FPT UNIVERSITY

School Attendance Using Facial Recognition

By:

Tran Giang Anh, Nguyen Tran Nhat Bao

Supervisor:

Do Thai Giang

Undergraduate Project

Monday 19th December, 2022

Acknowledgements

We would like to express ours special thanks of gratitude to our mentor - Mr. Do Thai Giang for his patience and time, for urging, instructing us to complete this project.

We would like to thank all teacher, to thank our University, FPT University, for giving us a opportunity to study and grow in the best possible environment.

We would like to thank our classmates in AI1501, for letting us to meet, to learn and to grow up together.

We also don't forget to thank our family, who even though we may not be sure what we are doing is best for us, are always there encourage and support. Thanks to them, we have the will, the energy and the confidence to pursue our goals.

Abstract

Pandemic situation has transformed the way of studying and educating students. Education is undertaken through a variety of online platforms. Not just in normal but in online education, monitoring the attendance of the students is very important as the presence of students reflects the good assessment for teaching and learning. One way to do this is taking attendance through faces - face recognition. Face recognition or face detection has become a major field of interest and has received considerable attention in recent years. Face Recognition is a computer application that is capable of detecting, tracking, identifying or verifying human faces from an image or video captured using a digital camera. It can be used for security, authentication, identification ,.... In educating, this system mostly aims to build a attendance system which uses the concept of face recognition as the existing manual attendance system is time consuming and cumbersome to maintain. There are also chances of fake - false attendance. Thus, the need for this system increases. In this project, we will build a system to take attendance through faces based on FaceNet, Haar Cascade and MTCNN (Multi-task Cascaded Convolutional Networks).

keywords: Haar Cascade, MTCNN, FaceNet, face recognition

Table Of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

‘ Over these last few years, technology and computer vision improving and spreading faster than ever before, their application increases efficiency and quality in nearly every field of profession. One particular area is the attendance system, which nearly all organizations with a human workforce require, nowadays utilizing state-of-the-art face detection and face recognition technology. Regular attendance methods, for example in a learning environment, usually take 5-10 minutes per day, sometimes the class is interrupted by 1-2 people arriving late. Sometimes there is also a case of missing attendance. 5-10 minutes is not much or the interruption is not a too serious problems, but sometimes it can interrupt the circuit, reducing the quality of the lesson. Sometimes more seriously, there is a case where the attendance records substitutes to study for the subject that the subject teachers cannot know. From the examples above, it can be seen that regular attendance has a lot of loopholes that can be bypassed with just a little carelessness. Especially in the current context, the pandemic has caused most of the learning methods to shift from face-to-face to online, which makes it even more necessary to ensure the quality of the lessons as well as the closeness in attendance, more than ever. Because of the above needs, it is common for people to apply machine learning, computer vision models to the education, especially attendance. By implementing this upgraded system, organizations and their employees no longer need to manually check themselves in, their

cameras will automatically capture their faces and mark their attendance. Which reduce the chances of the problems mentioned above happening, as roll call no longer needs to read each person's name, done manually, but is now read by identity, by each person's face. The quality of the lesson and work will be improved because it is not necessary to take 5-10 minutes to take attendance, and because reading according to each person's identification characteristics, it is almost impossible to fake attendance.

Therefore in this project, we create a system to automatically taking attendance. Specifically, we using Haar Cascade and MTCNN for face detection to crop a human face out of the video feed which thereafter is vectorized using FaceNet, outputting an embedding vector. By comparing these face embedding vectors, the system can recognize an individual quickly while containing only few face samples, reducing space and resource.

The Haar-Cascade face detection algorithm proposed by Viola and Jones [4] is able to achieve relatively good performance very quickly. However in real-work applications where there are multiple faces in different angles and lighting conditions, the detector may not ensure the desired result but its speed is a strong advantage to consider using. In recent years, Convolutional Neural Networks (CNNs) have attracted a lot of attention due to their successes in various computer vision tasks, including face detection. One of those successes is MTCNN (Multi-task Cascaded Convolutional Networks), a deep learning method for joint face detection and alignment. MTCNN includes 3 CNNs or 3 stages to find all the bounding boxes and facial landmarks in a frame. Due to its complexity, it delivers excellent result while being relatively slower than Haar-Cascade, especially on lower-end hardware.

Previous facial recognition algorithms mainly represent the face by feature vector over a bottleneck layer to reduce the data's dimension. However, the dimension of the feature vector is relatively large (usually ≥ 1000), which leads to decreased recognition speed. In order to improve that, the PCA algorithm is often applied to reduce the dimension of the feature vector and speed up recognition time. Furthermore, in older identification methods, the loss function only calculates the similarity between 2 images. This approach is not the most efficient as one training input only receives one result: either the same if the 2 images belong to the same class, or dif-

ferent if the two images belong to 2 separate classes. FaceNet improves upon those weaknesses by implementing a convolutional network as its base and directly trains its output as a 128-D embedding vector. The model is also optimized by the triplet loss function that will be discussed later in this report. For the project, FaceNet is implemented by basing on the Inception Resnet V1 architecture.

1.2 Literature Review

A significant research issue spanning many professions and disciplines is face recognition. This is due to the fact that facial recognition, in addition to having many real-world uses like bankcard identification, access control, security monitoring, and surveillance systems, is a basic human behavior that is necessary for successful interpersonal relationships.

Before 2000 there are many research and practical performances of the face recognition and detection was not satisfactory until Viola and Jones release their algorithm. These Viola and Jones [5][6] are the first who are applying windows (rectangular boxes) for the face. But, it has lot of drawbacks as its feature size was large. In a 24×24 image, the total number of Haar_like features about 160,000[7] and it í unable to handled unfamiliar faces and frontal faces

After seeing the above problems, more complicated method have been introduce with more complicated features(SURF, SIFT, HOG and ACF)[8][9]. Another well known method is Dlib[10] which took support vector machine as a classifier in the face detection.

Another face detection method is using Convolutional Neural Networks (CNNs). Recently, CNNs achieve remarkable progresses in a variety of computer vision tasks, such as image classification [11] and face recognition [12]. Inspired by the good performance of CNNs in computer vision tasks, some of the CNNs based face detection approaches have been proposed in recent years. [13] train deep convolution neural networks for facial attribute recognition to obtain high response in face regions which further yield candidate windows of faces. However, due to its complex CNN structure, this approach is time costly in practice. [14] use cascaded CNNs for face detection, but it requires bounding box calibration from face detection with

extra computational expense and ignores the inherent correlation between facial landmarks localization and bounding box regression.

Face alignment also attracts extensive interests. Recently,[15] proposed to use facial attribute recognition as an auxiliary task to enhance face alignment performance using deep convolutional neural network.

A lot of face detection and face recognition work has been research and complete. As it is the best way for recognizing a person. Since a lot of methods invented for face recognition and face detection.

Chapter 2

Fundamental Theory

2.1 Face Detection

Face detection is a computer technology being used in a variety of applications that identifies the locations and the landmarks of human faces in digital images. It is the first step in the whole face recognition pipeline. Face detection algorithms vary and this section would examine and evaluate 2 of the most popular methods: Haar-Cascade and MTCNN.

2.1.1 Haar Cascade

The Haar Cascade classifier is a very easy face detection method. It's quite popular in the face recognition community because it is an excellent object detection and classification tool because of its lightweight, easy-to-use structure and very high recognition speed. With a huge dataset, many different objects can be trained like faces, cars,...

Although there are many alternative algorithms (HOG + Linear SVM, SSDs, Faster R-CNN, YOLO, etc.) that are more accurate than Haar cascades, they are still helpful and important in the modern world. The quickness, simplicity of use, and quick object detection time of the Haar cascade are some of its advantages. Because of this advantage, Haar cascade tend to be less accurate than more "contemporary" algorithms available today, are more likely to produce false-positive detections, and necessitate parameter modification when used for inference or de-

tection.

Haar Cascade is an algorithm created based on those features to detect objects (like faces, body, eyes, hands, objects,...) proposed by Paul viola and Michael Jones in 2001[5]. Their report states: "Rapid object detection using a boosted Cascade of simple features". Face image is characterized by the set of pixels in the faces' region that make them different from other pixel regions. However, for an input image, using individual pixels is inefficient. In the paper, Paul Viola and Michael Jones presented a new and faster method for image processing and face detection using rectangular feature points as shown below. Rectangular features similar to kennels are used to detect different facial features such as eye and nose as shown in the illustration.

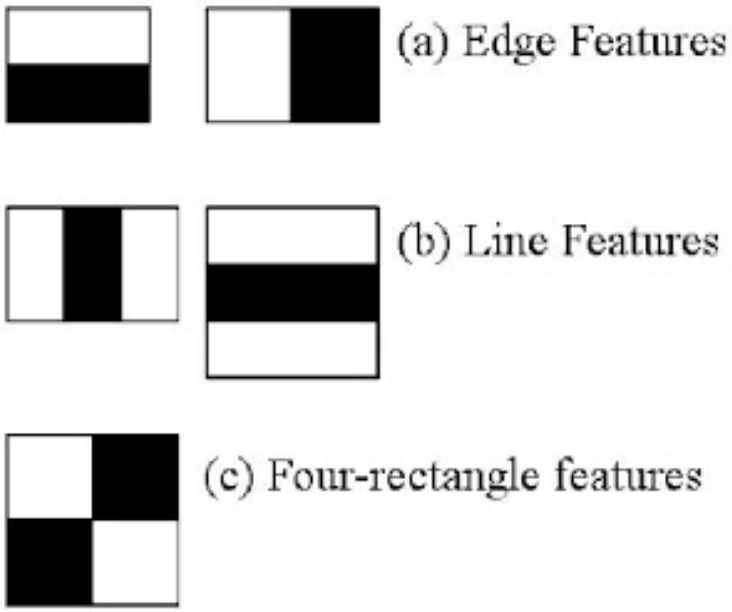

Figure 2.1: HaarCascade features

The rectangular features are shown run through the image and the total number of pixels in the white part is subtracted from the total number of pixels in the black part. In there

a) Are Haar-like feature that capture edges in image

b) Capture lines in image

c) About the feature of 4 squares

Haar-like feature values calculate by simply subtracts the sum of pixels under the white region from the sum of pixels under the black region, if it higher than certain self set threshold - it's a haar-like feature. It's interesting to note that these values are crucial for facial detection:

1. Cheek areas typically have a darker color than the eye areas.

2. Brighter than the eye area is the nose area.

Therefore, given these five rectangular regions and their corresponding difference of sums(values of Haar-like feature), we can form features that can classify parts of a face. But to calculate the values of Haar-like features for all locations on the image requires quite a large computational cost, which is not suitable for applications that require run-time. Integral Image is a method presented by Viola and Jones that uses a 2-D array the same size as the image to calculate Haar-Like feature values. Each element of this array is created by adding the image's (line-1) and its neighboring (column-1) scores.

$$P(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y') \quad (1)$$

It is quite easy to sum the gray level values of any place on the image after calculating the Integral Image in the manner shown below:

Assuming we must determine the overall gray level value of region D as displayed below, we may do so by using the following formula:

$$D = A + B + C + D - (A + B) - (A + C) - A \quad (2)$$

With A + B + C + D (x4,y4) represent the P4 point value in the Integral Image, therefore A + B (x3,y3) is P2 point value, A + C (x2,y2) is P3 value, and A is P1 value. The above expression can be re-write:

$$D = (x4, y4) + (x1, y1) - (x3, y3) - (x2, y2) \quad (3)$$

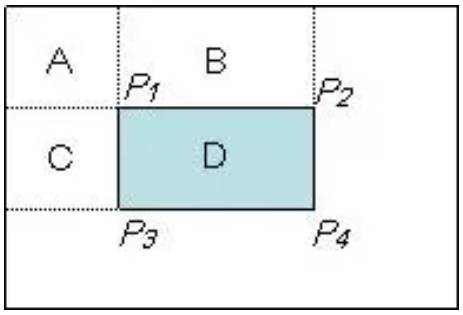

Figure 2.2: Integral image calculate formula

In the illustration below, the first rectangular feature is to calculate the difference in intensity between the eye area on the face. And the second rectangular feature is to measure the difference in intensity between the two areas of the eye and the bridge of the nose. Haar-like features can only seek in a specific area in image for identification. These specific area called sliding windows.

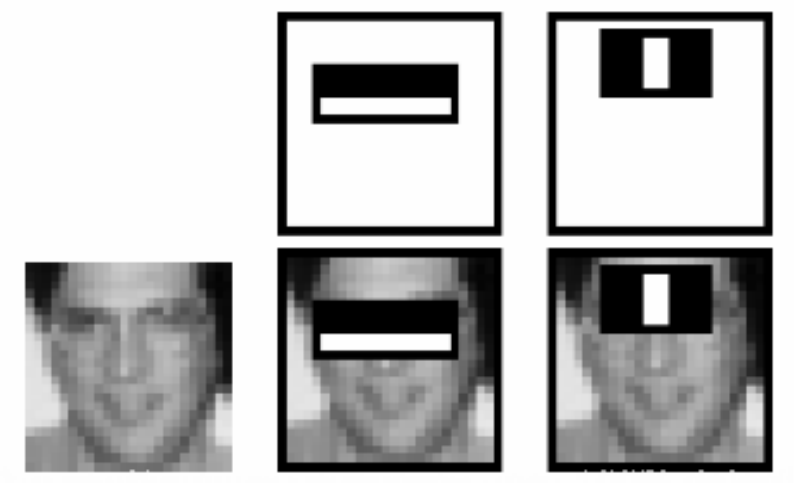

Figure 2.3: HaarCascade Demo

The term "sliding window" refers to the process of computing an output value for each central pixel of the kernel by sliding a fixed-size window across the image at various scales while moving a tiny matrix across the image from left to right and top to bottom. At each of these stages, the window comes to a stop, computes the characteristics, and then labels the region as either Yes or No depending on whether or not an item is there.

However, computing these Haar-like features first, then swiping a fixed sliding window across each (x, y)-coordinate of a picture, and then completing the classifi-

cation itself can be computationally expensive.

Viola and Jones introduced the idea of cascades or phases to address this. The sliding window must pass a series of tests at each stop along its path, each of which requires more computing than the one before it. The window is automatically discarded if any one test fails.

One advantage of the Haar cascade is that it can compute Haar-like features very quickly because integral pictures are used (also called summed area tables). The ability to recognize faces in photos regardless of the location or size of the face may be the most significant feature. The Viola-Jones object detection technique can also operate in real-time.

2.1.2 MTCNN

Another way to locate faces location in images is using MTCNN (Multi-task Cascaded Convolutional Networks) In terms of structure, MTCNN consists of 3 CNNs (Convolutional Neural Networks) stacking and concurrency works when detecting and identifying faces. The output data of MTCNN is the feature vector representing the face position identified in the image.

Stage 1: P-Net

First, in one image there may have more than one person - more than one faces. Beside, each faces have different size. Therefore, by using image resize, to create a series of copies from the original image with different sizes, from large to small, forming an Image Pyramid

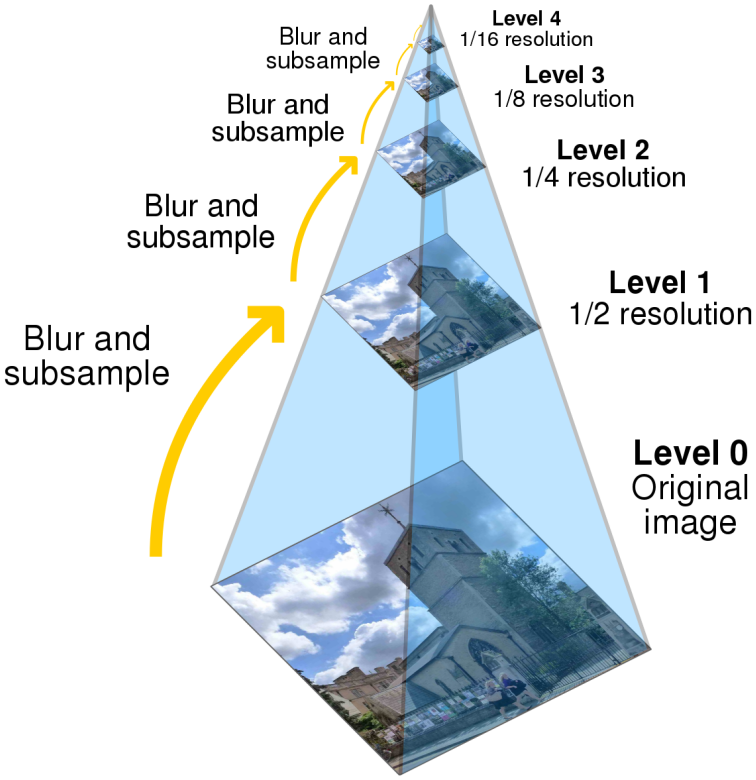

Figure 2.4: Image Pyramid

We employ a 12x12x3 pixel kernel and stride = 2 to iteratively scan each copy-resized duplicate of the original image in order to find faces. Although just utilizing one kernel with a set size, the network can readily detect faces in a variety of sizes since the duplicate of the original image is in varied sizes (greater image = larger face; smaller image = smaller image). Then, we'll add the kernels that were previously sliced and sent across the P-Net (Proposal Network).

P-Net is a CNN network to obtain bounding boxes which contain faces and regression vectors. Next, the bounding box contain faces calibrated based on regression vector regulation. Finally, the bounding boxes are stacked, merged to other calibrated bounding boxes to become one. The outputs of this process are a series of bounding boxes that can contain faces.

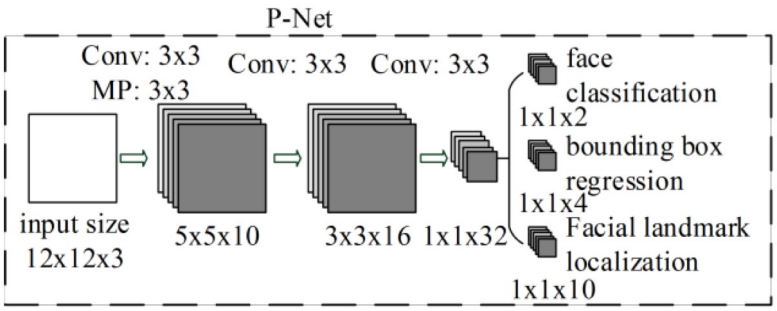

Figure 2.5: P-Net [1]

P-Net network using a CNN network with 3 convolution layers. As a result of P-Net, each kennel will have a number of bounding boxes, each of which will have four corner coordinates to identify its location (normalized to the range (0,1)) and confidence level. Setting the Threshold confidence level to remove low confidence boxes and utilizing NMS (Non-maximum Suppression) to remove the bounding boxes can help reduce the amount of bounding boxes on each image and kennel. with an overlap (Intersection Over Union) greater than a predetermined cutoff. The graphic below shows how duplicate boxes will be eliminated using NMS while only keeping the box with the highest level of confidence.

Figure 2.6: NMS Algorithm

The remaining box coordinates were converted to the original image coordinates after the unreasonable boxes were removed. Since the box's coordinates have been normalized to the kernel's interval (0, 1), all that is left to do is recalculate the kernel's length and breadth using the original image, then multiply the box's nor-

malized coordinates by the kernel's dimensions and add the coordinates of the kernel corners. The coordinates of the appropriate box on the original image will be the outcome of the aforementioned procedure. After that, the boxes will be shrunk to a square shape, given the new coordinates, and fed into the R-Net.

Stage 2: R-Net

All bounding boxes from stage 1 will be filtered by another CNN called R-Net to further eliminate a large number of sliding windows that do not contain faces.Stage 2 follows the same procedures as Stage P-Net. It also employs padding technique, which substitutes zero pixels for any missing pixels in the bounding box that extends beyond the edge of the image. Now, all bounding boxes will be scaled to 24x24x3 and supplied to R-Net as a kernel. A new coordinate for the remaining bounding boxes is produced as a result of this operation, and it is provided to O-Net in the following stage.

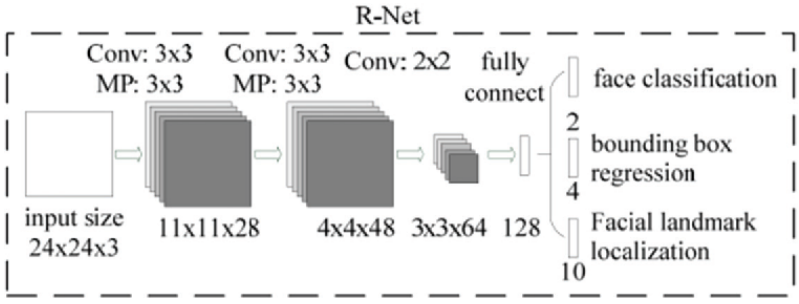

Figure 2.7: R-Net [1]

Stage 3: O-Net

Lastly is O-Net (Output Network). O-Net also use the same method as in R-Net. resizing to 48x48x3. However, the network's output now includes three more variables in addition to the box coordinates, such as the bounding box's four coordinates (out[0]), the coordinates of five facial landmarks (two eyes, one nose, two sides of the lips), and the confidence score for each box (out[2]). All output will be saved into a dictionary.

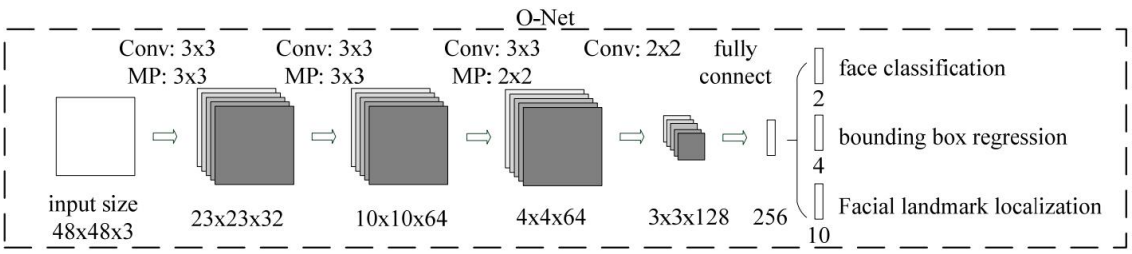

Figure 2.8: O-Net [1]

2.2 FaceNet

A face recognition system is a technology capable of matching a human face from a digital image or a video frame against a database of faces. It provides the ability to discern the identity of a person automatically and without human intervention which can be applied to various tasks, including taking attendance. One notable face recognition algorithm is FaceNet, which has been gaining a lot of popularity since published.

Schroff *et al.* [16]'s idea of FaceNet is using convolutional networks to represent a face image as a d-dimensional embedding vector in the Euclidean space. The model is trained in a way that the Euclidean distance between outputted vectors correlates with how the input faces relate to each other: the closer the distance the higher the similarity between matching faces and vice versa, visualized in Figure 2.9. To achieve that, FaceNet uses a triplet loss function. The way triplet loss works is that it tries to minimize the distances between positive pairs and maximize the distances between negative pairs. The speciality of FaceNet is the triplet loss function, the network behind it can be any convolutional network. For this project, FaceNet is built upon Inception Resnet V1.

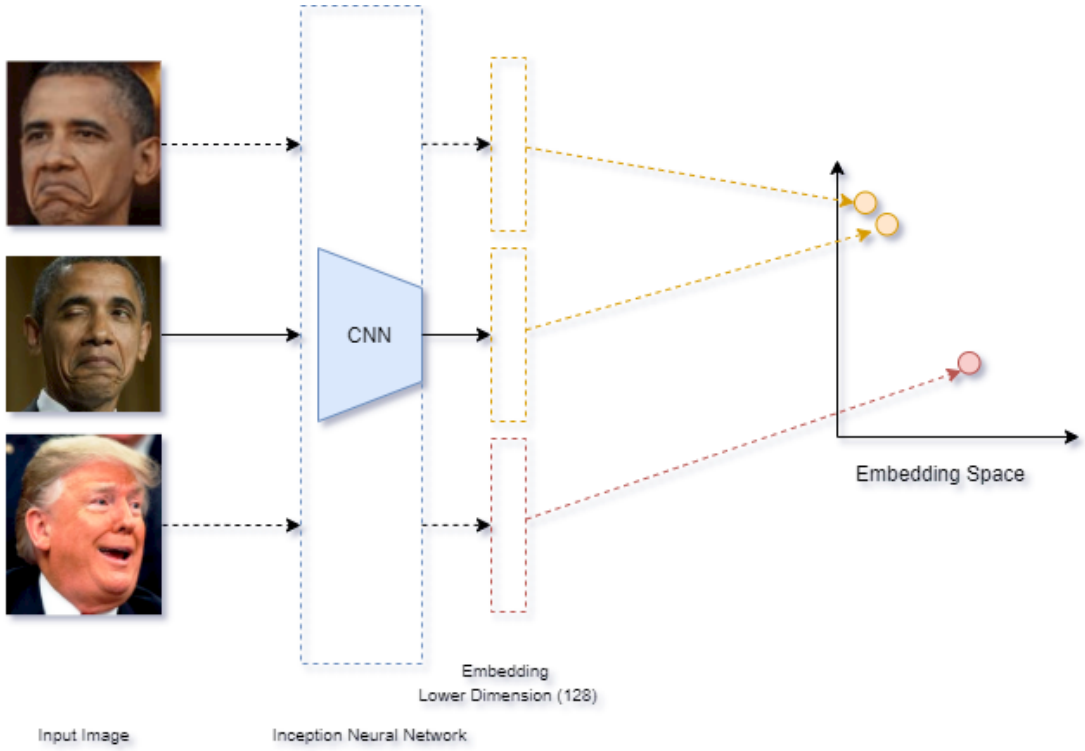

Figure 2.9: The objective of FaceNet

2.2.1 Inception Resnet V1

This network is a hybrid network inspired both by Inception and the performance of Resnet [2]. There are two versions: Inception Resnet V1 and V2. While their fundamentals are the same, V2 is more computationally complex than V1 and delivers higher accuracy. Similar to the Inception network, Inception Resnet V1 maintains its core structure by incorporating stem and Reduction blocks, however the Inception module within has its output added to the input (Residual Connection)

Layer	Size-in	Size-out	Kernel	Stride, Padding	Params	ReLU	Scale
ConV_BN_ReLU	160 × 160 × 3	79 × 79 × 32	3 × 3 × 3	2, 0	32 × 3 × 3 × 3	True	_
ConV_BN_ReLU	79 × 79 × 32	77 × 77 × 32	3 × 3 × 32	1, 0	32 × 3 × 3 × 32	True	_
ConV_BN_ReLU	77 × 77 × 32	77 × 77 × 64	3 × 3 × 32	1, 1	64 × 3 × 3 × 32	True	_
MaxPool2D	77 × 77 × 64	38 × 38 × 64	3 × 3	2, _	0	True	_
ConV_BN_ReLU	38 × 38 × 64	38 × 38 × 80	1 × 1 × 64	1, 0	80 × 1 × 1 × 64	True	_
ConV_BN_ReLU	38 × 38 × 80	36 × 36 × 192	3 × 3 × 80	1, 0	192 × 3 × 3 × 80	True	_
ConV_BN_ReLU	36 × 36 × 192	17 × 17 × 256	3 × 3 × 192	2, 0	256 × 3 × 3 × 192	True	_
5 × Inception A	17 × 17 × 256	17 × 17 × 256	Inception A	_	_	True	0.17
Reduction A	17 × 17 × 256	8 × 8 × 896	Reduction A	_	_	True	_
10 × Inception B	8 × 8 × 896	8 × 8 × 896	Inception B	_	_	True	0.1
Reduction B	8 × 8 × 896	3 × 3× 1792	Reduction B	_	_	True	_
5 × Inception C	3 × 3× 1792	3 × 3× 1792	Inception C	_	_	True	0.2
Inception C	3 × 3× 1792	3 × 3× 1792	Inception C	_	_	False	1.0
AvgPool2D	3 × 3× 1792	1 × 1× 1792	3 × 3	1, _	0		_
Flatten	1 × 1× 1792	1 × 1× 1792	_	_	_		_
Fully Connected	1 × 1× 128	1 × 1× 128	_	_	_		_
L2	1 × 1× 128	1 × 1× 128	_	_	_		_

Table 2.1: Inception Resnet V1 structure [3]

Residual Connection

One of the dilemmas of training neural networks is that we usually want deeper neural networks for better accuracy and performance. However, the deeper the network, the harder it is for the training to converge cause by the vanishing gradient problem. During training with gradient-based learning methods and backpropagation, the gradient might get extremely close to 0 and basically make no improvement to the weight, this is called the vanishing gradient problem. One way to mitigate the issue is by using residual connection (also known as skip connection) to jump over some layers.

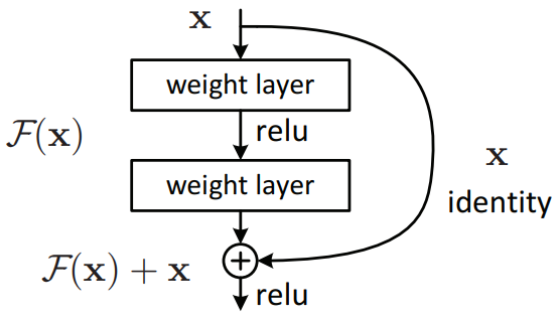

Figure 2.10: Residual Connection

Inception Modules

Inception modules are incorporated into convolutional networks in order to reduce computational cost, parameters and also overfitting. An Inception module performs convolution on an input with kernels of different sizes (1×1, 3×3, etc.), then concatenates all outputs into one before forwarding the result to the next layer with residual connection [2]. Prior to larger kernel convolutions which are more computationally expensive, an extra 1×1 convolution is added to reduce the number of input channels. Figure 2.8 details Inception modules in Inception Resnet V1.

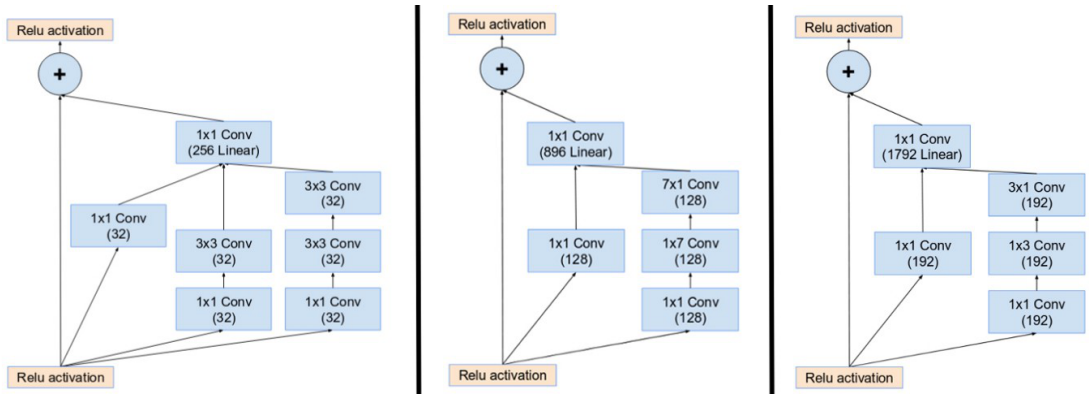

Figure 2.11: Inception Module A, B, C of Inception ResNet V1 [2]

In the above figure, the usual pooling layer is replaced by the residual connection which adds the output of the Inception module to the input. For the addition to work, the input and output after convolution must have the same dimensions. Therefore, an 1×1 convolution without activation is used to match their depth sizes as the Inception block reduces the dimension of the output. The pooling layer is moved to the reduction block. [2]

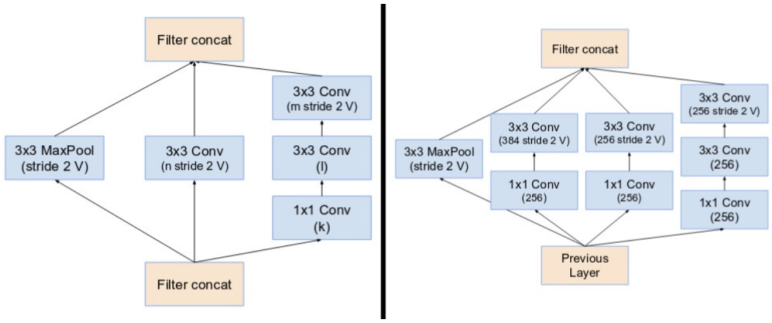

Figure 2.12: Reduction Block A, B of Inception ResNet V1 [2]

2.2.2 Triplet Loss

FaceNet outputs d-dimensional embedding vectors representing the input images (this process is depicted as $f(x) \in \mathbb{R}^d$). In this project, 128 is chosen for d as the result in [16] shows that 128 embedding outperforms other candidates such as 64, 128 or 512 [16]. Triplet loss that is used to optimize FaceNet is a loss function where a reference input (called *anchor*) is compared to a matching input (called *positive*) and a non-matching input (called *negative*). The goal is to minimize the distance from the anchor to the positive and to maximize the distance from the anchor to the negative.

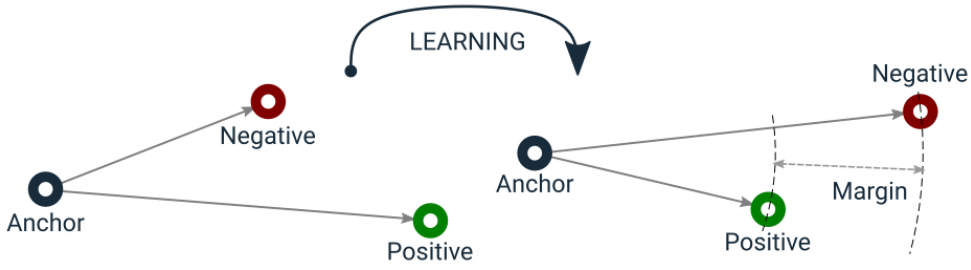

Figure 2.13: Triplet Loss Objective

Thus we want,

$$d(\mathbf{A}, \mathbf{P}) + \alpha < d(\mathbf{A}, \mathbf{N}) \quad (4)$$

$$\|f(\mathbf{A}) - f(\mathbf{P})\|_2^2 + \alpha < \|f(\mathbf{A}) - f(\mathbf{N})\|_2^2 \quad (5)$$

$$\|f(\mathbf{A}) - f(\mathbf{P})\|_2^2 - \|f(\mathbf{A}) - f(\mathbf{N})\|_2^2 + \alpha < 0 \quad (6)$$

where $\mathbf{A}$, $\mathbf{P}$, $\mathbf{N}$ respectively are anchor, positive, negative and α is a margin that is enforced between positive and negative pairs.

Therefore the loss function is:

$$L = \sum_{i}^{N} \|f(\mathbf{A}_i) - f(\mathbf{P}_i)\|_2^2 - \|f(\mathbf{A}_i) - f(\mathbf{N}_i)\|_2^2 + \alpha \quad (7)$$

where N is the total number of training triplets.

Since triplets that satisfy the constraint in equation (2) have no impact on the training and only delay convergence, their loss is adjusted to 0. Hence the loss function becomes:

$$L = \sum_{i}^{N} max(\|f(\mathbf{A}_i) - f(\mathbf{P}_i)\|_2^2 - \|f(\mathbf{A}_i) - f(\mathbf{N}_i)\|_2^2 + \alpha, 0) \quad (8)$$

2.2.3 Triplet Selection

Choosing suitable triplets is crucial to the training of FaceNet as it can result in faster convergence and higher performance. If training triplets are chosen at random by generating all possible permutations, there would be a large number of triplets that easily fulfill the constraint in equation (2), therefore slow the convergence down since they would still be passed through the network. The idea is to pick *hard triplets* that violate the constraint as they can actively improve the training. Hard triplets are:

$$\underset{\mathbf{P}_i}{\operatorname{argmax}} \|f(\mathbf{A}_i) - f(\mathbf{P}_i)\|_2^2 \quad (\textit{hard positive})$$

$$\underset{\mathbf{N}_i}{\operatorname{argmin}} \|f(\mathbf{A}_i) - f(\mathbf{N}_i)\|_2^2 \quad (\textit{hard negative})$$

However, it is impractical to calculate all the argmax and argmin across the dataset. The focused method in [16] is the generating triplets online method: compute and select the triplets from within a mini-batch during training.

Chapter 3

Data Preparation

3.1 Dataset

We collect a sample of 100 images of 10 famous persons from Google along with our own faces. These images vary in lighting conditions, angles, facial expressions but most of them only contain one person, some might have few small faces in the background due to a picture or being taken in public. Despite having quite a small sample size, it have been manually verified to have correct labels and clear images.

We also create a dataset of students in a classroom or a lecture while studying to simulate real-life scenarios. This dataset is collected by extracting frames that is facing the class, ensuring that they contain the students. Since this dataset is mostly collected by online sources, there is no label for any person, except for images we manage to capture in our class.

Also, to evaluate the face recognition system, we use both MTCNN and Haar Cascade to extract the faces. All results are saved as even mistake images that do not contain a human face are useful to determine whether or not FaceNet would embed a noise image close to a human face.

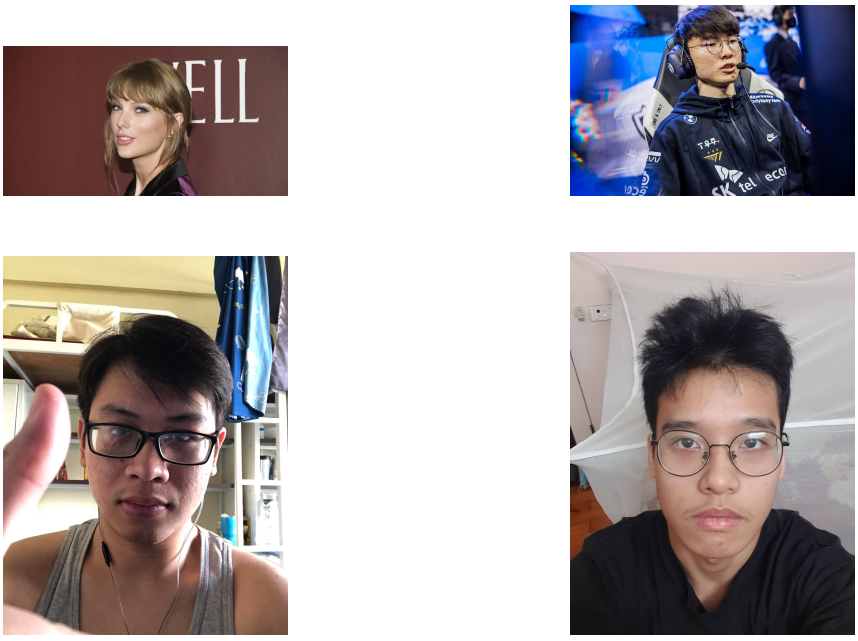

Figure 3.1: Face Images

3.2 Data preprocessing

3.2.1 Face Detection

For both Haar Cascade and MTCNN, it is very straightforward to preprocess the input image. Haar Cascade requires grayscale images meanwhile MTCNN works with RGB images. We use OpenCV to convert accordingly.

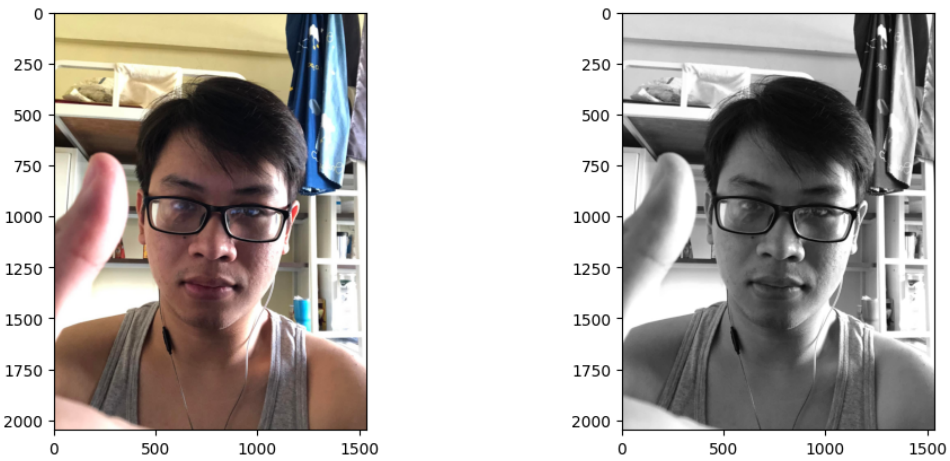

Figure 3.2: MTCNN/Haar Cascade preprocessing

3.2.2 FaceNet

Similar to the structure in Table 2.1, the input shape is (160, 160, 3), this FaceNet model expects 160 × 160 RGB face images to produce 128-d embedding vectors. We use only MTCNN to extract the faces in the dataset and resize those images to (160, 160, 3). It is also important to normalize all the pixels to [0, 1]

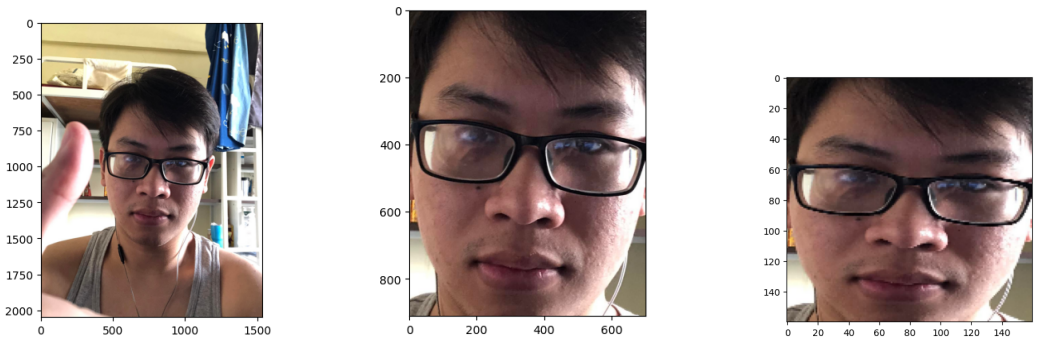

Figure 3.3: FaceNet preprocessing

While evaluating face detection algorithms, faces that do not belong to the class such as background people are still kept as it only matters if the algorithms can detect faces or not. Testing face recognition is however different and those faces that do not belong to the class need to be removed. That reduces the number of faces to 87 among 10 classes.

Chapter 4

Experimentation and Result

4.1 Face Detection

To compare and evaluate Haar Cascade and MTCNN, we test their reliability based on these metrics: time and accuracy.

For time we calculate the average time each method spend detecting faces in each image, also FPS in a live video detection. The results are obtained using a mobile Intel i5 9th gen processor and a mobile GTX 1050.

Method	**Average time(s)**	**FPS**
Haar Cascade	0.163	10
MTCNN	1.65	1

Table 4.1: Time result

For accuracy, we manually count the actual number of faces and the number of faces our algorithms detect. There are 104 faces in our dataset. TP (resp. TN) stands for true positive (resp. negative) and FP (resp. FN) for false positive (resp. negative). From these counts, one can compute the precision (p) and recall (r)

$$Precision(p) = \frac{TP}{TP + FP} \qquad Recall(r) = \frac{TP}{TP + FN}$$

The traditional F-measure or balanced F-score (F1 score) is the harmonic mean of precision and recall:

$$F1 - Score = 2 * \frac{p * r}{p + r}$$

Method	TP	FP	FN	Precision	Recall	F1-Score
Haar Cascade	94	33	10	0.7401	0.9038	0.8138
MTCNN	97	3	7	0.97	0.9327	0.951

Table 4.2: Accuracy result

From Table 4.2, Haar Cascade is more likely to mistake non-face entities as faces, however the amount of actual faces it detects is comparable to MTCNN while being much faster. This can be seen in Figure 4.1 and 4.2 where MTCNN manages to perfectly bound all the faces with no mistake, Haar manages to capture most of the faces and some mistakes.

Figure 4.1: MTCNN result

Figure 4.2: Haar result

Haar also struggles more in those images where human faces are not frontal, visible in Figure 4.3 and 4.4.

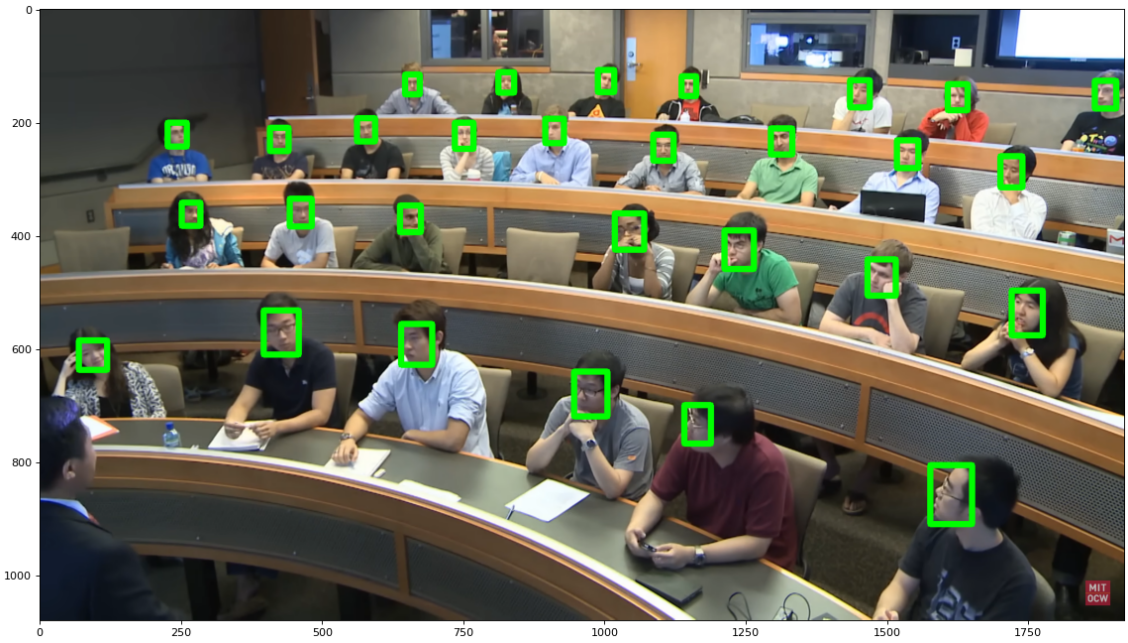

Figure 4.3: MTCNN result

Figure 4.4: Haar result

4.2 Face Recognition

When the face detection detects and extracts a portion of a frame in a video feed, whether it is a human face or not, it will be embedded by FaceNet to find another embedding vector with the closet distance to it in the database. If that distance surpasses a certain threshold, that frame does not contain any person in the database.

$$d\left(p,q\right)=\sqrt{\sum_{i=1}^{n}\left(p_i-q_i\right)^2}$$

Therefore, it is crucial to fine tune the most optimal threshold. We calculate the Euclidean distances between all positive pairs and negative pairs in our dataset. There are a total of 340 positive pairs and 3401 negative pairs. Matching pairs are labelled as 1 while non-matching ones are labelled as 0.

Distance	Match
0.788849	1
0.713350	1
...	...
1.336905	0
1.270743	0

Table 4.3: Distances of matching and non-matching pairs

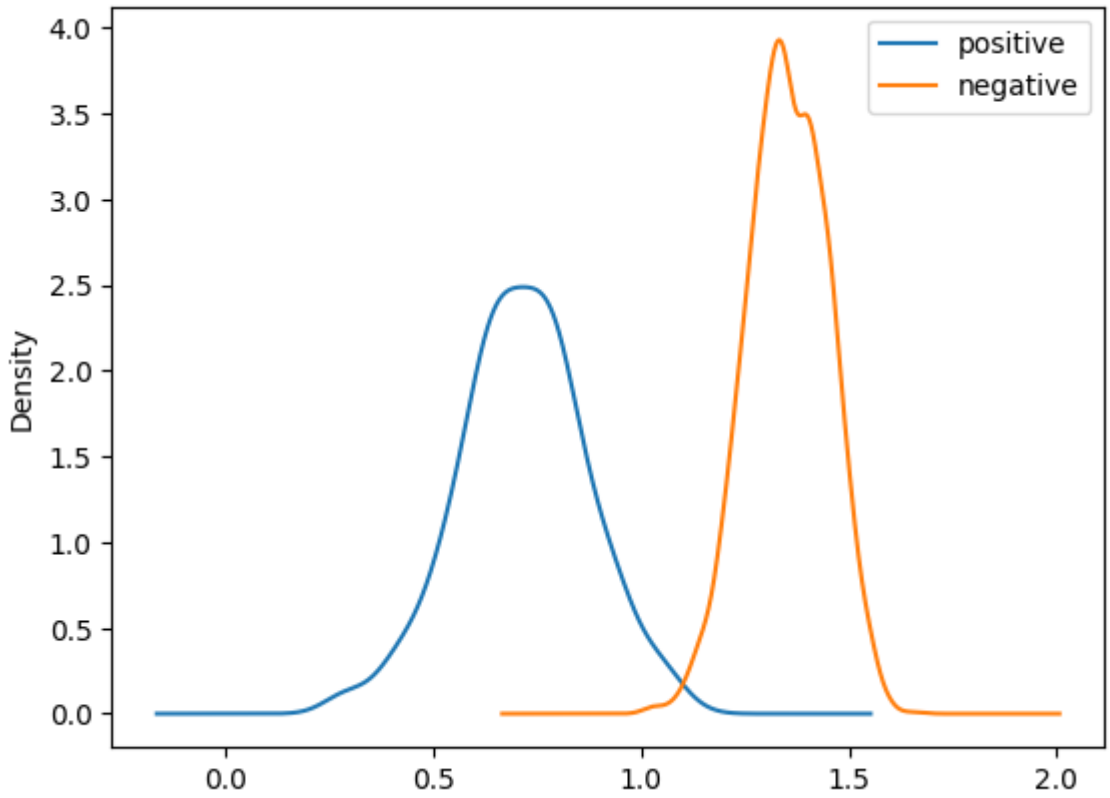

Figure 4.5: Distance kde plot

The maximum distance of the positive pairs is 1.122 whereas the minimum distance of the negative pairs is 1.001. This means that some samples have both positive and negative labels between 1.001 and 1.122. The optimal threshold should be in this range as it would not be too low to ignore a positive pair and not too high to mismatch a negative pair.

One way to find this value is by using Decision Tree with the depth of 1 as there is only one feature: distance. We can use decision tree here as this is a binary classification task. The scikit-learn library provides a simple decision tree implementation.

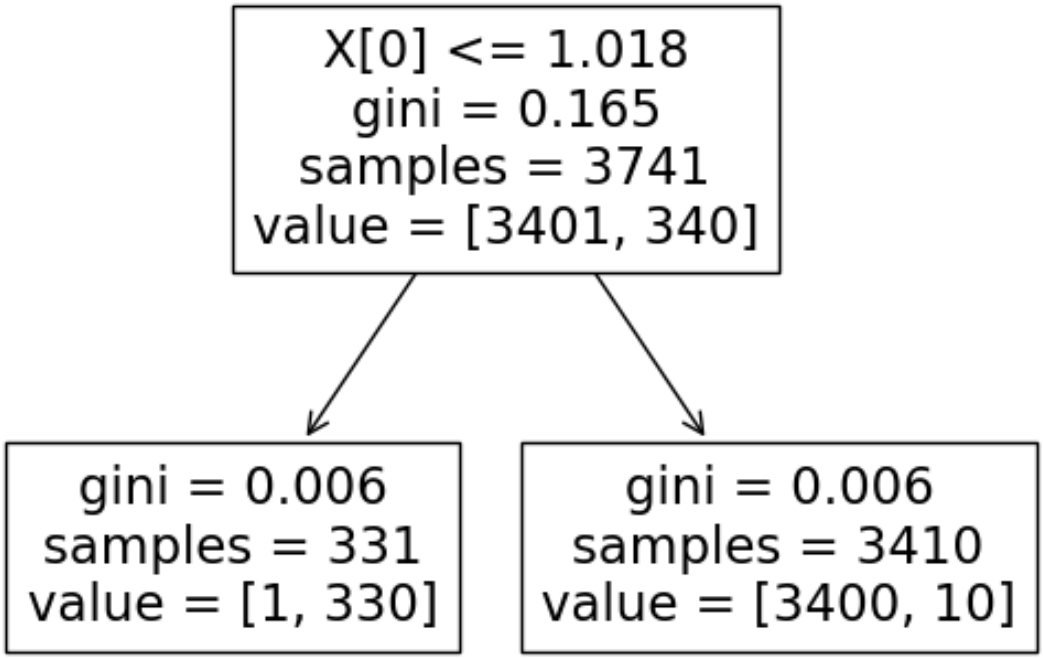

Figure 4.6: Decision tree result

1.018 therefore is our desired threshold and it is exactly in the aforementioned 1.001 to 1.122 range. We can then evaluate the metrics of this value with our dataset.

Threshold	**Accuracy**	**Precision**	**Recall**	**F1-Score**
1.001	0.9965	0.9967	0.9647	0.9806
1.018	**0.9971**	**0.9967**	**0.9706**	**0.9836**
1.122	0.9898	0.9016	0.9971	0.9469

Table 4.4: Threshold result

With threshold = 1.0018, any mistaken image by face detection would have the distance between it and any face surpassed the threshold and would not be incorrectly identified as a person.

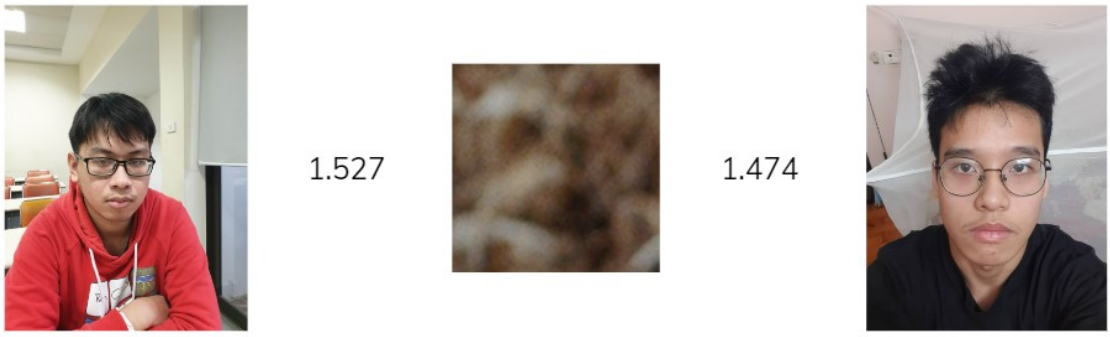

Figure 4.7: Distance between noise image and human

Chapter 5

Conclusion

While the precision and F1-score of MTCNN significantly surpass those of Haar Cascade, the recall value is slightly above by a small margin. This means that in real-life applications, Haar Cascade still manages to correctly detect faces most of the time, despite also detect non-face entities as faces. This is however not an issue as those non-face entities after being embedded by FaceNet, their embedding vectors would not be close to any of the face embedding in our database and would not be recognized as any individual. Furthermore, using Haar Cascade in a prolonged video feed would provide a large amount of time to detect all the faces and successfully mark their attendance. Higher FPS would mean more clarity for the people in charge to see how the system is working, which is a nice to have feature. MTCNN would be more suitable in preprocessing newly added images to the database as it will ensure extracting the faces and there is less time constraint for that workload.

Using FaceNet as the model for face recognition saves a significant amount of time and database space. With only one example per student in the database, with a good threshold value the model can effectively recognize their identity by pairing the recorded frame with the embedding in the database. Furthermore, when an additional student is added to the database, FaceNet does not require retraining, only the threshold value might need to be adjusted and that task can be accomplished easily with decision tree. Combining Haar Cascade, MTCNN and FaceNet and a reliable attendance system is created.

Bibliography

[1] Zhang. Kaipeng, Zhang. Zhanpeng, Li. Zhifeng, and Qiao. Yu. Joint face detection and alignment using multi-task cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[2] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alex Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI Conference on Artificial Intelligence*, 31(1):4278–4284, 2016.

[3] Xianbiao Qi and Lei Zhang. Face recognition via centralized coordinate learning. *ArXiv*, abs/1801.05678, 2018.

[4] Paul Viola and Michael Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[5] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. *2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1:I–511, 2001.

[6] Paul Viola and Michael Jones. Robust real-time face detection. *The International journal of computer vision (IJCV)*, 57(2):137–154, 2004.

[7] T. Mita, T. Kaneko, and O. Hori. Joint haar-like features for face detection. *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, 1:1550– 5499, 2005.

[8] Jianguo Li and Yimin. Zhang. Learning surf cascade for fast and accurate object detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, page 3468–3475, 2013.

[9] Bin Yang, Junjie Yan, Zhen Lei, and Stan Z. Li. Aggregate channel features for multi-view face detection. *IEEE International Joint Conference on Biometrics*, pages 1–8, 2014.

[10] Davis E. King. Dlib-ml. A machine learning toolkit. *Journal of Machine Learning Research*, 38:1755–1758, 2009.

[11] I. Sutskever A. Krizhevsky and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[12] X. Wang Y. Sun, Y. Chen and X. Tang. Deep learning face representation by joint identification-verification. *Advances in Neural Information Processing Systems*, pages 1988–1996, 2014.

[13] C. C. Loy S. Yang, P. Luo and X. Tang. From facial parts responses to face detection: A deep learning approach. *IEEE International Conference on Computer Vision*, pages 3676–3684, 2015.

[14] 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*, pages 5325–5334, 2015.

[15] Zhang. Kaipeng, Zhang. Zhanpeng, Li. Zhifeng, and Qiao. Yu. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016.

[16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 5(57):815–823, 2015.