University of Science and Technology of Hanoi

Information and Communication Technology Department

Bachelor Thesis

Academic year 2018 - 2021

Application of Face recognition to control devices

---

presented by

LUU Hai Nam

registered at University of Science and Technology Hanoi

supervised by Mr. LE Quoc Dat

internal supervised by Dr. TRAN Giang Son

Host organization: Harmony Enterprise Solution

Hanoi - Vietnam

ATTESTATION

_ I, LUU Hai Nam, hereby attest that my report **does not** contain plagiarism (copy/paste) from other sources without fully credited citation.

If plagiarism is to be detected, I am fully aware of the consequences of my actions, and I understand that my thesis and all of my internship results won't be accepted. In that case, I fully take responsibility and will happily accept any adequate penalty from the board and the university.

Tôi, Lưu Hải Nam, xin ký tên dưới đây nhằm tuyên thệ rằng bài báo cáo của tôi **không** đạo văn (cố ý sao chép) từ các nguồn khác mà không có sự trích dẫn đầy đủ.

Nếu cố tình sai phạm, tôi hoàn toàn nhận thức được hậu quả và hiểu rằng bài báo cáo cùng toàn bộ kết quả thực tập của tôi sẽ không được chấp nhận. Trong trường hợp đó, tôi xin nhận toàn bộ trách nghiệm cũng như hình thức kỷ luật tương xứng của phía hội đồng bảo vệ và nhà trường.

Ngày 14 tháng 5 năm 2022
Chữ ký/Signature

Lưu Hải Nam

Acknowledgements

I would like to express my thanks to my host organization Harmony Enterprise Solution for accepting me doing the internship. And my supervisor Mr. LE Quoc Dat, who gave me an incredible idea for the project. He also advised me, provided invaluable guidance and reviewed my entire project to make it complete. Secondly, I also would like to exhibit my thanks to my supervisor at the University of Science and Technology of Hanoi (USTH), Dr. TRAN Giang Son, for showing me some lackness in my work, which helps me to never stop expanding the work. I really appreciate my friends, all professors and staff at USTH for helping me the past three incredible years. After all, I express my sincere gratitude to my parents for their patience and their love throughout my life. Sincerely thank!

Lời cảm ơn

Trước tiên, tôi xin gửi lời cảm ơn tới Công ty Harmony Enterprise Solution đã cho tôi cơ hội được hoàn thành thực tập tại Công ty. Và tôi xin được gửi lời cảm ơn sâu sắc tới người hướng dẫn trong ba tháng vừa qua, Anh Lê Quốc Đạt, người đã cho tôi những ý tưởng độc đáo cho dự án này. Anh cũng chỉ bảo, cho tôi những lời khuyên quý giá và giúp tôi hoàn thiện dự án. Thứ hai, tôi cũng muốn gửi lời cảm ơn đến Tiến sĩ Trần Giang Sơn, người đã chỉ ra những thiếu sót của tôi để khiến tôi không ngừng phát triển dự án. Tôi cũng biết ơn sâu sắc tới những người bạn, các giảng viên và các anh / chị nhân viên tại USTH, đã giúp đỡ tôi trong suốt ba năm vừa qua. Cuối cùng, tôi muốn bày tỏ lòng biết ơn sâu sắc tới cha mẹ, vì sự nhẫn nại và tình yêu thương họ dành cho tôi cả cuộc đời này. Xin chân thành cảm ơn!

Table of contents

List of Figures

List of Tables

List of Acronyms

API Application Programming Interface. II, 22, 26, 27

CCU Central Control Unit. II, V, 4, 18, 20, 21, 22, 23, 26, 27

CNN Convolutional Neural Network. II, IV, 3, 8, 9, 10, 11, 12, 13, 15

COM COMMON. 21

COVID-19 Corona Virus Disease. 1, 29

DNN Deep Neural Network. 18

GPIO General Purpose Input/Output. 20, 21

IoT Internet of Things. II, 3, 4, 20, 21

MTCNN Multi-task Cascaded Convolutional Networks. II, 4, 8, 9, 18

NC Normally Closed. 21

SARS Severe acute respiratory syndrome coronavirus. 1

SDK Software Development Kit. 21

WSGI Web Server Gateway Interface Web Application. 14

Chapter 1

Introduction

1.1 Context and Motivation

A traditional door lock with a key or a door lock with fingerprint integrated is a standard security solution. It is still the most reliable and fast solution since the early days. Because each lock key has different spring-loaded pins in the cylinder, it requires a specific key to push the pins inside the lock body. We all have to touch fingers to that surface as a traditional door lock or a newer lock with a numeric pad or fingerprint, and this will make it like a mediate surface for the virus to stay.

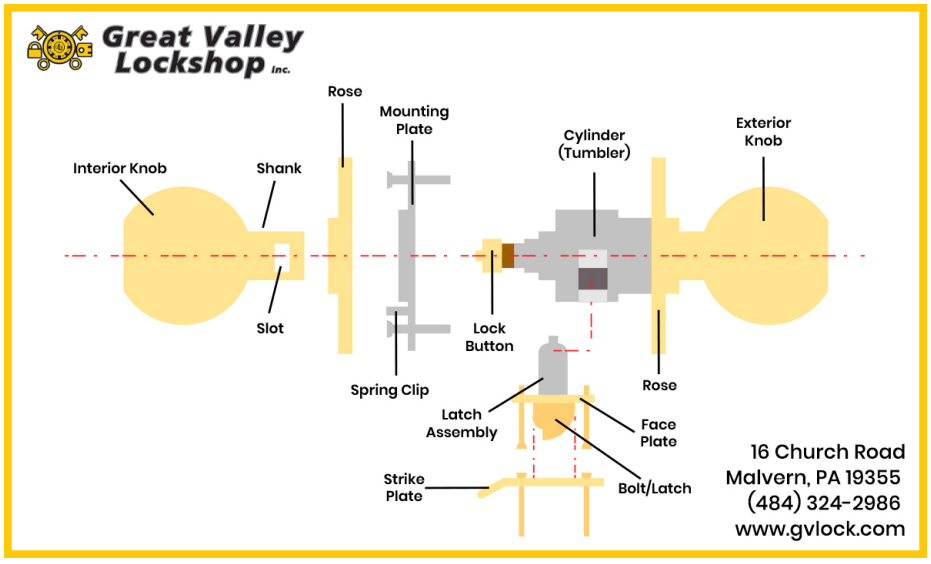


Figure 1.1: Diagram showing the inside of a door lock

source: gvlock.com

Corona Virus Disease is the most dangerous infectious disease in the last three years. From the first case in Wuhan City, China, it is a severe pandemic throughout the world. Max Roser, Hannah Ritchie, Esteban Ortiz-Ospina, and Joe Hasell[1] indicate that from late 2019, there are 176 million confirmed cases, with 3.81 million deaths. In Vietnam, as we experienced in the SARS epidemic in 2003, the situation here is in control now. However, since the pandemic is still a big problem outside Vietnam, according to Havard Health Publishing[2], we should clean frequently touched objects and surfaces regularly and wash our hands often with soap and water. An early

stage of prevention is essential in order not to have another outbreak here.

On the other hand, face recognition will solve the problem. As we do not have to touch any surface to verify that we are granted to open the lock, just a simple step is to look at the camera. The hard part is for the local server with a deep learning model to compare the face with a local database.

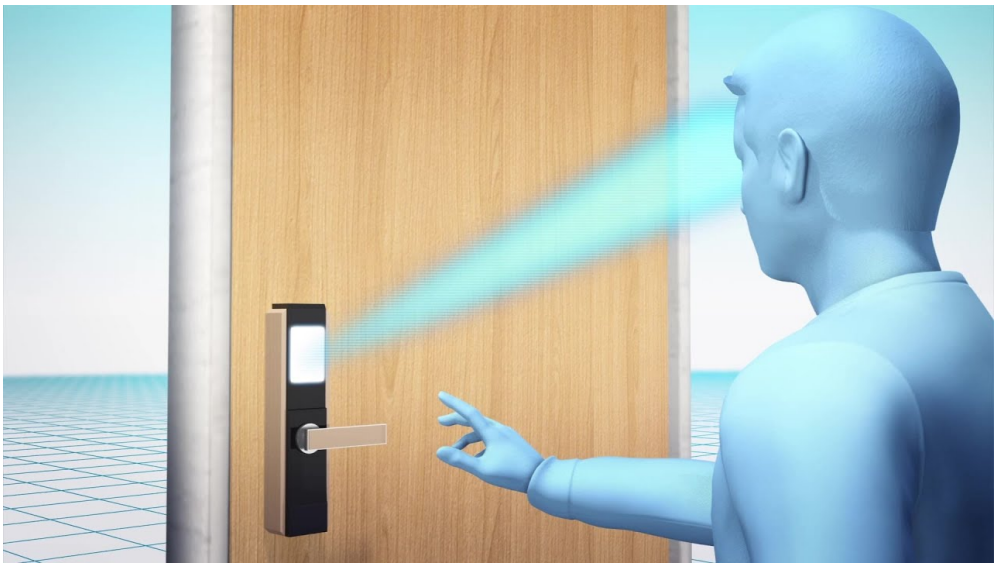


Figure 1.2: Door lock integrated with face recognition

As usual, many researchers in face recognition objects choose to train a model with a large of images for each class. This consumes much time, power to train, and we must re-train the model from scratch if a new person comes. Meanwhile, following to Florian Schroff et al.[3] choosing to follow the one-shot learning method can decrease the training model time, which benefits a scale-up in the future.

1.2 Objectives

The main topic of this internship is to improve the accuracy of the deep learning model to classify Asian faces better based on the pre-trained weights on the LFW dataset. It also applies IoT devices for hand-less control for domestic usage and integrates with attendance checks for household and industrial usage. Our goal after the internship is to study and use IoT connectivity, combined with a deep learning model to create a product that uses face recognition to validate if the person is granted to open the door and control smart home devices so that they can do it without hand touch.

The main objectives of this internship include:

- Study knowledge of Convolutional Neural Network and implement it on a pre-trained FaceNet[3] model.
- Propose to apply the Asian dataset to the face recognition model for better performance on recognizing Vietnamese.
- Study knowledge of IoT connectivity

1.3 Thesis Organization

This report is organized as follows:

1. Materials and Methods: presents our dataset, proposed methodology, and the evaluation scenario.
2. Result and Discussion: presents the evaluation results and discusses our results.
3. Conclusion and Future-work: conclude the work and presents future research directions.

Chapter 2

Materials and Methods

The product contains two main things that need to understand: making the model that is reliable enough to minimize the percentage of allowing a person that is not in the database can control the lock and making the connection between the recognition system to the CCU fastest, without delay, and most secure. Face recognition plays a role as an authentication layer, allow the action will be authenticated before start. To understand face recognition, we have to study image classification and feature extraction. IoT connection is responsible as a bridge layer to smart devices. Making it fast and instantly is easy, but we have to research blockchain connections for securing the connection.

With the proposed problem, we have split it into more minorissues:

- How can we detect faces in the camera stream.
- How to capture one frame containing a face from the camera stream.
- How to resize the captured frame from a full HD resolution to a smaller size.
- How can we verify the face in the captured frame.
- Which architecture of the model we are aiming to build.
- Which way to connect and control IoT devices.
- How to secure the connection between the recognition system to Central Control Unit.

To solving above problem, there are some tools and libraries we need to solve it:

- **OpenCV**: Using to resize the original frame to a smaller size, converting the image to grayscale for quickly detecting faces and further reducing the computational complexity. Besides, we need OpenCV to capture frames from a connected camera.
- **Haar Cascade**: Using for detecting faces.
- **Multi-task Cascaded Convolutional Networks**: Help us to extract keypoints of faces. With keypoints, we can measure if the presented face is a real face.
- **One-shot learning**: One-shot learning is used for reducing the retraining model from scratch problem when there is a newcomer.

To connect between the face recognition server to Central Control Unit for approving access to control IoT devices, we are using **Flask** because of its simplicity in development and maintenance.

2.1 Materials

2.1.1 OpenCV

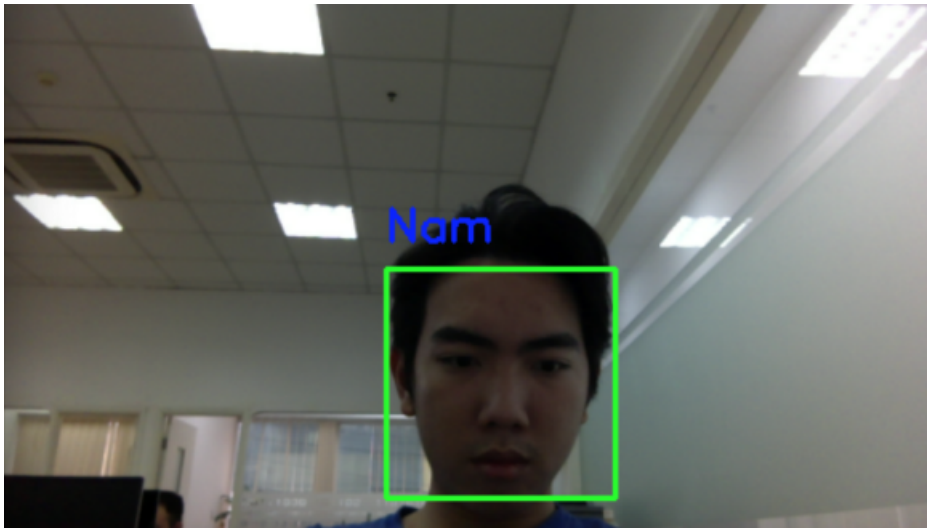


Figure 2.1: OpenCV Example

OpenCV was introduced at Intel in the year 1999 by Gary Bradsky. The first release came in late 2000. OpenCV represents for Open Source Computer Vision Library. Even though it is written in optimized C/C++, it has interfaces for Python and Java alongside C++. OpenCV has an active user base worldwide, and the number of users is increasing day by day due to the rise in computer vision applications.

OpenCV uses for image and video analysis, like facial detection, license plate reading, photo editing, advanced robotic vision, optical character recognition, and many more. OpenCV-Python is the Python API for OpenCV. It acts like a python wrapper around the C++ implementation of OpenCV.

OpenCV-Python is fast, and it is not difficult to code and deploy(due to the Python wrapper in the foreground). This makes it a great choice to perform computationally intensive programs.

2.1.2 Haar Cascade

The main aim of face recognition is to detect faces in the frame. Haar cascade is one of the fast, effective ways to achieve it. First published by Paul Viola and Michael Jones[4] in 2001, it is a machine learning approach to detect almost any object, but it mainly solves face detection problems.

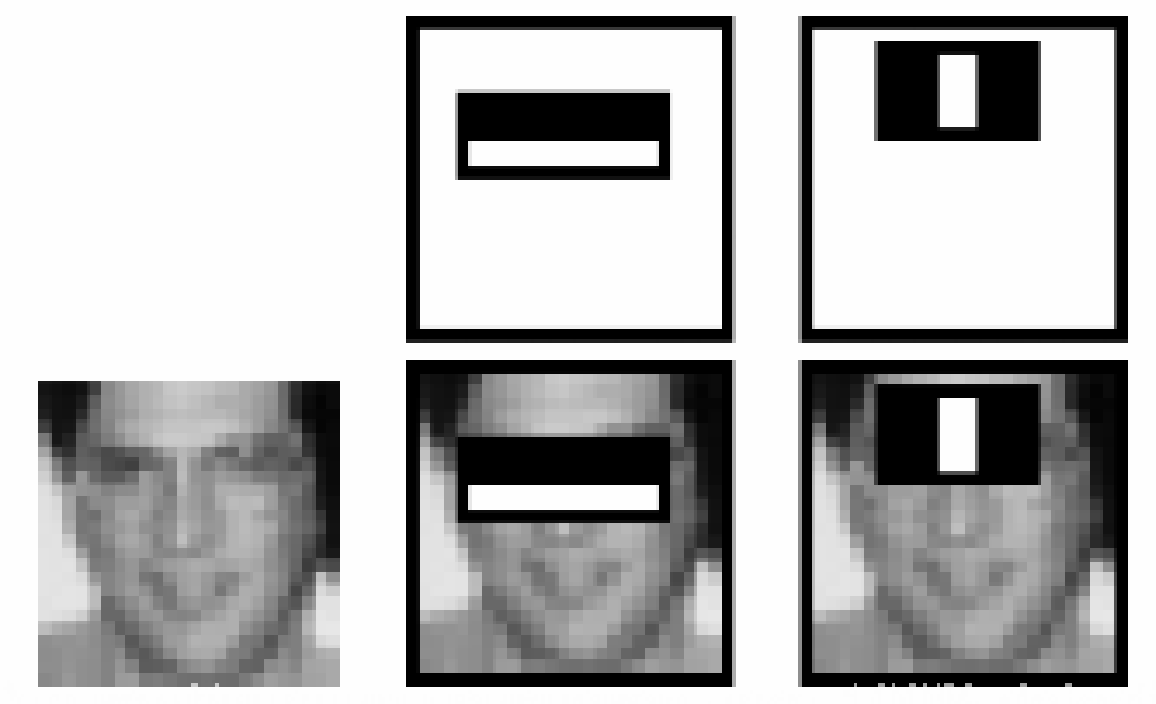


Figure 2.2: Haar Cascades

source: docs.opencv.org

Haar Cascade can understand using many **Haar** features and then using those features many times (**Cascade**) to build up a face detection. It has four steps: Haar Cascade can understand using many **Haar** features and then using those features many times (**Cascade**) to build up to face detection. It has four steps:

Haar Feature extraction

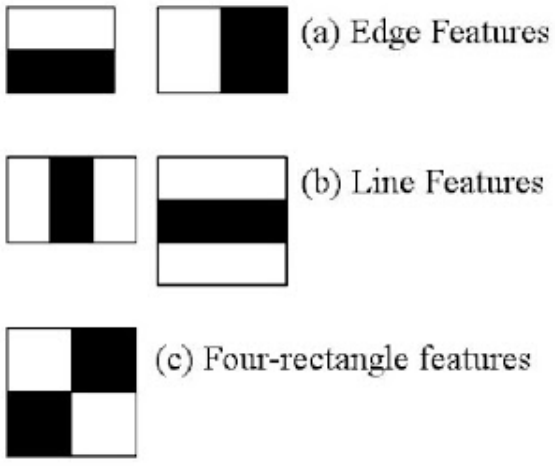


Figure 2.3: Four Haar Features

source: docs.opencv.org

First, we have to collect the Haar feature. As a feature to encode every domain is operate faster than pixel system, objects are mainly classified on many simple features. A **Haar Feature** is a calculation performed on adjacent rectangular regions on a specific location in the window. Haar Feature is shown in figure 2.3 below.

Integral Image Representation

Integral Images efficiency speed up the calculation of these Haar features in one pass over the image. It creates sub-rectangles and creates array references for each sub-rectangles instead of computing at every pixel—figure 2.4 followed by Zhang, Cha, and Zhang, Zhengyou [5].

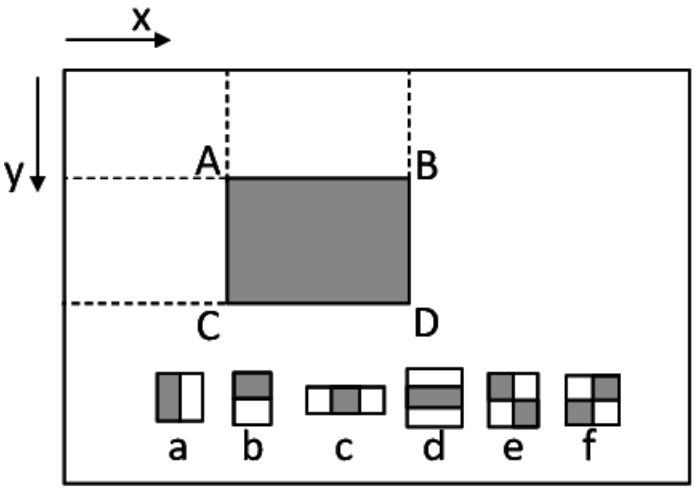


Figure 2.4: How integral image works

Adaboost Training

As we do not know which Haar classifier is the best fit, we have a solution to group every weak classifier into a robust classifier using **Adaboost** (adaptive boosting). It chooses the best features and trains the classifier to use them.

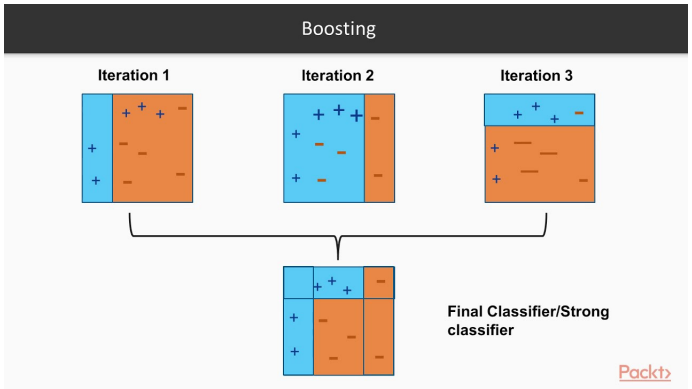


Figure 2.5: Adaptive boosting algorithm

source: Packt

Cascading Classifier Architecture

Paul Viola and Michael Jones ensured in their paper that employing a **cascade of classifiers** makes the algorithm performs faster. The cascade classifier consists of series of stages, and each stage contains a collection of strong classifiers. It removes the need to apply all features on a window simultaneously. Every time the sub-window slides into a window, if it detects the face, we move to the next step; else, we discard that region and slide to another part. The rejected region will keep at rejected sub-windows. The process is shown in diagram2.6, originally from Lee, Deok Gyu et al.[6].

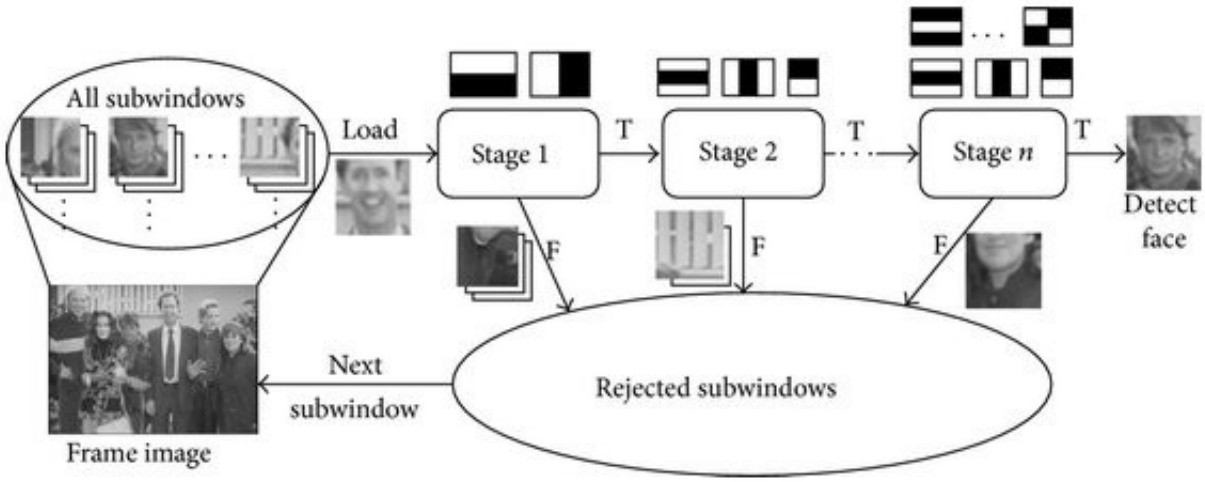


Figure 2.6: Cascade structure of Haar classifier

2.1.3 Multi-task Cascaded Convolutional Networks

Multi-task Cascaded Convolutional Networks (MTCNN) is a solution for both face detection and face alignment. Proposed by Zhang et al.[7], it is one of the most popular and accurate face detection tools today. This model has three Convolutional Neural Network (P-Net, R-Net, O-Net) connected in a cascade.

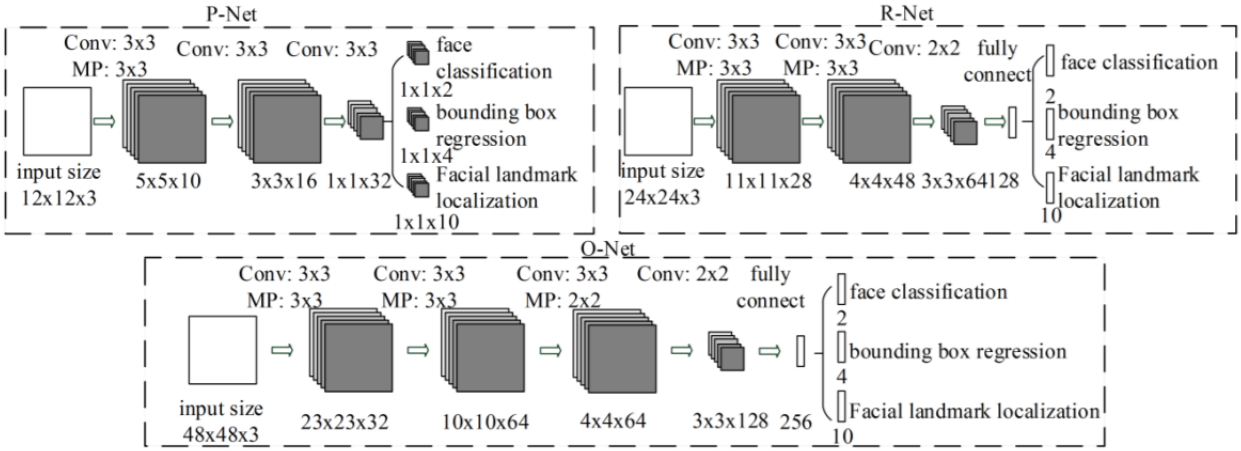


Figure 2.7: MTCNN structure

MTCNN can integrate both recognition and alignment because of multi-task learning. There are three-stage corresponding to three CNN. In the first stage, the Proposal Network (P-Net) quickly produces windows thanks to a shallow CNN. The Refine Network (R-Net) refines the proposed candidate windows into a more complex CNN to the second stage. Lastly, Output Network (O-Net) is more complicated than others, refining the result and showing the facial landmark positions.

2.1.4 One-shot learning

One-shot learning is a supervised algorithm that only needs one or only a few pictures for each class. We use a simple CNN algorithm to predict who it is from the input picture of one class.

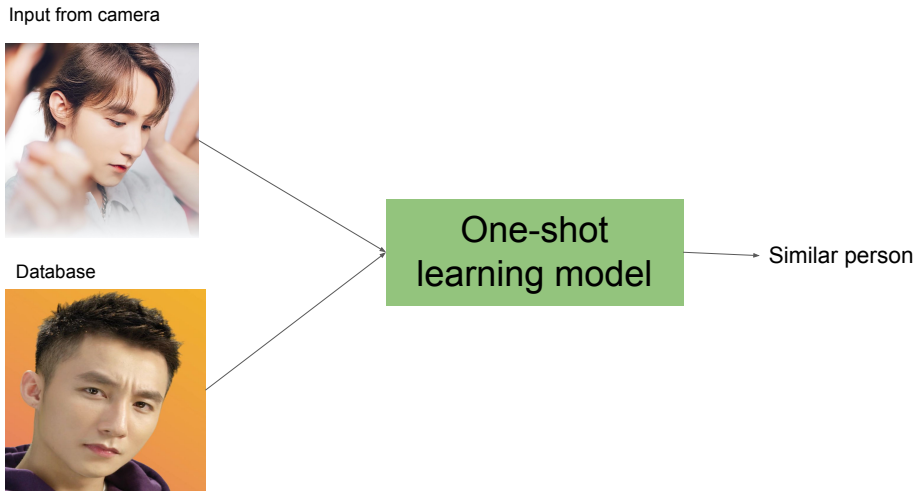


Figure 2.8: One-shot learning example

This method will save much man's power and power consumption. Because we only a little picture, every time newcomers show up, we will make it easier by taking only one picture of them. And the process of creating embeddings for existing images is also taking less time than the traditional way that is training a model with thousands of pictures for each class. We can scale up the system for a large number of classes quickly.

2.1.5 Learning similarity

Learning similarity is based on a distance calculation between two pictures, usually a ℓ_1, ℓ_2 norm so that if that is the same person, the distance will be smallest, else it will be biggest:

$$\begin{cases} d(img1, img2) \leq \tau \Rightarrow \text{same} \\ d(img1, img2) \geq \tau \Rightarrow \text{different} \end{cases}$$

When using this method, we have to choose a threshold to decide if this picture is the same or different. For example, in figure 2.9, if the image from the left has a distance smaller than the threshold, which is 0.2, then it is the same picture:

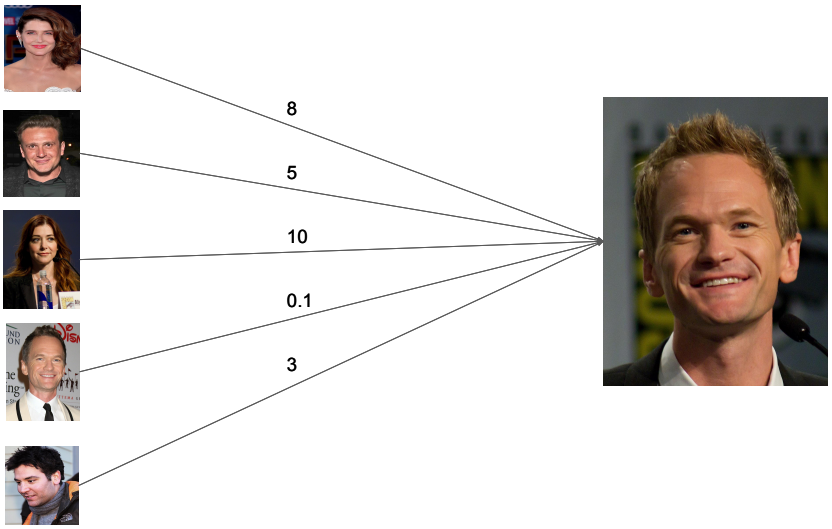


Figure 2.9: Learning similarity method

We can see that learning similarity has the advantage that we can find the similarity picture without re-training the model again. Besides, it will not depend on the number of classes. So we do not have to train the model again when there is a new class.

2.1.6 Siamese neural network

Siamese network was first introduced by Yaniv Taigman et al.[8]. It is a network architecture that can answer if two pictures are the same person in it. The architecture of a Siamese network is a Convolutional Neural Network without an **output layer** to encoding a picture to an embedding vector. The input of a Siamese network is two random pictures chosen from the database. After the network handling it, it returns two vectors corresponding to two input pictures. The **loss function** is responsible for calculating the distance between two vectors to show the difference between them. Usually, the loss function in this network is a ℓ^2-norm function.

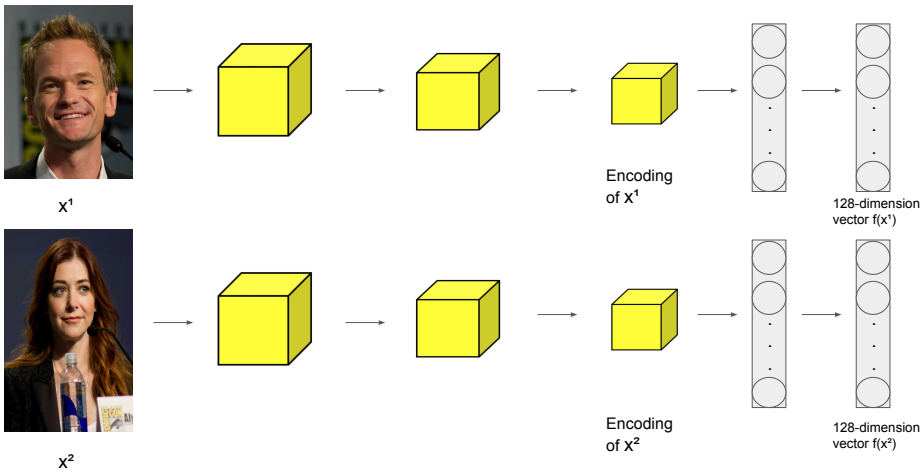


Figure 2.10: Example of Siamese network

We conclude if two pictures were the same person by this function:

$$\delta(x_1, x_2) = \begin{cases} min||f(x_1) - f(x_2)|| & \text{two person are the same} \\ max||f(x_1) - f(x_2)|| & \text{two person are different} \end{cases}$$

2.1.7 Convolutional Neural Network

Convolutional Neural Network (CNN) is a Deep Learning neural network that can take structured arrays of data like images, analyzing properties, aspects from the input images to differentiate one from the other. The name Convolutional Neural Network shows that it employs the mathematical convolution operation. We can stack up layers inside it for our purpose. For example: with three layers, the network can recognize handwriting; with 25 layers, the network can recognize human faces. We can add more layers for extracting and recognizing more features.

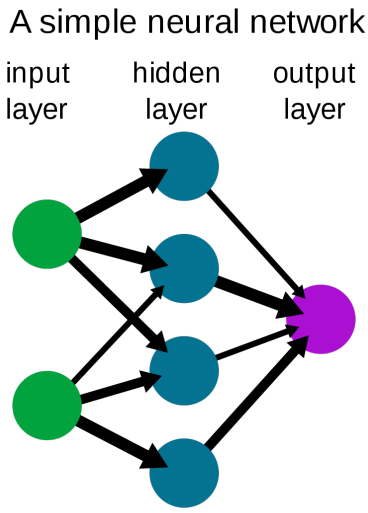


Figure 2.11: Simple neural network

source: Wikipedia

The CNN was typically built by three main layers:

- **Input layer**: takes input images then fits them into the network.
- **Hidden layers**: contains layers that perform a specific task.
- **Output layer**: decide which one belongs to which class.

We can control how many layers are inside hidden layers, but there are three main layers in it:

- Convolutional layer: This layer extracts features from the image by taking the dot product between the image and a set of a learnable parameter known as the kernel. The kernel is smaller than the image but more in depth.

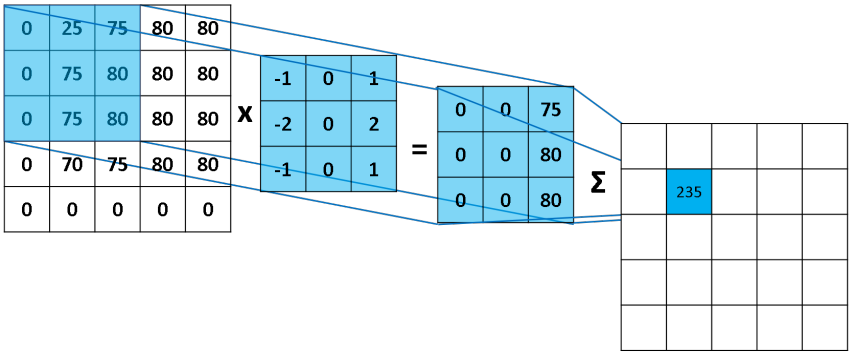


Figure 2.12: Convolutional layer

During the forward pass, the kernel slides across the image, creating a two-dimensional image representation. The sliding size of the kernel is called **stride**. The bigger stride, the faster the kernel slide across the image.

It can be seen that the feature map should be smaller if the kernel size is bigger. To avoid reducing feature map size after every convolutional layer, we have **padding** with value zero for having the image and feature map in the same size. The model contains many feature maps for extracting features from the input. It includes a stack of feature maps with different kernel values to extract clear features.

- Activation layer: The Activation layer is put at the end or between the Convolutional Neural Network. It helps to calculate the argument for the next layer. We have different activation functions like Rectified Linear Unit (ReLU), sigmoid, and softmax. Commonly ReLU is used after the Convolutional layer, sigmoid and softmax are used at the classification output layer.

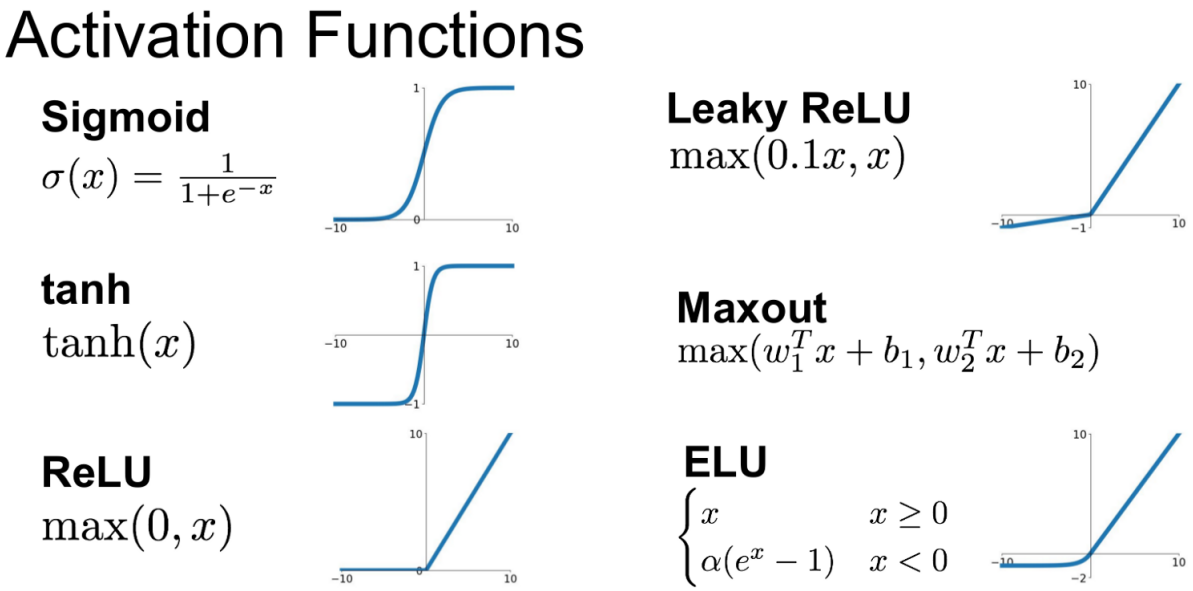
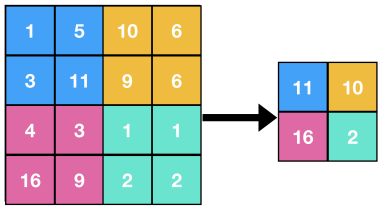


Figure 2.13: Activation functions

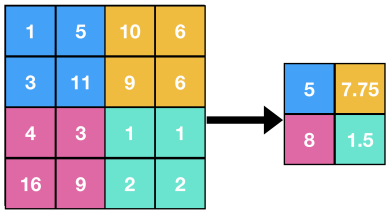
- Pooling layer: The pooling layer replaces the output of the network at some locations by deriving summary statistics of nearby outputs. It reduces the number of parameters and computations in the network, spatial size of the network to control overfitting.

  There are two operations in this layer:

  – Max Pooing: like its name states, it takes the max value from a pool.



  – Average Pooing: it takes the average values from all value from a pool.



- Fully Connected layer: Neurons in this layer have a complete connection with all neurons from the previous layers. It can be computed by matrix multiplication followed by a bias offset. Fully Connected layers compile data extracted from earlier layers to get the final output.

- Dropout layer: Randomly drop connection from the previous layer.

- Batch Normalization: Normalize and scale data after each activation.

The combination of all these above layers is the basic concept of the CNN model, as in figure 2.14.

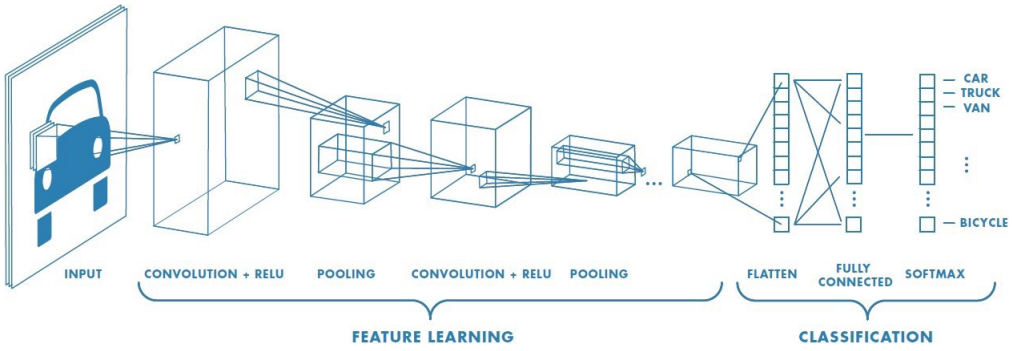


Figure 2.14: Basic concept of CNN.

2.1.8 Flask

Flask is a micro web framework written in Python. Flask takes advantage of the simplicity of Python, which makes it easy to build a primary application to backend APIs. It is classified as a micro-framework because it does not require any libraries.

It is also called a Web Server Gateway Interface Web Application (WSGI) framework. Flask gives us many choices when developing web applications and provides the necessary tools to build and deploy a web. Here are some of the merits of using Flask:

1. **Easy to use**: Flask framework is easy to understand. The simplicity in the framework helps us to navigate around and create applications easily.

2. **Very flexible**: Most of the components of Flask can be altered. It allows users to customize the website.

3. **Testing**: It allows unit testing through its integrated support, built-in development server, fast debugger, and RESTful request dispatching.

2.2 Methods

2.2.1 Building a Face recognition system

FaceNet

In 2015, Florian Schroff et al.[3] introduced FaceNet. It transforms the face image into 128 dimensions Euclidean space, similar to word embedding. Once the FaceNet model having been trained with triplet loss for different classes of faces to capture the similarities and differences between them, the 128-dimensional embedding returned by the FaceNet model can be used to clusters faces effectively. It is the backbone of many open-source face recognition models like OpenFace[1], facenet[2], etc.

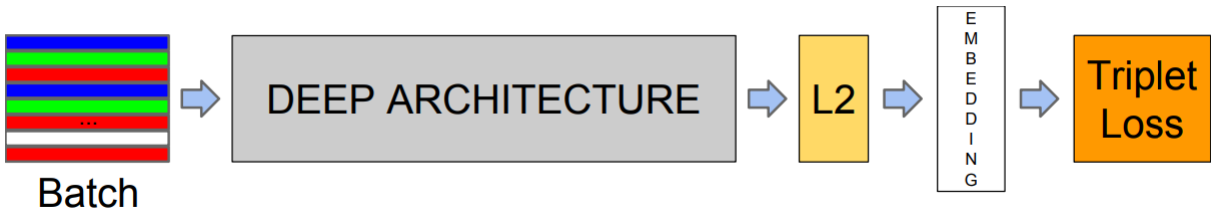


Figure 2.15: FaceNet Architecture

FaceNet has solved two problems in face recognition algorithm before it:

1. Apply a CNN and use 128-dimensional data to not need a bottleneck layer for reducing data dimensionally.

2. Loss function can learn the similarity between two pictures in the same class or distinguish two pictures in different classes simultaneously.

This model outputs an embedding of image $f(x)$ with L_2 normalization on it. After that, these embeddings pass into a loss function to make the squared distance between two images is small when two images belong to the same identity, whereas the squared distance will be significant. This loss function is called **Triplet loss**.

Triplet loss

The encoding of the Convolutional Neural Network helps us encode the picture into a 128 dimensional vector $f(x)$ called embedding. It is normalized such that:

$$||f(x)||_2^2 = 1$$

To implement Triplet loss, we need to take three pictures, including: anchor picture (x_i^a), positive picture (x_i^p) which is a picture of the same person), negative picture (x_i^n) which is from the different person, to satisfy that:

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

[1] `https://cmusatyalab.github.io/openface/`
[2] `https://github.com/davidsandberg/facenet`

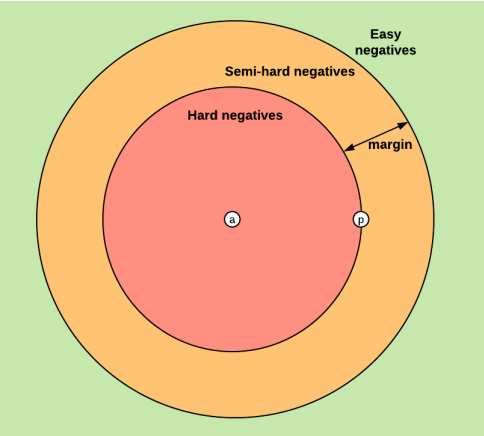


Figure 2.17: Chossing hard triplets

α is an enforced margin to differentiate between positive and negative pairs. So that the loss function can be defined:

$$\mathcal{L}(x^a, x^p, x^n) = \Sigma_i^n[||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha]$$

Figure 2.16: Triplet loss learning

If the above function can easily be satisfied by choosing close triplets, it would not help the training. So choosing triplets that violate that function is essential.

Triplet images input selection

If the chosen triplets were easy to distinguish, the learning images would not make any sense. We need to choose hard triplets to make the model harder to learn and help the model discriminate effectively between faces.

This mean that for a given x_i^a, we need to choose 2 pairs of anchor-positive and anchor-negative to please:

1. Hard positive: $\arg\max(|f(x_i^a) - f(x_i^p)||_2^2)$

2. Hard negative: $\arg\min(||f(x_i^a) - f(x_i^n)||_2^2)$

Computing **Hard positive** and **Hard negative** can do on the previous checkpoint or do it on every mini-batch to minimize computationally expense when generating the whole training data set. Choosing triplets on purpose can make the model giving the result more accurate.

Experimental with pre-trained FaceNet model

Based on Siamese neural network, we will choose a based network and drop out the output layer. VGG16 is one of the most powerful networks in the classification topic. It contains multi-layers of Convolutional, Pooling, and Fully Connected layers. The input runs through two convolutional layers with 64 filter channels of 3*3 kernel with the same padding from the input. Following a Max Pool layer of stride 2*2, two layers have convolution layers of 256 filter channels with filter size 3*3. Then there is a Max Pooling layer of stride 2*2, following by two convolution layers of filter size 3*3 and 256 filter channels. There are two sets of 3 convolution layers and a max pool layer, and each has 512 filters of 3*3 sizes with the same padding. After all, the data is passed through three fully connected layers.

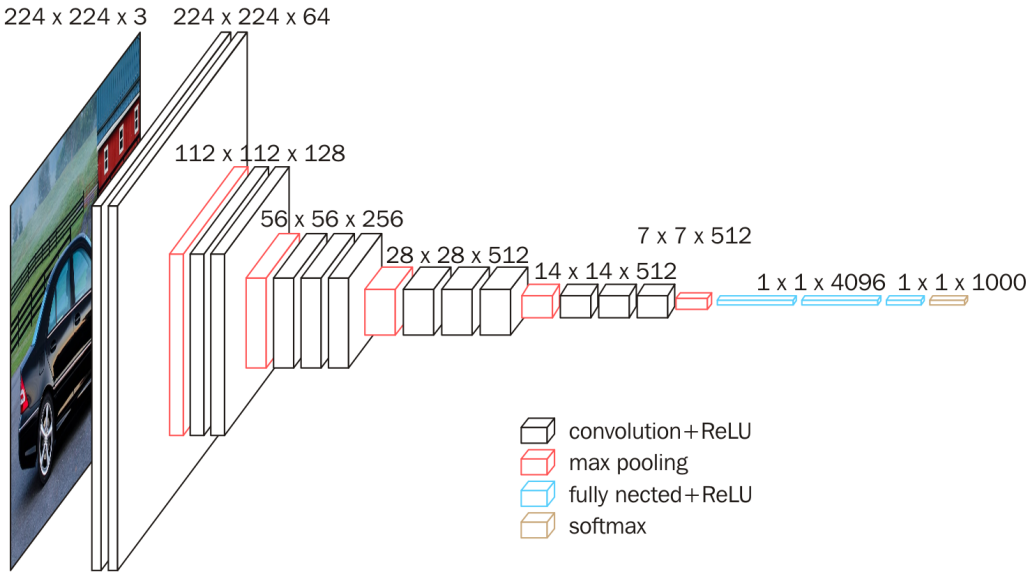


Figure 2.18: Architecture of VGG16 network.

source: towardsdatascience.com

After getting the base network, we add the Triplet loss function to determine the input belongs to which class. To save training time, we are using the weights from OpenFace[3] model. We will train the model with VN-celeb dataset, published by Sun* AI team, to use this with Vietnamese people.

Making the recognition system

We are using the trained model, OpenCV, and MTCNN to create the system. To avoid using an image to spoof the system, we are setting up two lights from behind the camera. If someone came with a picture, the system would not detect it.

First, we are connecting the camera to the server. The server has a job that captures every frame that contains the human face with the help of OpenCV. The model implemented on the server creates embedding vectors for every registered face in the database. Moreover, these vectors are stored only in the running state and deleted after the system stops to avoid outside access.

[3] `https://cmusatyalab.github.io/openface/`

After that, the system creates an embedding vector for the current face that shows at the camera, which is detected using Haar Cascade. Compare with a recent detecting model like dlib[4], MTCNN[7], OpenCV DNN Face Detector, Haar Cascade shows some outdated and worst result in detecting faces in a picture that has many faces in that or large picture size. Besides, it also depends much on lighting, face rotation, and the quality of the image. But during the recognition, there is only one person at the camera at that time. And faces in the database with face in the camera are equal in lighting and quality. So using the Haar Cascade is the best option for us at the detecting time.

Figure 2.19: Comparison along with face detection model

source: Vardan Agarwal - towardsdatascience.com

Since we got vectors from the database and the vector from the camera, the system will calculate the Euclidean distance with these vectors to find the smallest compare with the threshold we choose.

After the recognition is finished, the liveness test will continue. The liveness test is for avoiding people using a printed image to trick the system. We are using MTCNN to extract key points from the face. It will store the default key points of the face. MTCNN detector will track the key points. When it satisfies the math function of calculating the distance of eyes and nose, the system will request the Central Control Unit.

[4] `https://github.com/davisking/dlib`

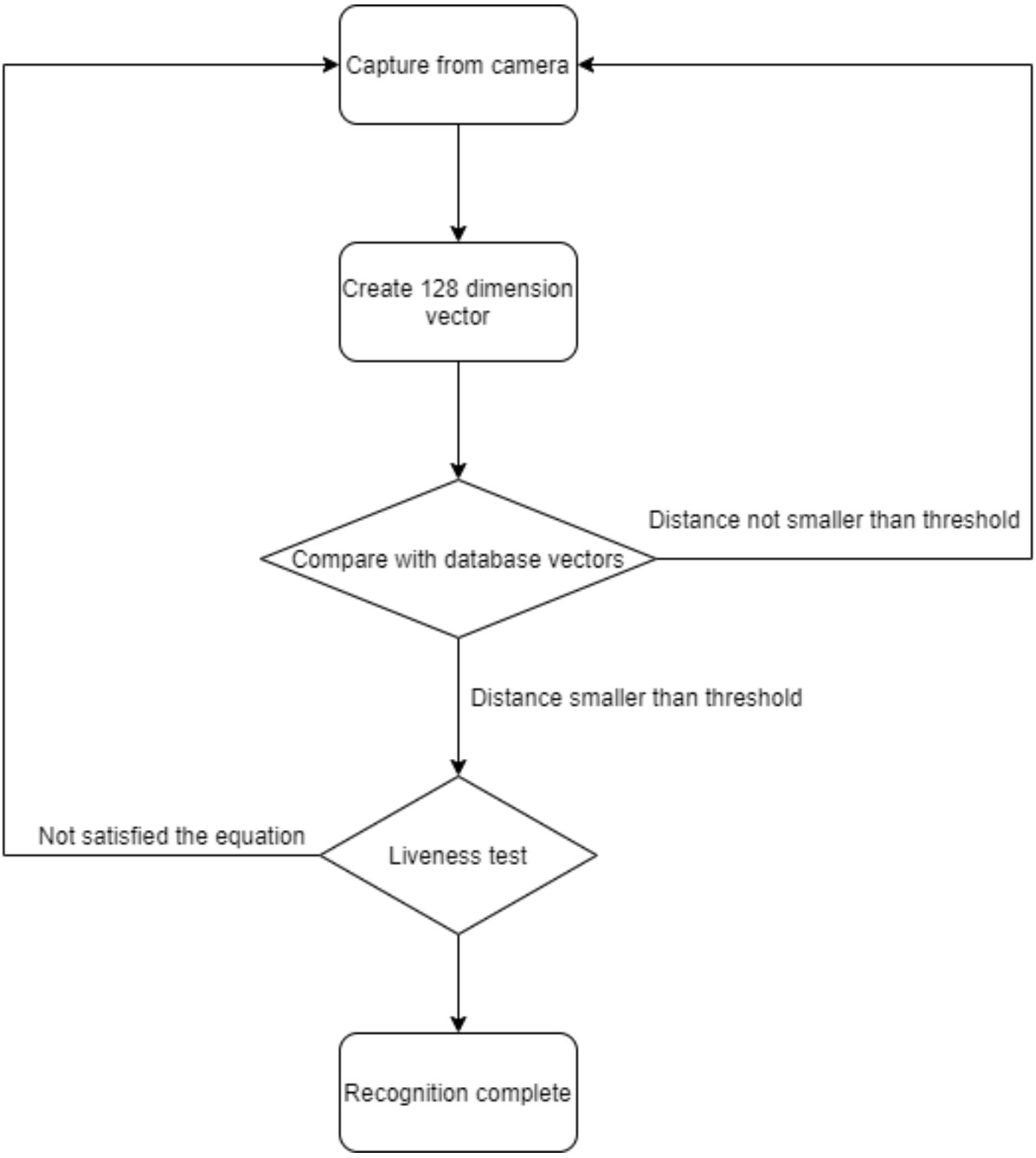


Figure 2.20: Recognition system data flow

2.2.2 IoT devices setup and API connection

IoT devices setup

IoT is one of the buzzwords in the technology industry in the past years, stands for Internet of Things. IoT means all of the things that are connected to the Internet and can be controlled remotely. According to Wikipedia[5]: "..the network of physical objects - devices, vehicles, buildings, and other items - embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data."

The main factor in the upwards trend in IoT systems is DIY (do it yourself) electronic platforms like Raspberry Pi and Arduino. Raspberry Pi (figure 2.21) has many advantages: small, inexpensive, easy to use, etc. And after all, it allows us to connect various accessories via a 40-pins GPIO connection built-in on board.

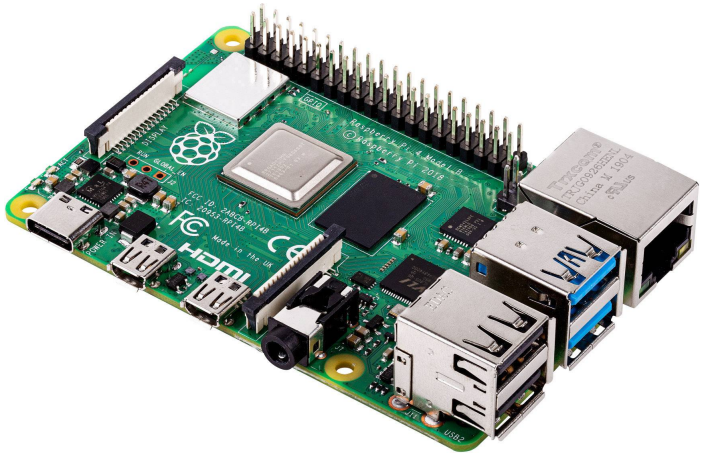


Figure 2.21: Raspberry Pi board

To interact with all IoT devices, we use Raspberry Pi version 4 with a specification in table 2.1 as a CCU(Central Control Unit).

Table 2.1: Central Control Unit specification

OS	Raspberry Pi OS
Kernel version	5.10
CPU	Quad-core Cortex-A72 (ARM v8) 64-bit Soc
RAM	4GB

Since the limitation of the voltage output of the CCU device, we are controlling one electric door lock and two light bulbs. The diagram of connecting electric door lock is in figure 2.22

[5] https://en.wikipedia.org/wiki/Internet_of_things

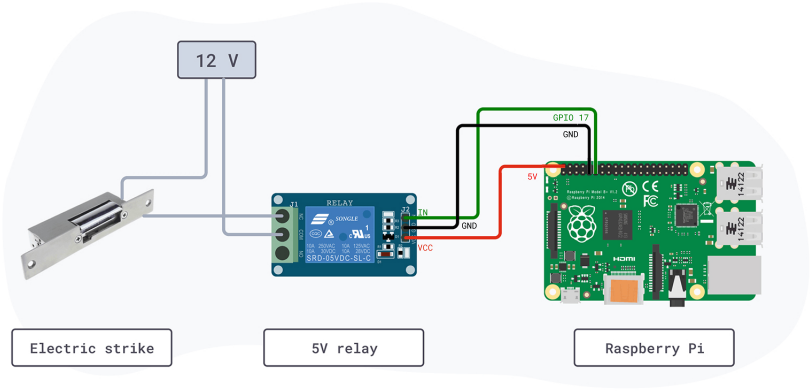


Figure 2.22: Electric door lock connection diagram

We are using a 12V DC lock. So we connect the lock with a 12V external power, which is three 3.7V 18650 batteries. We are using a 5V relay for controlling the switch with code. The connection between CCU and relay: connect **5V GPIO pin** to **VCC** for power to the relay, **GND pin** to **GND** for ground, **GPIO 17** to **IN** for controlling.

The central part of controlling the lock is connecting the lock and external power to the relay. Here we connect the lock to **NC** (Normally Closed); External power to **COM** (COMMON). The COM connection is the moving part of the relay. We set the initial state of the INPUT from CCU as **GPIO.low**, High/Low switch on the relay to **High**. This setup works from the initial setup before; If the input from CCU is low, then the lock does not have any power equal to lock state; else, if the input is high, then the lock has power equal to unlock state. We are setting the lock state to **GPIO.low**, unlock state to **GPIO.high**.

For light bulbs, the setup is much simpler:

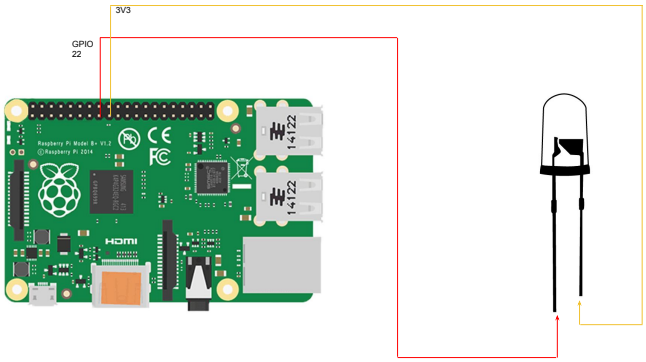


Figure 2.23: Light bulbs connection diagram

We connect one pin to **GPIO 22** pin to control, the other to a **3V3** pin for power. We set automation for every unlock, and the light turns on for two minutes for a home scenario.

Google Assistant SDK

The best way to set up a smart home with IoT devices is automation. Because of the limitation of our CCU, adding Google Assistant SDK to it will extend the ability to connect more smart devices using the Google Home network.

Google Developers team has clear documentations[6] about how to implement it on a Raspberry Pi using Python.

API connection

To connect a server for Face recognition to Central Control Unit, we create an Application Programming Interface (API) using Flask to link these two components.

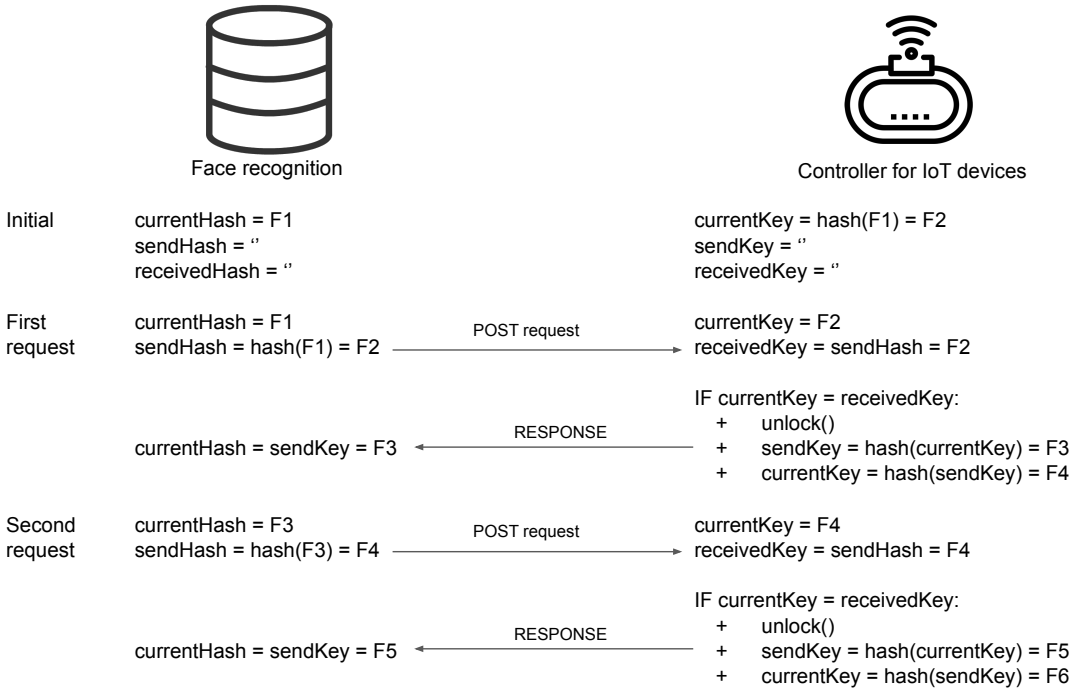


Figure 2.24: API connection flow

We are applying a blockchain-like method on API for security. Both server and CCU has two different initial private key. For example, the initial private key on the server is F1, so the initial of CCU is SHA-256 encode of F1. After every request, a private key that store on both side change, as shown in figure 2.24. If someone can hack to the connection, they neither can trace back to the first private key nor find out the encoding algorithm.

Latency in this system is also a problem to be a suitable replacement in real life since they share the same local network, the system work in almost real-time.

[6] `https://developers.google.com/assistant/sdk/guides/service/python#steps`

Chapter 3

Result and Discussion

3.1 Experimental scenarios

Our evaluation is performed on a MacBook Pro as the Face recognition system; Raspberry Pi version 4 as the Central Control Unit to control IoT devices. Table 3.1 summarizes the hardware configuration. The Central Control Unit hardware configuration is mentioned in table 3.2.

Table 3.1: Face recognition server configuration

OS	macOS Mojave 10.14
CPU	Intel(R) Core(TM) i7-3720QM
RAM	16GB
GPU	NVIDIA GeForce GT 650M (1GB VRAM)

Table 3.2: Central Control Unit specification

OS	Raspberry Pi OS (Linux based)
Kernel version	5.10
CPU	Quad-core Cortex-A72 (ARM v8) 64-bit Soc
RAM	4GB

As face recognizing is memory intensive, Raspberry Pi can not compute as fast as on a server. So that to have a better user experience, we are choosing to implement the model on a MacBook Pro for testing. On a retail product, we may have to build up a server for each client.

Figure 3.1 shows the device setup diagram for evaluating, testing, and also in the public product:

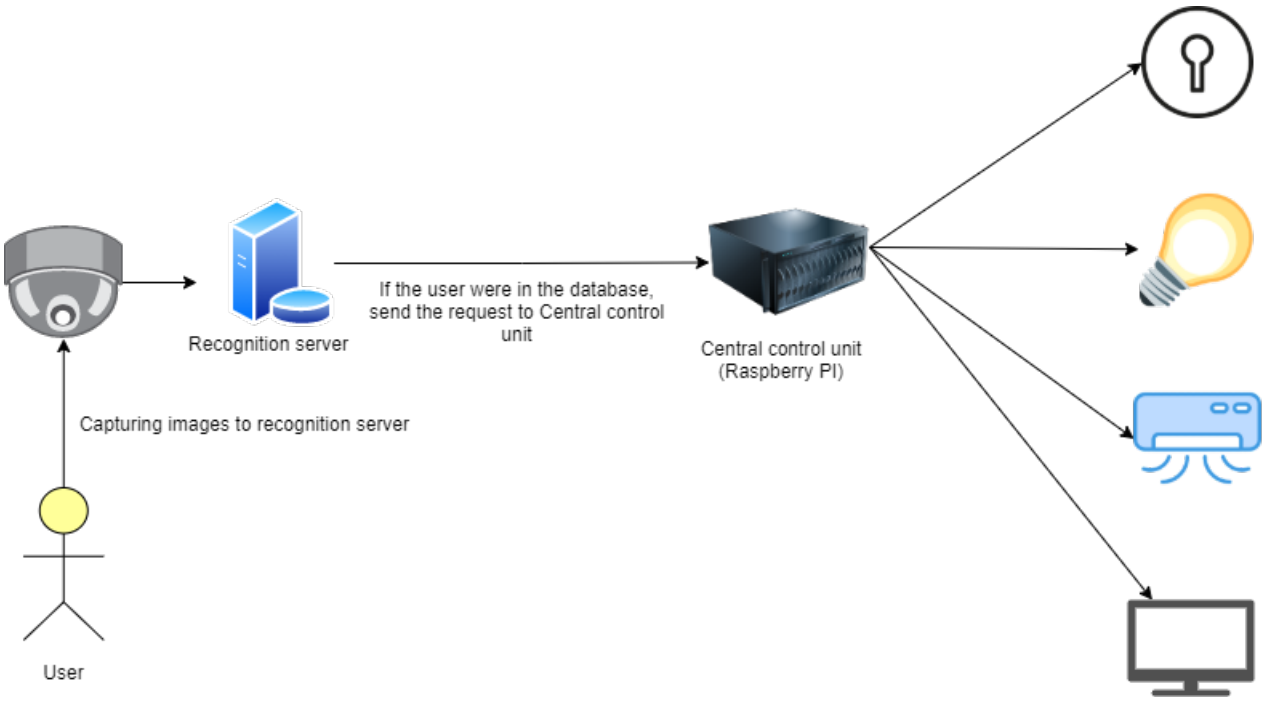


Figure 3.1: Product flow

3.2 System evaluation

3.2.1 Recognition system

We are evaluating the model as the performance of the recognition system. The Confusion matrix is built to evaluate how well the model is.

Table 3.3: Confusion matrix

	Actual Negative	Actual Positive
Predict Negative	True Negatives (TN)	False Negatives (FN)
Predict Positive	False Positive (FP)	True Positive (TP)

- **Accuracy**: Accuracy is the value that shows how accurate the model is, overall the class.

$$Accuracy = \frac{TN + TP}{TN + FN + TP + FP}$$

- **Precision**: Precision defines how good the model is when predicting a class if the precision is low, meaning your model makes many mistakes in predicting.

$$Precision = \frac{TP}{TP + FP}$$

- **Recall**: Recall(or Sensitivity) shows the accuracy of your model on the positive case, so the higher Recall, the better your model.

$$Recall = \frac{TP}{TP + FN}$$

- **F1 score**: F1 score show the balance between Precision and Recall

$$F1 = 2 \times \frac{Precision * Recall}{Precision + Recall}$$

Table 3.4: Evaluation with confusion matrix

Accuracy	Precision	Recall	F1 score
0.983740	0.986542	0.982919	0.983147

We test the effective of the model by using a set of test images that includes many classes. Then the model will extract the embedding vectors for each image. After getting vectors, we use t-sine for reducing the dimensional of the vectors then visualize it.

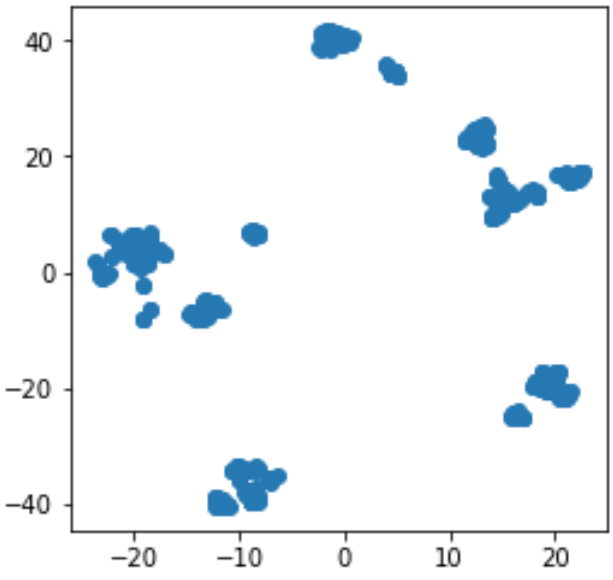


Figure 3.2: Scatter plot of the embedding vectors

For a better visual, we will show it on TensorBoard. The model works flawlessly on moving the picture of a same person together.

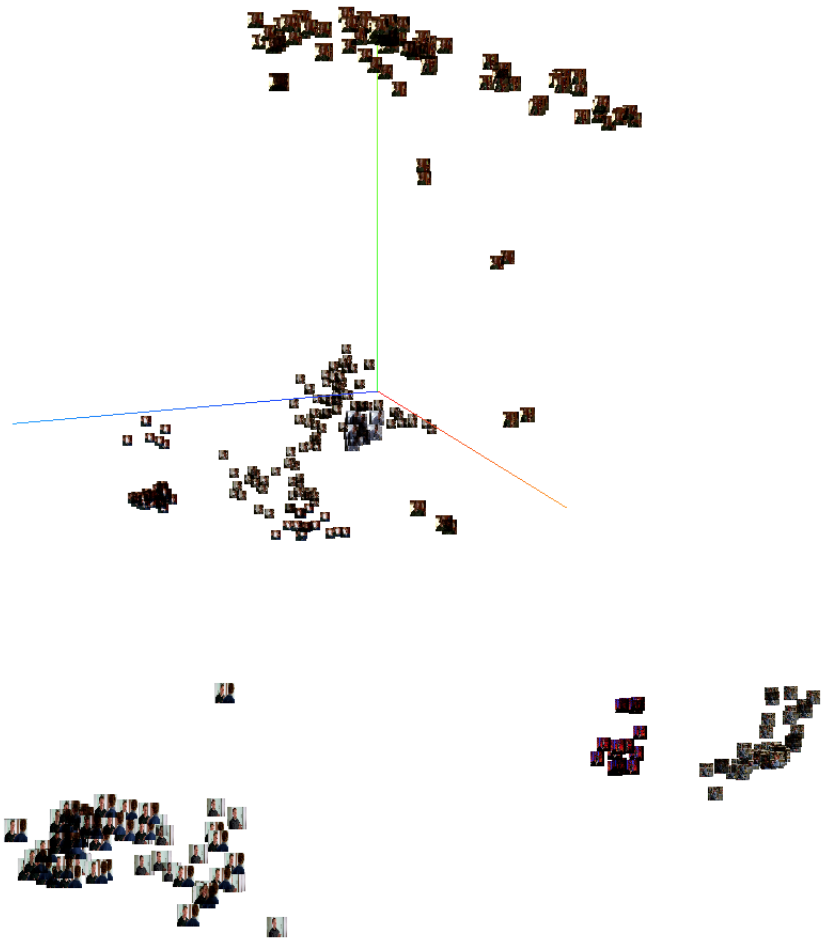


Figure 3.3: TensorBoard visualisation of embedding vectors

We applied the system onto our home on real-time usage, and it has an estimated 95% accuracy based on successful unlock, failed to unlock.

Table 3.5 shows the time of processing for each scenario (we choose one picture for each class):

Table 3.5: Processing time for each scenario

Number of the registered person	Processing time(seconds)
3	2.17
100	39.18
4000	313

The processing time above indicates the time of calculating pictures to 128 dimension vectors. It can improve by using better hardware for less executing time.

3.2.2 Central Control Unit and API performance

The performance of CCU is calculated on the percentage of success time and failed time. For most of the testing time, the system runs perfectly with a 100% success rate.

About the API connection, the only network connection can affect its performance. Since we shared a local network on both devices, API runs without any delay.

3.3 Discussion

In everyday usage, our system works without any problems. But compare with other products on the market, there are some drawbacks to our approach:

Inaccuracy with an extensive database

On our evaluation, with an extensive database (more than 4000 people), the model seems not to recognize as accurately as before. Many elements can affect the performance, but we think that in a sizeable registered person, there will be a small percentage that two people can have a similar face landmarks as well as similar Euclidean vectors. That leads to an equal distance when compare them to each other.

We think that our model is not trained enough to extract more precise face landmarks. This problem can be solved by a well-trained model with a large dataset.

Set up the system on multiple devices

Many public products by large company that integrated all elements on only one device like in 3.4. Compare with our product in 3.5, we have to implement it on two devices: one for recognition and one for control the lock, as well as other smart home devices.

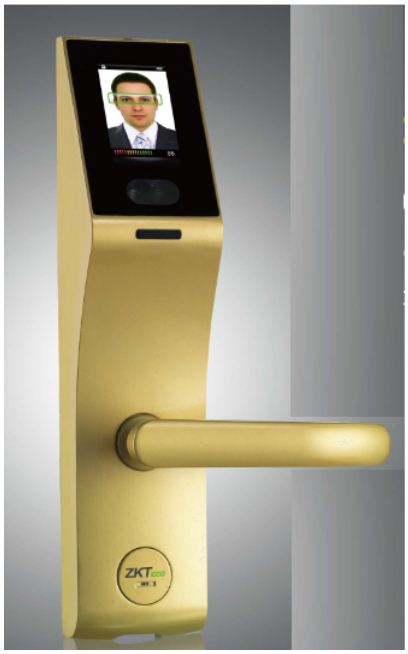


Figure 3.4: An example of an on-sale product

We have tried to implement the recognition system on the CCU. Still, since it was relatively computationally and memory intensive, the recognition has the worst performance when running on the CCU.

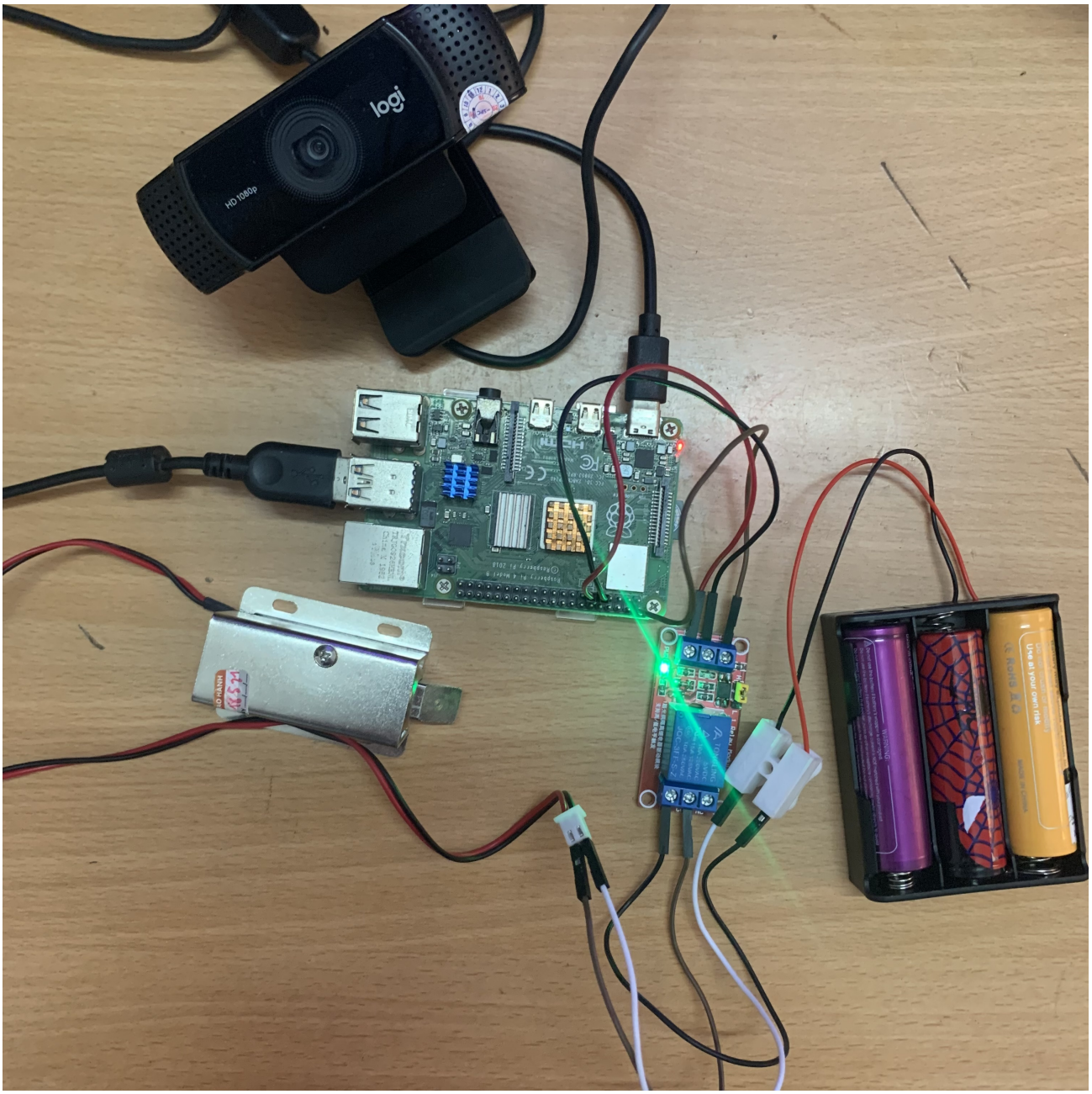


Figure 3.5: Our door lock setup

Chapter 4

Conclusion and Future-work

Face recognition is a new security solution. Despite that many companies still prefer traditional ways for security like door locks using keys or fingerprint locks, face recognition still shows its strength in the easy to use and fast for customers. Looking to a post-COVID-19 world, we believe that the investment in a facial recognition system now will be cost-effective in the long run.

Three months of internship is precious for understanding many concepts of machine learning. Besides the work I have done, I have been thinking of improving the model and expanding the system. Rather than using only one photo for each class, we can apply Data Augmentation to broaden the training dataset but not making the user inconvenient in taking many photos. And I am thinking of creating a model smaller that can fit a mobile app. That way can expand the usage of this work to many devices.

Bibliography

[1] Esteban Ortiz-Ospina Max Roser, Hannah Ritchie and Joe Hasell. Coronavirus pandemic (covid-19). *Our World in Data*, 2020. https://ourworldindata.org/coronavirus.

[2] Harvard Health Publishing. Preventing the spread of the coronavirus, May 2021.

[3] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *CoRR*, abs/1503.03832, 2015.

[4] Paul Viola and Michael Jones. Rapid object detection using a boosted cascade of simple features. volume 1, pages I–511, 02 2001.

[5] Cha Zhang and Zhengyou Zhang. *A Survey of Recent Advances in Face Detection*. 01 2010.

[6] Mooseop Kim, Deok Gyu Lee, and Ki-Young Kim. System architecture for real-time face detection on analog video camera. *International Journal of Distributed Sensor Networks*, 2015:1–11, 05 2015.

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

[8] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. 09 2014.