, enabling the learning of a mapping from face images to a compact Euclidean space. This transformation ensures that distances within this space correspond to a meaningful measure of face similarity. Triplet loss function is a common loss function used in FaceNet to train the neural network. The basic idea behind loss triplet is to learn a mapping from face images to a high-dimensional embedding space where images of the same person are close to each other, while images of different people are far apart.

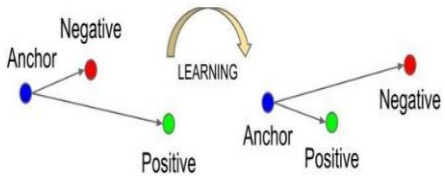


Fig. 7 The Triplet Loss [13]

To achieve this, the Triplet loss requires triplets of images for training. A triplet consists of an anchor image, a positive image, and a negative image. The anchor and positive images are images of the same person (same class), while the negative image is an image of a different person (different class) [14]. The triplet's objective function is the distance between the embedding anchors with a positive smaller than the negative anchor distance. Figure 6 provides a visual representation of the learning process using the triplet loss. It showcases how the triplet loss function is utilized in the model to optimize the embedding of face images into a compact Euclidean space, where the distances between embeddings reflect the degree of similarity between faces.

3.3.2. Face detection

Object detection is a computer vision task focused on identifying specific object instances present in images or videos. The objective is to locate the presence of these objects by enclosing them with bounding boxes, indicating their positions in the visual data. This enables the system to recognize and localize multiple objects of interest simultaneously. For example, with an input image, it generates one or more bounding boxes and attaches labels for it.

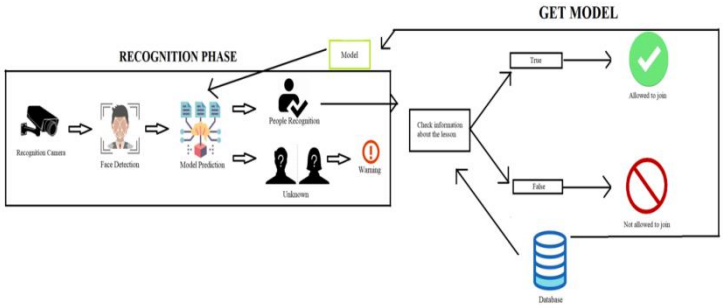


Fig. 8 Face detection

To detect objects, we study 2 techniques MediaPipe [16] and MTCNN to detect the presence and position of objects. Each object detection technique has its own strengths and weaknesses, which need to be carefully considered. Factors such as accuracy, speed, robustness, flexibility, and interpretability vary among the different methods. These differences influence their suitability for specific applications and scenarios, and ongoing research aims to address their limitations and improve their performance in real-world use cases. The following content is our test on a set that is not large enough (200 frames of a student's video) but contributes to the decision of choice in object recognition for each system.

	Mediapipe	**MTCNN**
Time(second)	2.421788454	161.3650258
Detection rate (%)	90	99.5
Average time/frame (only detected) (second)	0.01345438	0.810879526
Average time/ frame (included not detected) (second)	0.012108942	0.806825129

Table. 1 *Compare Mediapipe and MTCNN*

Through the results table, we observed that the MTCNN exhibits higher accuracy and successfully detects all faces in the frames (99.5%), whereas Mediapipe falls shorter in this aspect (90%). However, Mediapipe demonstrates a significantly faster processing speed with only about an average of 0.012 seconds per frame meanwhile MTCNN is about an average of 0.8 seconds per frame (Mediapipe is faster more than 60 times). Considering the requirement for a dataset with high accuracy for model training, we have opted to choose MTCNN. However, during the implementation phase, we encountered a challenge with MTCNN's time-consuming processing, which would lead to overtime when testing a larger number of students. Consequently, we have decided to prioritize Mediapipe due to its faster processing speed, enabling us to efficiently test a larger number of students without incurring excessive delays.

Utilizing Mediapipe to recognize objects, they are entering the examination room and are required to stand before a camera to the system recognize their faces. If an object is identified as being on the object list of the exam, it is permitted to enter the room. In contrast, an object who is either not recognized or identified as not being on the object list is not allowed to enter. The process for conducting the test involves the following steps:

Step 1. Face Detection: Utilizing Mediapipe, faces are detected within each frame of the video. Once detected, the faces are cropped and resized to a standardized size of (160, 160, 3).

Step 2. Model Prediction: The pre-trained model is employed to predict information about the student appearing in each frame.

Step 3. Assessment: For each test, a total of 50 frames are analyzed. The model predicts a student code for each frame. The correctness of the predictions is evaluated by comparing them against the ground truth or expected student code.

Step 4. Acceptance Criteria: If the correct assessment rate exceeds 80% through the 50 frames for a particular student, their student ID is considered accepted and they can proceed to the next step. However, if the accuracy falls below 80%, the student will not be permitted to continue.

Step 5. Database Verification: Once the student ID is accepted, the system checks the database to verify if the student has a scheduled exam at the current time in that room. If the verification is successful, the student is allowed to join the exam. If the verification fails, it indicates that the student does not have an exam scheduled for the day, they will not be allowed to proceed.

And finally, all allowed student IDs will be saved to a temporary file. By following this process, students with a high degree of accurate identification or feature extraction across the frames will be granted access to the exam room, ensuring a reliable and secure examination environment.

3.3.3. Objects tracking

Fig. 9 Tracking Phase

After the system recognizes the object, they continue to interact with the object as the object appears within the camera area. As soon as objects are allowed into the room, using the previously stored temporary file and using the tracking camera to detect objects to get the bounding box of them. The IDs are stored in the temporary file. When an object enters the tracking area, the tracking camera will use the ID stored in the temporary file and assign it to that object. At the same time, the camera will record the object's arrival time and mark the start time in the room. The tracking pane will exist and track the object activities during its time in the room. When an object leaves the monitoring area of the exam room, the tracking camera captures the event and records the time at which the object disappears from its view. This marked the end of its time in the room. At the same time, it records the time entering and leaving the room of objects in the database.

In our study, we used YOLOv8 [17] to perform object tracking (version launched in early 2023). YOLOv8 was built on the success of previous versions . It supports AI tasks: detection, segmentation, pose estimation, tracking, and classification, thereby providing real-time predictions. The YOLOv8 architecture consists of two main parts: the backbone and the head. The backbone of the model can utilize a pre-trained convolutional neural network to extract features and compute feature maps from the input images. The neck connects the backbone and the head, it basically collects feature maps from different stages of the backbone (it’s a feature aggregator). The head section of the model analyzes the concatenated features and generates predictions for bounding boxes, objectness scores, and classification scores.

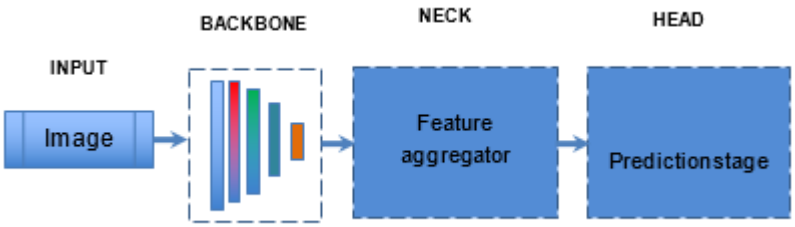


Fig. 10 Model structure of YOLOv8 detection models

IV. EXPERIMENT

We choose the object for the test to be students, the test data set is all students of class AI17B. The student will record their own video and send it to the system administrator. They evaluated the quality and length of each submitted video to ensure they met the required standards (assessed factors such as resolution, clarity, and duration). If a video failed to meet the quality or length criteria, we informed the respective student and asked them to re-record the video. We retrieved the bounding box coordinates for each detected face to secure input to the model and saved them as images into folders named after the corresponding student ID of the student who made the video.

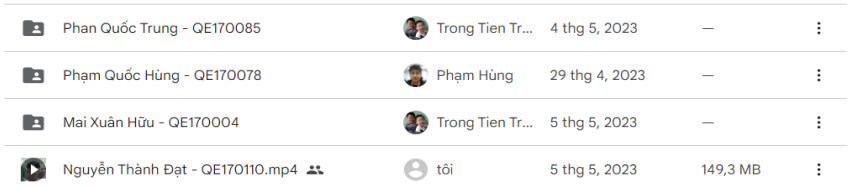


Fig. 11 *Uploaded videos*

We continue to extract the frames from the video using the obtained bounding box coordinates. We resized each frame to the desired size of (160, 160, 3) to ensure consistent input for the model. We save the processed frames as individual images into folders named after the corresponding student ID who recorded the video. Each student's images were stored in his/her respective folder.

Fig. 12 *The data set that stores for an object*

After processing the dataset, we put it into the train model. We positioned a camera in front of the entrance of the inspection room to capture the students' faces as they approached for inspection. We utilized the database to filter out the list of students who had scheduled exams on a specific day and it was stored in temporary memory or a list. Students took turns standing in front of the recognition camera for inspection. and we set a recognition threshold of about 80% out of 50 successful detections.

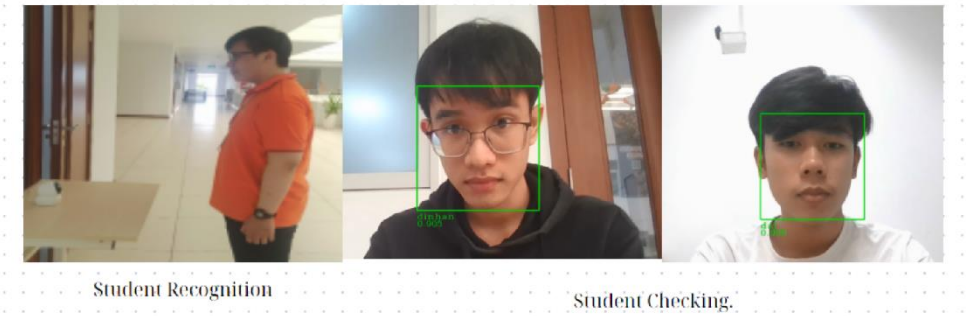


Fig. 13 *Student Recognition*

Case the student's face was recognized successfully and exceeded the recognition threshold, we retrieved the student code associated with that recognition (compared this code with the list of students who had scheduled exams for the day), if the student's code was found in the list of scheduled exam takers, we allowed the student to enter the room. In the opposite case, we raised a warning about an unrecognized student and denied access to the room. By following these steps, we ensured that only students with scheduled exams were granted, while unidentified individuals were denied entry to the room.

Each time a student is allowed into the room, that student's ID will be saved in a temporary file in the order in which the test was performed. Then use this ID to label students when they enter the tracking area of the camera tracking. The time that a student is labeled or appears in the frame of the tracking camera will be recorded and written to the database. When a student leaves the room, it means that he has disappeared on the tracking camera, recorded that time and written it in the database. In case students are forced to go out in the middle of the exam, they must re-identify when entering.

Fig. 14 *Student tracking in the exam room*

During testing we discovered the default function to read and show the live video from the camera got delayed very much, the fps is too low (from 1.45 to 5). If it got delayed for a long time, the frame would be lost and cause an error. The solution to this problem, we use FileVideoStream (a class provided by the imutils library in Python) - is designed to read frames from a video file or a video stream efficiently and provides a convenient way to iterate over frames, allowing you to process them one by one. This can be particularly useful for us when working with large video files or video streams that you want to process in a memory-efficient manner.

V. EXPECTED RESULTS

We used the same generated videos with two different models (*the bounding box method of MTCNN and the bounding box processing method has been improved to avoid mold distortion—the face of the object*),

the results of the improved model show accuracy on each frame significantly increased, and the rate of wrong object recognition also improved increased by a tiny amount.

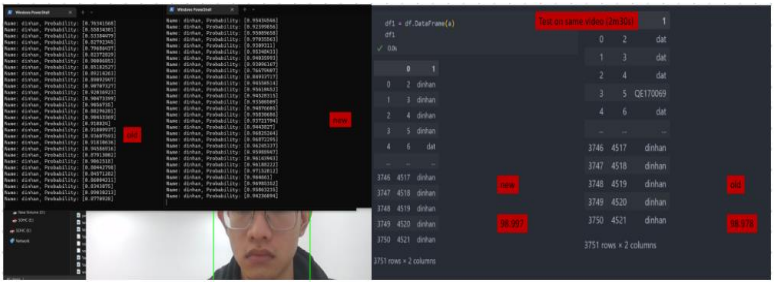


Fig. 15 *Compare effect off the way got bounding boxes to model*

We experimented with the original FaceNet model and our project model in 2 cases: identification in case of low light or bad recognition conditions and identification under ideal conditions. The result with environmental conditions is not good, the accuracy of both models is quite low (*43% and 55%*), which means that both models do not really perform well in this case. But they achieved high accuracy and the model solved the bounding box problem and got higher precision Fig16.

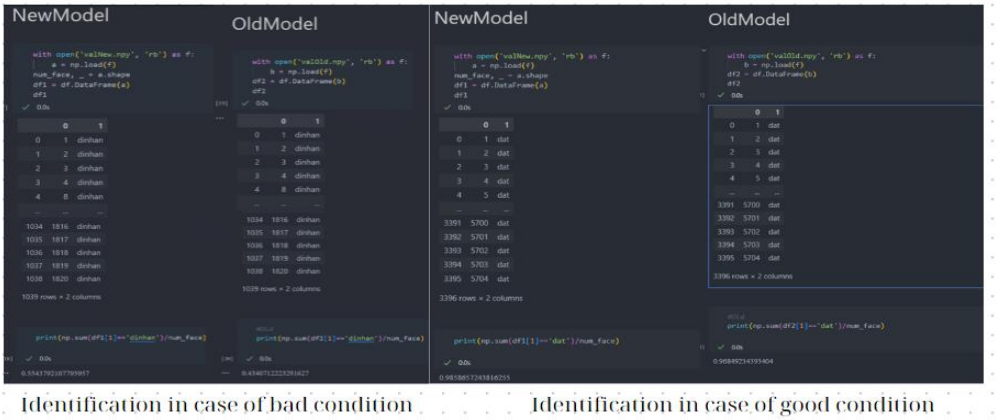


Fig. 16 *Compare two identity models*

The first method involved using FileVideoStream, which generally achieved higher frames per second (fps) compared to the OpenCV of YoLoV8. Sometimes in case, FileVideoStream is slightly lower but FileVideoStream is shorter. This problem can be attributed to two main factors. Firstly, there might be limitations in the algorithm used for deep sort tracking, which could be further optimized. Secondly, the hardware and equipment of the system may not be well-suited for the task, potentially requiring upgrades or more powerful components to handle the processing demands effectively.

Table. 2 Compare *FileVideoStream and default OpenCV*

	FIleVideoStream	Default
0	0	0
1	23.22055	19.96346
2	37.0358	29.48544
3	31.23993	31.17399
4	40.0174	32.3405
5	41.66844	26.31539
6	32.24304	30.29844
7	32.25569	32.26388

8	29.32465	34.46995
9	35.7711	34.47987
10	40.11615	28.58187
11	40.00023	30.30348
12	37.03743	35.71353
13	29.40132	28.57136
14	37.05412	34.48356
15	39.99947	37.03645
16	39.99947	38.46186
125	26.31704	39.98459
126	27.77225	41.68791
…	…	…
963	47.64035	43.47871
964	41.64734	38.4608
965	45.45784	35.71444
966	45.4741	41.66638
967	39.99832	35.71414
968	37.02501	38.44881
969	43.47871	38.46256
970	45.45833	34.48129
971	47.63223	40.0029
972	41.65107	40.00175
973	32.25991	37.03514
974	45.45439	41.66844
975	45.4741	38.45445
976	45.43765	38.46503
977	45.44947	43.47781
978	35.70654	37.03809
979	32.26636	38.45974
980	47.61493	47.64197
981	45.46967	38.45269
982	47.6252	38.45974
983	47.60412	34.48242
984	38.45798	41.65975
985	35.71535	40.01817
986	43.47871	35.71444
987	45.44947	37.01619
988	43.4742	41.67755
989	47.64738	37.04005
990	37.03711	38.47174
991	33.32436	28.57039
992	43.49223	43.48051
993	43.46429	39.97964
994	45.44897	43.50171
995	40.01778	38.46221
996	30.2945	38.45093
997	52.65522	41.6788
998	47.62034	37.02599

999	45.45144	39.99527
1000	49.97443	40.00366
Average	41.1952762	37.1552461
Delay time in beginning	15 seconds	45 seconds

VI. CONCLUSION

Our research project is to build a facial recognition system to identify candidates in the exam room. The project is based on FaceNet, MTCNN, Mediapie, and YOLO8 to improve recognition accuracy. Our system has high accuracy when recognizing in good condition environments but not really good in bad recognition conditions for many reasons: quality video will affect the training dataset and our data set has not covered all the cases, computer hardware conditions, and camera quality, and our deep sort algorithm is not computationally optimized, leads to a long delay and when tracking is also affected not less.

In future studies, we continue to improve the recognition process and supplement the image data set. We will experiment with other operations on the Quy Nhon campus such as: quickly searching for student information, and going to the library to borrow books,...

ACKNOWLEDGMENT

This project is sponsored by the RESFES 2023, FPT University.

REFERENCES

[1] Đoàn Hồng Quang, Lê Hồng Minh, Thái Doãn Nguyên, "*Nhận dạng khuôn mặt trong video bằng mạng nơ ron tích chập"*, Vietnam Science and Technology Magazine, 62(1),2020.

[2] Đặng Nguyên Châu, Đỗ Hồng Tuấn, "*Tổng quan các phương pháp nhận dạng khuôn mặt dựa trên các đặc trưng cạnh*, Journal of Science and Technology University of Danang, No. 05(114),2017.

[3] Mai Văn Hà, Nguyễn Thế Xuân Ly, "*Ứng dụng thuật toán FaceNet xây dựng hệ thống nhận dạng khuôn mặt*", Journal of Science and Technology University of Danang, No. 07(19),2021.

[4] Mihir Jain, Suman Mitra and Naresh Jotwani, "*Eye detection and face recognition using line edge map*", Proc.National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics, Gandhinagar, India, 2008

[5] MapYongsheng Gaoand Maylor K.H. Leung, "*Face Recognition Using Line Edge Map*", EEE transactions on Pattern analysis machine intelligence, Vol 24(06),2002.

[6] Vipin Kumar and Mohit Mehta, "*Face Recognition using Line Edge Map*", International Journal of Multidisciplinary and Current Research, 2014.

[7] Muhammad Zamir, Nouman Ali, Amad Naseem, Areeb Ahmed Frasteen, Bushra Zafar, Muhammad Assam, Mahmoud Othman and El-Awady Attia, "F*ace Detection & Recognition from Images & Videos Based on CNN & Raspberry Pi*", Computation 2022.

[8] Saibal Manna, Sushil Ghildiyal and Kishankumar Bhimani,"*Face Recognition from Video using Deep Learning*", Proceedings of the Fifth International Conference on Communication and Electronics Systems , ICCES 2020.

[9] Bishop and Christopher M. "*Pattern recognition and Machine Learning*", Springer, 2006.

[10] Nguyễn Phương Hạc và Nguyễn Thu Nguyệt Minh, "*Nghiên cứu các kỹ thuật trích rút thuộc tính trong bài toán nhận diện khuôn*", tạp chí Công Thương, số 22, 2020.

[11] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li and Yu Qiao, "*Joint face detection and alignment using multitask cascaded convolutional networks*" IEEE Signal Processing Letters, vol. 23, no. 10, pp. 1499–1503, 2016.

[12] Edwin Jose, Greeshma M., Mithun Haridas T. P, Supriya M. H., "*Face Recognition based Surveillance System UsingFaceNet and MTCNN on Jetson TX2*", 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS).

[13] Florian Schroff, Dmitry Kalenichenko, James Philbin, "*FaceNet: A unified embedding for face recognition and clustering*" Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 815-823. doi: 10.1109/CVPR.2015.7298682.

[14] Kilian Q. Weinberger, and Lawrence K. Saul, "*Distance Metric Learning for Large Margin Nearest Neighbor Classification*", Journal of Machine Learning Research 10 (2009) 207-244.

[15] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "*Rethinking the Inception Architecture for Computer Vision"*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Pages: 2818-2826, 2016.

[16] Jeong-Seop Han, Choong-Iyeol Lee, Young-Hwa Youn, Sung-Jun Kim4, "*A Study on Real-time Hand Gesture Recognition Technology by Machine Learning-based MediaPipe*", Journal of System and Management Sciences, Vol. 12 (2022) No. 2, pp. 462-476.

[17] Haitong Lou, Xuehu Duan, Junmei Guo, Haiying Liu, Jason Gu, Lingyun Bi, Haonan Chen, "*DC-YOLOv8: Small Size Object Detection Algorithm Based on Camera Sensor*", Preprints (www.preprints.org), 7 April 2023.

[18] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, Zbigniew Wojna, "*Rethinking the Inception Architecture for Computer Vision*", IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.