Ho Chi Minh National University
UNIVERSITY OF TECHNOLOGY
Electrical - Electronic Engineering
---------------o0o--------------

FINAL REPORT
SMART DOOR SYSTEM USING FACE RECOGNITION
INTERNSHIP 2024

INSTRUCTOR: MR. TRƯƠNG QUANG VINH
NAME: NGUYỄN PHẠM MINH KHÔI
ID: 2151105

HỒ CHÍ MINH CITY, JULY 27TH 2024

THANK YOU!

With a short nine-week internship, I have learned a lot of useful knowledge and improved my existing professional knowledge. Therefore, I realize that internship and practical experience in studying are extremely important for students. This also helps me learn many necessary skills when working and consolidate my knowledge.

To complete the internship report, I received help from Mr. Truong Quang Vinh, who created conditions, introduced me to the company and instructed me enthusiastically. Moreover, I would like to thank Mr. Quoc for helping and guiding me in my work at the internship company. Thank you Duy Anh SMS Company for creating conditions for me to complete the task. Thanks to the guidance and instruction of Mr. Quoc, Mr. Quoc and Duy Anh SMS Company, I have achieved the results I have today.

However, my internship report is short due to the general objective conditions, so there are certainly many shortcomings. I really hope to receive valuable comments from you to further consolidate my knowledge in this field. Thank you very much!

Ho Chi Minh City, August 29th, 2024.

Student

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

1.1 Introduction to company

Duy Anh System Management Solution Co., Ltd. is a company specializing in providing comprehensive ERP hardware & software solutions for customers who need digital transformation and scale expansion in all manufacturing and business sectors.

Enterprise Resource Planning (ERP) is a software that manages business operations integrating all aspects of production and business such as planning, production, purchasing, sales, etc.

ERP supports business processes with the goal of centralizing and optimizing information, connecting and sharing data between business departments, between production and business activities, and with suppliers and customers. ERP provides an overview of all operations with full visibility of the organization's information and data.

1.2 Mission of Internship

- Content 1: Learn about Software and XAF
- Content 2: Design a real embedded system
- Content 3: Evaluate and apply the system in practice

1.3 Internship time and schedule

Students clearly state the internship time at the company, describe the internship schedule.

Internship time: 2 months (from June 18, 2024 to August 13, 2024)

Internship schedule:

- Week 1 (June 18 - June 24): Learn about XAF, Database and SQL.
- Week 2 (June 26 - July 1): Come up with ideas to design an automatic door system using CNN recognition system.
- Week 3 (July 2 - July 8): Write the operating block diagram and basic algorithm diagram of the system.
- Week 4 (July 9 - July 15): Learn about how the ESP32-Cam module works.
- Week 5 (July 16 - July 22): Learn about CNN.

- Week 6 (July 23 – July 29): Write programming code for ESP32-Cam and create interface using Visual Studio.
- Week 7 (July 30 – August 5): Write code for training face recognition.
- Week 8 (August 6 – August 12): Build real system and synthesize results.

2. CONTENT

Sinh viên trình bày chi tiết các công việc thực tập

2.1 Overview

In today's digital age, security and safety are important factors for many areas, from homes, offices to public facilities. One of the widely applied advanced security methods is the facial recognition system, a form of biometric authentication. This topic focuses on developing an automatic door system based on facial recognition technology using a Convolutional Neural Network (CNN) and an ESP32-CAM camera.

The use of traditional authentication methods such as mechanical locks, magnetic cards, or PIN codes is becoming obsolete due to their vulnerability and insecurity. In this context, facial recognition technology has emerged as a modern security solution, which does not require users to carry any physical devices and is more difficult to fake. This project aims to solve the security problem by developing an automatic door system that can recognize faces quickly and accurately, helping to improve user experience and enhance security.

The main objective of the project is to design and build an automatic door system using facial recognition technology to authenticate users. Specifically, the system will:

- Recognize and authenticate the face of the user registered in the database.
- Automatically open the door when a valid face is detected.
- Ensure high security and reliability during operation.

This system combines many modern technologies to achieve its goals:

- Convolutional Neural Networks (CNNs): CNN is a type of neural network specifically designed to process and analyze image data. With the ability to recognize and classify features in images, CNN is the foundation for accurate face recognition.
- ESP32-CAM: ESP32-CAM is a compact module that integrates a camera and Wi-Fi connectivity, allowing the system to take photos and send data to the server easily.

- Flask: Flask is a lightweight web framework used to build APIs that help connect and process data between user applications and databases.
- SQLite: SQLite is an embedded database management system, used to store facial information and personal information of users in the system.
- UI: The user interface in the facial recognition automatic door system plays an important role in providing an easy and efficient way for users to interact with the system.

This facial recognition and automatic door control system is designed to improve security and convenience, while providing a flexible solution for access control in different environments.

The facial recognition automatic door system consists of three main components: hardware, software, and facial recognition model. Each of these components plays an important role in ensuring the efficient and accurate operation of the system:

Hardware:

- ESP32-CAM:
  - Function: ESP32-CAM is a module that integrates both a camera and Wi-Fi connection, allowing for capturing and transmitting image data over a wireless network. This module is used to collect images of the user's face and send them to the server for processing.
  - Features: ESP32-CAM supports high-resolution image capture and is capable of transmitting data via HTTP protocol, which makes it easy to integrate with software systems and API servers.
- Door Opening Sensor:
  - Relay Module: Relay module is an electronic component that allows control of other electronic devices. In this system, it is used to control the opening of the door when receiving commands from the software.
  - Electronic Lock: Electronic lock is attached to the door and is controlled by the relay module. When receiving a signal from the authentication software, the electronic lock will open the door.
- Power Supply:

  - Function: Provides power to the entire system, including ESP32-CAM, relay module, and electronic lock. Ensuring a stable power supply is essential for the system to operate continuously and accurately.

Software

- User Interface (UI):
  - Platform: Built with WinForms on the Visual Studio platform, the user interface provides the necessary functions to interact with the system. Including capturing facial images, registering new faces, and displaying recognition results.
  - Function: The interface allows users to perform operations such as capturing facial images via ESP32-CAM, entering personal information, and sending registration or recognition requests to the API server.
- API Server:
  - Function: The API server processes requests from the user interface and interacts with the database to perform face registration and recognition. The API also performs encoding and decoding of image data for comparison with the database.

Face Recognition Model:

- Convolutional Neural Network (CNN) Model:
  - Function: The CNN model is used to extract features from facial images and perform recognition. CNN is a type of deep learning neural network designed to process image data, using convolutional layers to detect important features.
  - Structure: The CNN model consists of convolutional layers, pooling layers, flattened layers, and fully connected layers. The model is trained to classify and recognize faces based on the extracted features.

Operation Process

- Facial Registration:
  - Users provide personal information and take a photo of their face through the user interface.

  - The photo and personal information are sent to the Flask server, where they are processed and stored in the database along with the face encoding.
- Facial Recognition:
  - When a user wants to access, they take a photo of their face, which is sent to the Flask server.
  - The Flask server compares the user's face encoding with the data in the database to determine if the face matches any registered users.
- Door Control:
  - If the face is recognized, the Flask server sends a door-opening command to the ESP32-CAM.
  - The ESP32-CAM activates the relay to open the door, allowing the user to enter and exit.

This system not only ensures security by using facial recognition technology, but also provides convenience and flexible integration with existing hardware devices.

2.2 HARDWARE AND SOFTWARE

2.2.1 Hardware

A. ESP32-CAM

The world has been developing IoT technology, that is, the Internet of Things, for quite a long time. From creative enthusiasts to developers, everyone is interested in smart technology, designing various circuits and products to launch on the market.

The chips in the ESP32 series are one of the smart modules commonly used in IoT. ESP32-CAM AI-Thinker is an upgraded version of ESP8266-01 launched by Espressif with many features. The ultra-small, low-power module has two high-performance 32-bit LX6 CPUs with a 7-layer pipeline architecture.

ESP32-CAM has integrated Wi-Fi, Bluetooth and can be used with OV2640 or OV7670 cameras. The ESP32 IC has high-resolution ADC, SPI, I2C and UART protocols for data communication. The module has a Hall sensor, temperature sensor and touch sensor, and a watchdog timer.

The RTC peripheral can operate in different modes. The module has a maximum clock frequency of 160 MHz for a computing capacity of up to 600 DMPIS. Moreover, it is quite durable and highly reliable when connected to the internet.

The ESP32-CAM AI-Thinker module has many applications such as home automation, smart devices, positioning systems, security systems and is suitable for IoT applications.

This board is a 27x40.5x4.5 DIP PCB board. The figure below shows the components of the ESP32-CAM board on both the top and bottom sides.

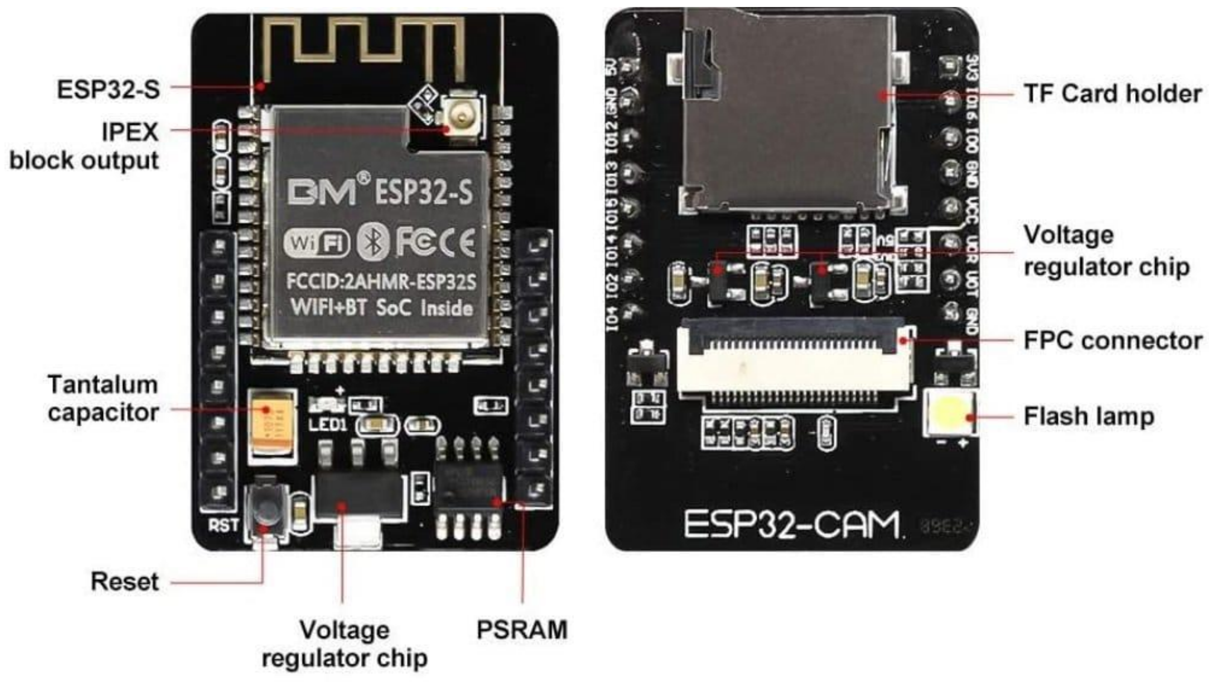


Figure 1: Structure of ESP32-Cam

- ESP32-S Chip: This module is a main chip with two 32-bit high-performance LX6 CPUs with a 7-layer pipeline architecture and is used in all processing.
- IPEX block output: IPEX is connected to the GSM antenna for signal transmission.
- Tantalum capacitors: Tantalum capacitors are mainly used on small-sized modules. They are durable and provide power filtering for stable signal output.
- Reset button: When pressed, the reset button will reset the code executed on the module.
- Voltage regulator chip: Voltage regulator chip on the module to maintain stable output voltage regardless of changes in the input power supply. It regulates the voltage at 3.3V.

- PSRAM: 4MB low-power "pseudo" random access memory is integrated in the module for fast processing speed. It helps the camera run smoothly.
- TF card port: ESP32 series has a micro-SD card port for data storage. All data transmission is through serial peripheral communication.
- FPC connector: For attaching the camera, ESP32 module has a flexible circuit connector. The height of the connector is proportional to the signal reliability.
- Flash: The flash generates electrical pulses that act as a flash for the camera to capture clear images.

The following diagram shows the pinout of the AI-Thinker ESP32-CAM Module:

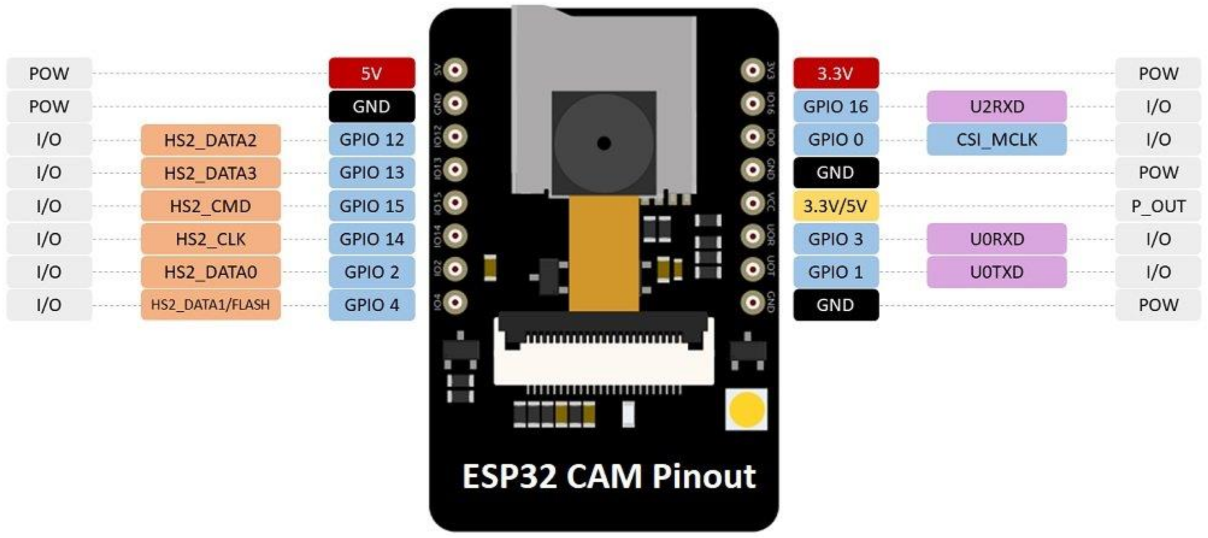


Figure 2: ESP32-Cam Pins

Unlike the ESP32 family kit or other ESP32 boards, the ESP32-CAM AI-Thinker board does not have any built-in programmer to write code to the ESP32-S chip.

Therefore, to upload code, we must use an external USB to UART Converter or FTDI Programmer.

In my system, ESP32-CAM is used to capture the user's face. The image is then sent to the Flask server for processing. In addition, ESP32-CAM is also used to control automatic doors through a relay module.

B. Relay and Electric Lock

Relay is an electromagnetic switch controlled by electrical signal. In this system, relay is used to control electric lock. Once the user's face is authenticated, the relay receives a command from the ESP32-CAM to unlock the door. The relay connects to the electronic lock and provides the door opening signal. Use 5V relay module to control electric lock. Electronic lock is controlled by signal from relay, allowing or prohibiting door opening.

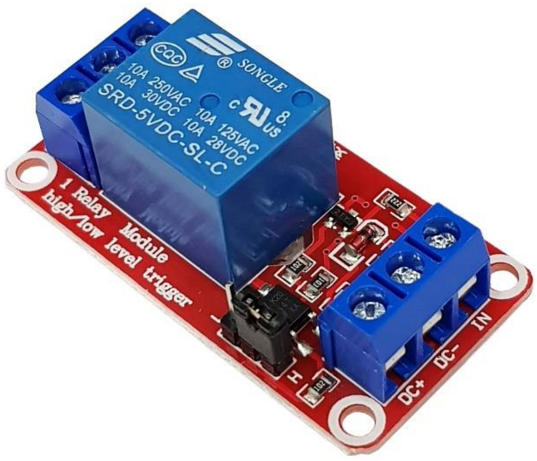


Figure 3: Relay Module 5V

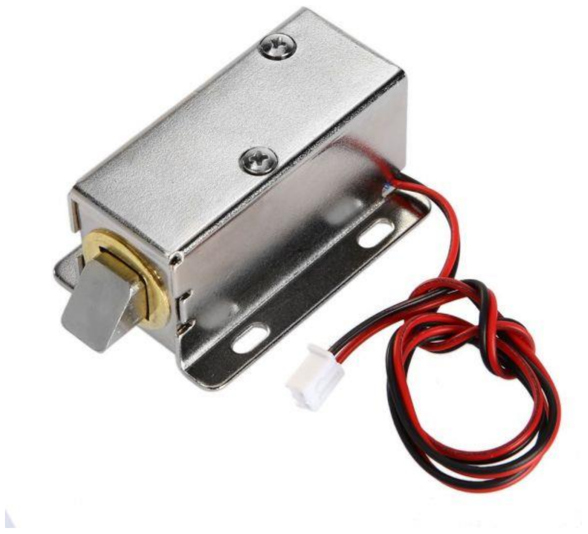


Figure 4: Electric Door Lock

2.2.2. Software

A. Arduino IE (ESP32)

To control the ESP32-CAM and integrate it into the system, we use Arduino IDE, a popular tool for microcontroller programming. ESP32-CAM is a microcontroller module with an integrated camera, allowing to take pictures and transmit image data over Wi-Fi network. Below are the details of the ESP32-CAM programming process using Arduino IDE and the main functions of the source code.

Below is sample source code for using ESP32-CAM for taking photos and controlling doors. This code sets up a web server on ESP32-CAM, allowing requests to be sent to take photos and open doors remotely:

```
#include "esp_camera.h"
#include <WiFi.h>
#include <ESPAsyncWebServer.h>

// Thay đổi các thông tin dưới đây với thông tin mạng của bạn
const char* ssid = "your_SSID";
const char* password = "your_PASSWORD";

// Cấu hình camera
#define CAMERA_MODEL_AI_THINKER
#include "camera_pins.h"

// Khởi tạo máy chủ web
AsyncWebServer server(80);

// Hàm khởi động WiFi
void startWiFi() {
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("Connected.");
  Serial.print("IP Address: ");
  Serial.println(WiFi.localIP());
}

// Xử lý yêu cầu chụp ảnh
void handleCapture(AsyncWebServerRequest *request) {
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    request->send(500, "text/plain", "Failed to capture image");
    return;
  }
  String response = "data:image/jpeg;base64,";
  response += base64::encode(fb->buf, fb->len);
  esp_camera_fb_return(fb);
  request->send(200, "text/plain", response);
}

// Xử lý yêu cầu mở cửa
void handleOpenDoor(AsyncWebServerRequest *request) {
  // Thực hiện các hành động mở cửa ở đây
  request->send(200, "text/plain", "Door opened");
}

void setup() {
  Serial.begin(115200);
  camera_config_t config;
  // Cấu hình camera ở đây (cần phải thay đổi tùy theo module bạn sử dụng)
  // ...

  if (esp_camera_init(&config) != ESP_OK) {
    Serial.println("Camera init failed");
    return;
  }

  startWiFi();

  server.on("/capture", HTTP_GET, handleCapture);
  server.on("/open-door", HTTP_GET, handleOpenDoor);

  server.begin();
}

void loop() {
  // Không cần làm gì ở đây
}
```

Communication: ESP32-CAM communicates via HTTP with other components of the system, such as the Flask server and the user interface.

Function: Provides APIs for taking photos and opening doors, serving the needs of facial recognition and remote door control.

B. Visual Studio (UI)

Visual Studio is a powerful integrated development environment (IDE) used to develop Windows Forms applications for facial recognition systems. With support for C# programming and deep integration with project management tools, Visual Studio provides a stable and efficient platform for building and deploying complex software applications.

Features and Benefits:

- Integrated Development Environment (IDE): Visual Studio provides an integrated development environment with tools and features that support writing, testing, and debugging source code. Notable features include an intelligent source code editor, a built-in debugger, and a drag-and-drop user interface design tool.
- User Interface Design: In this system, Visual Studio is used to design the user interface (UI) for Windows Forms applications. Using Visual Studio Designer, developers can easily create and edit interface elements such as buttons, text boxes, and images. Using a graphical interface increases design efficiency and reduces development time.
- Integration with APIs and Other Components: Visual Studio supports easy integration with APIs and other system components. In this case, the Windows Forms application uses the APIClient class to interact with the Flask API, and the ESP32Controller class to communicate with the ESP32-CAM. Visual Studio provides tools for managing libraries and dependencies, as well as the ability to integrate with other systems through web services.
- Project and Version Management: Visual Studio supports project management through version control tools such as Git, which helps manage source code and coordinate team work effectively. This is important in maintaining and evolving software systems over time.

Configuration and Deployment:

- Project Configuration: In Visual Studio, project configuration includes setting compilation parameters, configuring the runtime environment, and managing dependencies. This ensures that Windows Forms applications operate reliably and efficiently in development and deployment environments.
- Deployment and Testing: Visual Studio supports application testing and deployment processes, including building applications, running automated tests, and creating

installation packages. This ensures that applications meet quality standards and can be deployed easily on target systems.

C. Flask (Python)

The API Server is the component responsible for handling requests from the user interface and interacting with the database. It is implemented using Flask, a lightweight framework for Python, which provides endpoints for face registration and recognition.

Flask is a popular web framework written in the Python programming language. The technology is often used to build websites from simple applications to more complex systems.

Flask is designed to be extensible and functional, and it also provides the tools and libraries needed to develop effective web applications. Flask also has a strong creative community and support from the Python community.

Flask Framework has some important features that developers often use to build web effects. Here are some of the key features of Flask:

- Lightweight and Easy to Use: The technology is lightweight and the source code is easy to read, making it easy for developers to learn and customize according to their specific needs.
- Flexible Routing: Flask provides a routing mechanism that allows developers to define URL patterns and assign them to corresponding handlers. This helps manage and process HTTP requests efficiently.
- Template Engine: Flask integrates Jinja2, which is a powerful template compiler that allows for the creation of user interfaces.
- Extensively Extensible: Despite its stripped-down nature, Flask is still highly extensible through the use of community extensions and libraries. Users can analyze features such as authentication, login, navigation, interactive databases, and more.
- Built-in Development Server: Flask provides a contract development server, making it easy for developers to test and develop applications without additional configuration.
- Send RESTful requests: Flask supports building APIs and RESTful applications in a functional and efficient way.

- Large and active community: Flask has a large user base and strong support from the Python community, making it easy for developers to find information and documentation.

An example of how a Flask application can develop a Hello World program in just a few lines of code is below:

```
from flask import Flask

app = Flask(__name__)

@app.route('/')

def hello_world():

   return 'Hello World!'

if __name__ == '__main__':

   app.run()
```

If you want to develop an application from your local computer, you can also do it easily with Flask. You will save this program as server.py and run it with python server.py

```
$ python server.py

 * Serving Flask app "hello"

 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

The program will then start a web server that is only available on your computer. In your web browser, open localhost on port 5000(urk) and you should see “Hello World” displayed.

System Functions:

- Face Registration API: Flask handles face registration requests from the user interface. When a user sends a request to register a new face, Flask receives a face image along with personal information, encodes the face using a CNN model, and stores the data in a SQLite database.
- Face Recognition API: When a face recognition request comes from the ESP32-CAM, Flask processes the incoming image, encodes the face, and compares this encoding with the database to identify the user.

- Flask is used to create and manage API endpoints like /register and /recognize, which allow communication between the user interface and the database.

Face Registration API Example:

```
from flask import Flask, request, jsonify
app = Flask(__name__)

@app.route('/register', methods=['POST'])
def register():
    # Xử lý yêu cầu đăng ký
    pass

@app.route('/recognize', methods=['POST'])
def recognize():
    # Xử lý yêu cầu nhận diện
    pass

if __name__ == '__main__':
    app.run(debug=True)
```

B. TensorFlow/Keras (Python)

Widely used by data scientists, software developers, and training environments, TensorFlow is an open-source framework for machine learning that uses data flow graphs. Nodes in the graph represent mathematical operations, while edges in the graph represent multidimensional arrays of data (tensors) that flow between them. This flexible architecture allows machine learning algorithms to be described as graphs of connected operations. They can be trained and executed on GPUs, CPUs, and TPUs on a variety of platforms, from mobile devices to desktops to high-end servers, without rewriting code. This means that programmers from all backgrounds can use the same toolset to collaborate, greatly improving efficiency. Originally developed by the Google Brain Team to conduct research on machine learning and deep neural networks, the system is flexible enough to be applied to many other areas.

There are three distinct parts that define the TensorFlow workflow, namely data processing, model building, and model training to make predictions. The framework imports data in the form of a multidimensional array called a tensor and executes in two different ways. The primary approach is to build a computational graph that defines the data flow to train the model. The second, more intuitive approach is to follow imperative programming principles and evaluate operations immediately.

Using the TensorFlow architecture, training is done on the desktop or in the data center. In both cases, the process is accelerated by placing tensors on GPUs. The trained models can then run on multiple platforms, from desktops to mobile devices to the cloud.

TensorFlow also contains many supporting features. For example, TensorBoard , which allows users to visually monitor the training process, basic computation graphs, and metrics for debugging and evaluating model performance. Tensorboard is a unified visualization framework for Tensorflow and Keras.

Keras is a high-level API that runs on TensorFlow. Keras enhances the abstraction of TensorFlow by providing a simplified API for building models for common use cases. The driving idea behind the API is to be able to translate ideas into results in the shortest possible time.

Benefits of TensorFlow:

- TensorFlow can be used to develop models for a variety of tasks, including natural language processing, image recognition, handwriting recognition, and various computational simulations such as partial differential equations.
- The main benefits of TensorFlow are its ability to perform low-level operations on multiple accelerated platforms, automatic gradient computation, production-level scalability, and interactive graph output. By providing Keras as a high-level API and implementing it as an alternative to data flow modeling on TensorFlow, it is always easy to write code comfortably.
- As the original developer of TensorFlow, Google has remained a strong supporter of the library and has fostered its rapid development. For example, Google has created an online hub to share various user-generated models.

Functions in my System:

- Building CNN Model: Convolutional Neural Network (CNN) model is designed to analyze and recognize features in facial images. This model learns to recognize and classify faces from training data.

- Model Training: TensorFlow/Keras is used to train the CNN model using facial image data. The training process involves adjusting the weights of the model to improve prediction accuracy.

D. Database

A database is an organized collection of data, typically stored and accessed electronically from a computer system. As databases become more complex, they are often developed using formal design and modeling techniques.

The role and importance of database:

- First, database helps ensure data security. Ensuring data integrity and security is the most important point in data storage.
- Second, database helps ensure data security. Systematic arrangement structure - this is what makes the biggest difference between normal data and database. Data will be stored in a certain structure, with high consistency. With this feature, database helps users conveniently create, store, search and use data accurately and quickly.
- Third, ensure data retrieval. Many people can use the database at the same time without having to go through complicated steps thanks to accessing from different ways. Therefore, there will be many advantages in using, managing, accessing data, ...
- Fourth, management is easier with database. A database is designed to support the creation, updating and exploitation of information more easily. Data will be updated regularly and completely non-duplicated. Using a database helps create more professional products, systematic storage, and easy management.
- Fifth, the database is flexible to change according to the needs of the user. You can flexibly change the size and complexity of a database. There are databases that only contain a few hundred records (a list of students in a class) and there are databases with very large capacity (such as a database to manage goods in a supermarket system).

E. SQLite

A database management system (DBMS) is software that interacts with end users, primarily using a database to collect and analyze data. DBMS software includes the core utilities provided for database administration. The sum of the database, DBMS, and related applications can be called a “database system”. These model data in the form of rows and columns in a series of tables and mostly use SQL to record and query data.

SQL stands for Structured Query Language. It is a tool designed to manage data used in many fields, allowing you to access and perform operations such as retrieving rows or modifying rows, extracting, creating, deleting data. SQL is also the standard language for RDBMSs.

The object of SQL is data tables and these tables include many columns and rows. Columns are called fields and rows are the records of the table. Columns with specific names and data types make up the structure of the table. When a table is systematically organized for a certain purpose or work, we have a database.

SQL provides us with many benefits:

- First, creating new databases when designing websites or programming software.
- Second, creating new tables and views in the database.
- Third, easily creating, inserting, deleting records in a database.
- Fourth, retrieving data from a database.

SQL functions:

- First, allowing access to databases in many different ways, thanks to the use of commands.
- Second, users can access data from relational databases.
- Third, SQL also allows users to describe data.
- Fourth, allowing users to define data and manipulate it when needed in a database.
- Fifth, can create, delete databases and tables.
- Sixth, allowing users to create views, functions, procedures in a database.

System Functions:

- Face Data Storage: SQLite is used to store registered face information along with the user's personal information. Each face is encoded and stored as a BLOB (Binary Large Object).
- Table Structure: The faces table in the database contains fields for ID, name, date of birth and face encoding.

2.3 FACE RECOGNITION MODEL

2.3.1 Convolution Neural Network

Convolutional Neural Network is one of the main methods when using image data. This network architecture appears because image data processing methods often use the value of each pixel. So with an image with a value of size 100x100 using RGB channel, we have a total of 100 * 100 * 3 equal to 30000 nodes in the input layer. That leads to a large number of weights and biases, leading to the neural network becoming too massive, making it difficult to calculate. Moreover, we can see that the information of pixels is often only affected by the pixels right next to it, so ignoring some nodes in the input layer in each training will not reduce the accuracy of the model. So people use convolutional windows to solve the problem of a large number of parameters while still extracting image features. Technically, in a CNN deep learning model, the input image is passed through a series of convolutional layers with filters, then a Pooling layer, then fully connected layers (FC) and finally a softmax function is applied to classify an object based on a probability value between 0 and 1.

Convolutional Neural Networks are very similar to ordinary Neural Networks: they are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. And they still have a loss function (e.g. SVM/Softmax) on the last (fully-connected) layer and all the tips/tricks we developed for learning regular Neural Networks still apply.

So what changes? CNN architectures make the explicit assumption that the inputs are images, which allows us to encode certain properties into the architecture. These then make the forward function more efficient to implement and vastly reduce the amount of parameters in the network.

Neural Networks receive an input (a single vector), and transform it through a series of hidden layers. Each hidden layer is made up of a set of neurons, where each neuron is fully connected to all neurons in the previous layer, and where neurons in a single layer function completely independently and do not share any connections. The last fully-connected layer is called the “output layer” and in classification settings it represents the class scores.

Regular Neural Nets don’t scale well to full images. In CIFAR-10, images are only of size 32x32x3 (32 wide, 32 high, 3 color channels), so a single fully-connected neuron in a first hidden layer of a regular Neural Network would have 32*32*3 = 3072 weights. This amount still seems manageable, but clearly this fully-connected structure does not scale to larger images. For example, an image of more respectable size, e.g. 200x200x3, would lead to neurons that have 200*200*3 = 120,000 weights. Moreover, we would almost certainly want to have several such neurons, so the parameters would add up quickly! Clearly, this full connectivity is wasteful and the huge number of parameters would quickly lead to overfitting.

3D volumes of neurons. Convolutional Neural Networks take advantage of the fact that the input consists of images and they constrain the architecture in a more sensible way. In particular, unlike a regular Neural Network, the layers of a ConvNet have neurons arranged in 3 dimensions: width, height, depth. (Note that the word depth here refers to the third dimension of an activation volume, not to the depth of a full Neural Network, which can refer to the total number of layers in a network.) For example, the input images in CIFAR-10 are an input volume of activations, and the volume has dimensions 32x32x3 (width, height, depth respectively). As we will soon see, the neurons in a layer will only be connected to a small region of the layer before it, instead of all of the neurons in a fully-connected manner. Moreover, the final output layer would for CIFAR-10 have dimensions 1x1x10, because by the end of the ConvNet architecture we will reduce the full image into a single vector of class scores, arranged along the depth dimension. Here is a visualization:

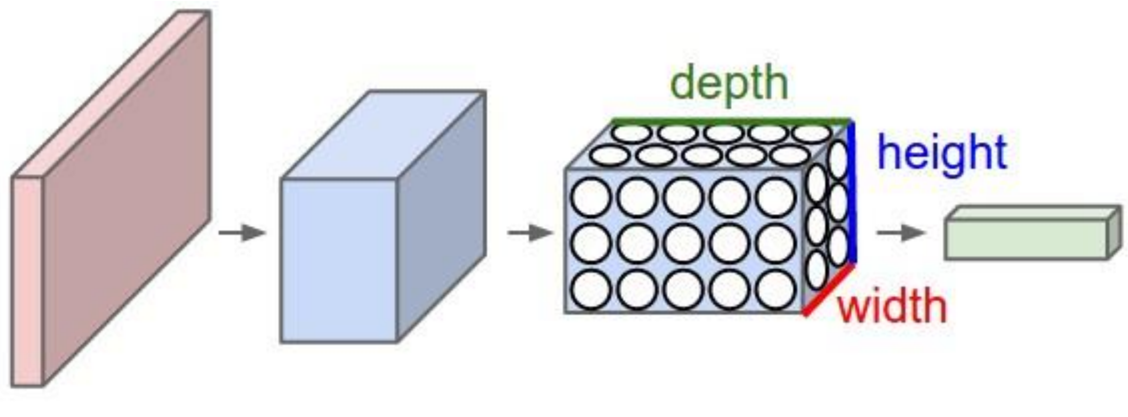


Figure 5: Basic layers of CNN

Layers used to build CNN:

As we described above, a simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. We use three main types of layers to build CNN architectures: Convolutional Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We will stack these layers to form a full CNN architecture.

Example Architecture: Overview. We will go into more details below, but a simple ConvNet for CIFAR-10 classification could have the architecture [INPUT - CONV - RELU - POOL - FC]. In more detail:

- INPUT [32x32x3] will hold the raw pixel values of the image, in this case an image of width 32, height 32, and with three color channels R,G,B.
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.
- RELU layer will apply an elementwise activation function, such as the max(0,x) thresholding at zero. This leaves the size of the volume unchanged ([32x32x12]).
- POOL layer will perform a down sampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].
- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

In this way, CNN transform the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and other don’t. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume, but also of the parameters (the weights and biases of the neurons). On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers will be trained with gradient descent so that the class scores that the CNN computes are consistent with the labels in the training set for each image.

In summary:

- A CNN architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores)
- There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular)
- Each Layer accepts an input 3D volume and transforms it to an output 3D volume through a differentiable function
- Each Layer may or may not have parameters (e.g. CONV/FC do, RELU/POOL don't)
- Each Layer may or may not have additional hyperparameters (e.g. CONV/FC/POOL do, RELU doesn't)

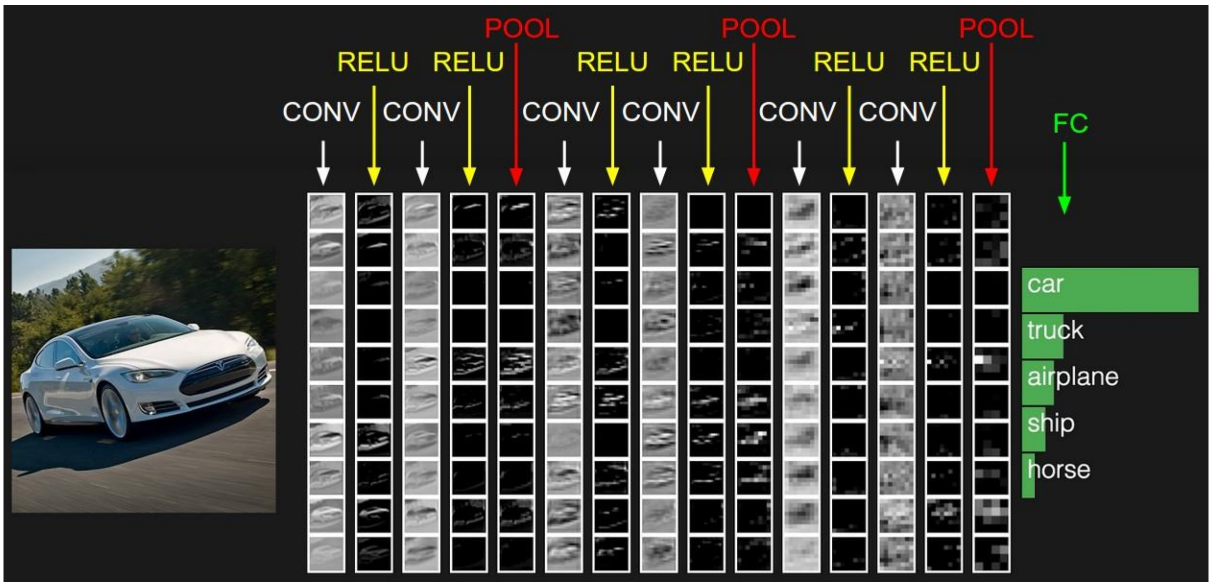


Figure 6: The activations of an example CNN architecture

We now describe the individual layers and the details of them.

- Convolution Layer: The convolution layer is the first layer that extracts feature from the image. This layer parameter consists of a set of learnable filters. The filters are small, usually of the first two dimensions 3x3 or 5x5, ... and have a depth equal to the depth of the input. By gradually sliding the filter horizontally and vertically across the image, they obtain a Feature Map containing the features extracted from the input image. The process of sliding filters often has specified values including:
  - padding: specifies the filter buffer or the gray part added to the image
  - stride: specifies the step in the implementation process.

The following illustration will help us visualize the above process better:

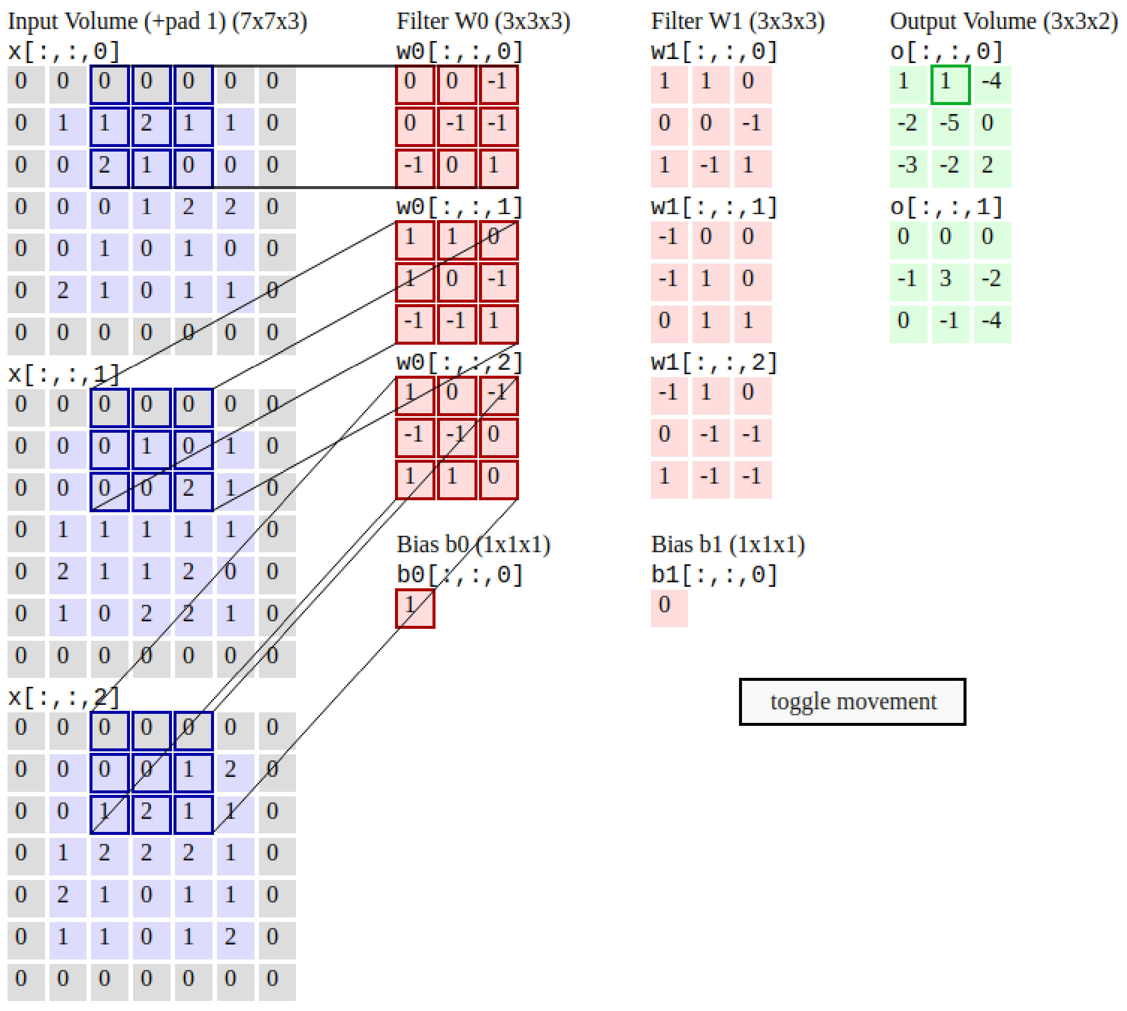


Figure 7: Convolution layer

With each different kernel, we will learn different features of the image, so in each convolutional layer, we will use many kernels to learn many properties of the image. Because each kernel outputs a matrix, k kernels will give k output matrices. We combine these k output matrices into a 3-dimensional tensor with depth k. The output of the convolutional layer will go through the activation function before becoming the input of the next convolutional layer.

- Pooling Layer: Pooling layers are often used between convolutional layers, to reduce the size of the data while preserving important features. The reduced size of the data reduces the computation in the model. In this process, the stride and padding rules are applied as in the image convolution calculation.

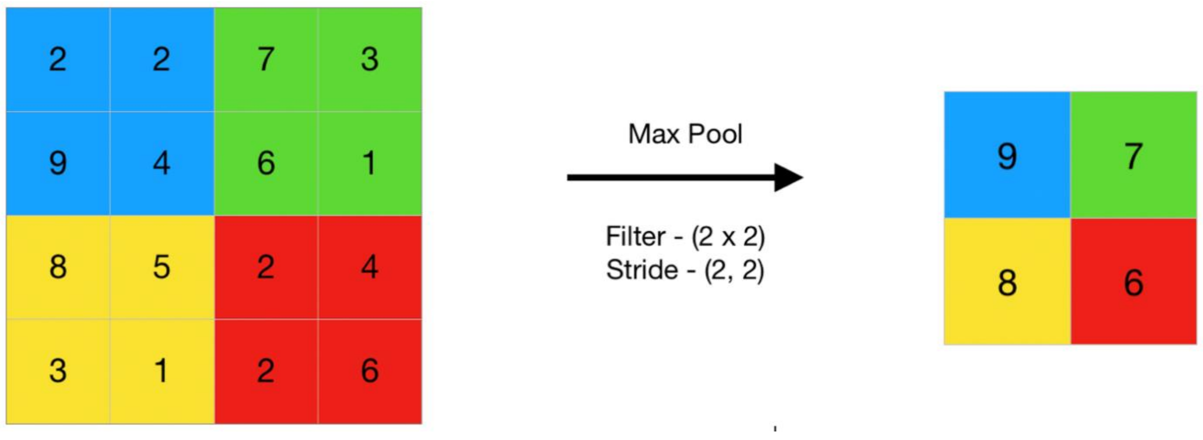


Figure 8: Pooling layer

- Fully-connected Layer: After the image is passed through many convolutional layers and pooling layers, the model has learned the relative features of the image, the tensor of the output of the last layer will be flattened into a vector and fed into a connected layer like a neural network. With the FC layer combined with the features together to create a model. Finally use softmax or sigmoid to classify the output.

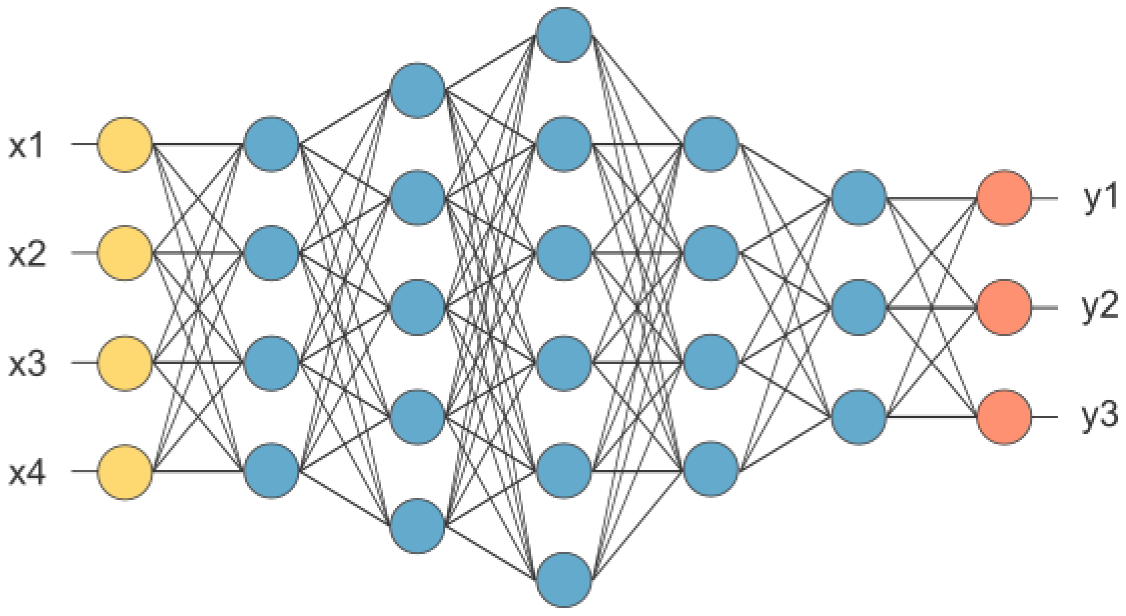


Figure 9: Fully-connected Layer

2.3.2 CNN Model Training

Data Preparation:

- Dataset: Image data is prepared in the form of face images, with each image assigned a corresponding label. The data is usually divided into two sets: training set and validation set.
- Preprocessing: Images are normalized to a uniform size (e.g. 64x64 pixels), and pixel values are divided by 255 to bring them into the range [0,1].
- Code:

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

datagen = ImageDataGenerator(rescale=1./255)
train_data = datagen.flow_from_directory('train_data', target_size=(64, 64), batch_size=32,
class_mode='binary')
val_data = datagen.flow_from_directory('val_data', target_size=(64, 64), batch_size=32,
class_mode='binary')
```

Model Building:

- Model Initialization: Create a CNN model with convolutional and pooling layers for feature extraction, and fully connected layers for classification.
- Code:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(1, activation='sigmoid')  # Hoặc sử dụng softmax cho phân loại đa lớp
])
```

Model Training:

- Model Compilation: Use loss function and optimizer to compile the model. For example, use binary crossentropy and Adam optimization.
- Training: Train the model on the training set and validate its performance on the validation set.

- Code:

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
history = model.fit(train_data, epochs=10, validation_data=val_data)
model.save('face_recognition_model.h5')
```

Model Evaluation:

- Performance Evaluation: After training, the model is evaluated on the test dataset to measure its accuracy and generalization ability.
- Code:

```
test_loss, test_accuracy = model.evaluate(test_data)
print(f'Test Accuracy: {test_accuracy}')
```

2.3.3 CNN Model Deployment

Save Model:

- Storage: After training, the model is saved to an HDF5 (.h5) file for use in real-world applications.
- Code:

```
model.save('face_recognition_model.h5')
```

Load Model:

- Reload: When deployed, the model can be reloaded from the saved file to perform predictions.
- Code:

```
from tensorflow.keras.models import load_model
model = load_model('face_recognition_model.h5')
```

Complete program for face recognition model:

```
import numpy as np
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img,
img_to_array
from tensorflow.keras.models import Sequential, load_model
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
import os

# 1. Chuẩn Bị Dữ Liệu
def prepare_data(train_dir, val_dir, test_dir):
    datagen = ImageDataGenerator(rescale=1./255)

    train_data = datagen.flow_from_directory(
        train_dir,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary'
    )

    val_data = datagen.flow_from_directory(
        val_dir,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary'
    )

    test_data = datagen.flow_from_directory(
        test_dir,
        target_size=(64, 64),
        batch_size=32,
        class_mode='binary'
    )

    return train_data, val_data, test_data

# 2. Xây Dựng Mô Hình CNN
def build_model():
    model = Sequential([
        Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64, 3)),
        MaxPooling2D((2, 2)),
        Conv2D(64, (3, 3), activation='relu'),
        MaxPooling2D((2, 2)),
        Flatten(),
        Dense(128, activation='relu'),
        Dense(1, activation='sigmoid')
    ])

    model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
    return model

# 3. Huấn Luyện Mô Hình
def train_model(model, train_data, val_data, epochs=10):
    history = model.fit(
        train_data,
        epochs=epochs,
        validation_data=val_data
    )
    return history

# 4. Lưu Mô Hình
```

```
def save_model(model, model_path):
    model.save(model_path)

# 5. Tải Mô Hình và Dự Đoán
def load_and_predict(model_path, image_path):
    model = load_model(model_path)

    img = load_img(image_path, target_size=(64, 64))
    img_array = img_to_array(img) / 255.0
    img_array = np.expand_dims(img_array, axis=0)

    prediction = model.predict(img_array)
    return prediction

# 6. Đánh Giá Mô Hình
def evaluate_model(model, test_data):
    test_loss, test_accuracy = model.evaluate(test_data)
    print(f'Test Accuracy: {test_accuracy}')
    return test_loss, test_accuracy

# Main Function
if __name__ == '__main__':
    # Đường dẫn tới thư mục dữ liệu
    train_dir = 'train_data'
    val_dir = 'val_data'
    test_dir = 'test_data'
    model_path = 'face_recognition_model.h5'

    # Chuẩn bị dữ liệu
    train_data, val_data, test_data = prepare_data(train_dir, val_dir, test_dir)

    # Xây dựng mô hình
    model = build_model()

    # Huấn luyện mô hình
    history = train_model(model, train_data, val_data, epochs=10)

    # Lưu mô hình
    save_model(model, model_path)

    # Đánh giá mô hình
    evaluate_model(model, test_data)

    # Dự đoán trên một hình ảnh mới
    image_path = 'path_to_image.jpg'  # Thay thế bằng đường dẫn hình ảnh thực tế
    prediction = load_and_predict(model_path, image_path)
    print(f'Prediction: {prediction}')
```

2.4 FACE REGISTRATION PROCESS

The face registration process is an essential step in a facial recognition system, which aims to collect, process, and store the user's facial information. This process involves not only capturing images, but also preprocessing the data, encoding the face, and storing the information in a database. Below is a detailed and academic description of the face registration process.

2.4.1 Collecting Face Images

Objective: Collect user images to create a facial database for the recognition system.

Process Details:

- Activate the Capture Function: The user interface (UI) provides a function button that allows the user to capture a facial image. When the user presses the button, a capture request is sent to the ESP32-CAM.
- Capture and Send Image: The ESP32-CAM captures a facial image and sends the image as a binary data (byte array) to the server via HTTP.

Example code (C# - WinForms)

```
private async void BtnCapture_Click(object sender, EventArgs e)
{
    byte[] image = await Esp32Controller.CaptureImageAsync();
    if (image == null || image.Length == 0)
    {
        MessageBox.Show("Failed to capture image.");
        return;
    }
    using (MemoryStream ms = new MemoryStream(image))
    {
        pictureBox.Image = Image.FromStream(ms);
    }
}
```

2.4.2 Image Preprocessing

Goal: Convert the captured image into a format suitable for encoding and storage.

Process Details:

- Read Image Data: The API server receives image data from the user interface and opens the image from binary format.
- Size Conversion: The image is resized to a standard size (e.g. 64x64 pixels) to fit the input of the CNN model.
- Normalization: Convert pixel values from the range [0, 255] to [0, 1] to normalize the data, helping the deep learning model work more effectively.

Example code (Python – Flask)

```
from PIL import Image
import numpy as np
import io
def preprocess_image(file):
    image = Image.open(io.BytesIO(file.read()))
    image = image.resize((64, 64))
    image = np.array(image) / 255.0
    return np.expand_dims(image, axis=0)
```

2.4.3 Face Encoding

Objective: Convert face image into feature encoding for comparison and storage.

Process Details:

- Face Prediction: Use trained CNN model to predict face encoding from preprocessed image.
- Encoding Conversion: Face encoding is converted into binary format (byte array) for storage in database.

Example code: (Python – TensorFlow)

```
def encode_face(image):
    model = tf.keras.models.load_model('face_recognition_model.h5')
    face_encoding = model.predict(image)[0]
    return face_encoding
```

2.4.4 Information Storage

Objective: Store the user's face encryption and personal information in the database for future identification.

Process Details:

- Database Connection: Open a connection to the SQLite database to perform the storage operation.
- Perform Storage: Execute an SQL query to insert the user information (name, date of birth, ID) along with the face encryption into the faces table.

Example code: (Python – SQLite3)

```
import sqlite3

def save_face_data(name, dob, face_encoding):
    conn = sqlite3.connect('faces.db')
    cursor = conn.cursor()
    cursor.execute('INSERT INTO faces (name, dob, face_encoding) VALUES (?, ?, ?)',
                   (name, dob, face_encoding.tobytes()))
    conn.commit()
    conn.close()
```

Completed program for face register (include create Flask Server):

```
from flask import Flask, request, jsonify
import numpy as np
import tensorflow as tf
from PIL import Image
import io
import sqlite3

app = Flask(__name__)

# Load mô hình CNN đã huấn luyện
model = tf.keras.models.load_model('face_recognition_model.h5')

# Kết nối cơ sở dữ liệu
def connect_db():
    conn = sqlite3.connect('faces.db')
    return conn

# Tạo bảng cơ sở dữ liệu nếu chưa tồn tại
def create_table():
    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute('''
        CREATE TABLE IF NOT EXISTS faces (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            name TEXT,
            dob TEXT,
            face_encoding BLOB
        )
    ''')
    conn.commit()
    conn.close()

create_table()

# Tiền xử lý hình ảnh
def preprocess_image(file):
    image = Image.open(io.BytesIO(file.read()))
    image = image.resize((64, 64))  # Thay đổi kích thước hình ảnh
    image = np.array(image) / 255.0  # Chuẩn hóa hình ảnh
    return np.expand_dims(image, axis=0)  # Thêm chiều batch

# Mã hóa khuôn mặt
def encode_face(image):
    face_encoding = model.predict(image)[0]
    return face_encoding

# API đăng ký khuôn mặt
@app.route('/register', methods=['POST'])
def register():
    data = request.form
    name = data['name']
    dob = data['dob']
    file = request.files['image']

    image = preprocess_image(file)
    face_encoding = encode_face(image)

    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute('INSERT INTO faces (name, dob, face_encoding) VALUES (?, ?, ?)',
                   (name, dob, face_encoding.tobytes()))
    conn.commit()
    conn.close()

    return jsonify({'message': 'Registration successful'})

# API nhận diện khuôn mặt
@app.route('/recognize', methods=['POST'])
def recognize():
    file = request.files['image']
    image = preprocess_image(file)
    face_encoding = encode_face(image)

    conn = connect_db()
    cursor = conn.cursor()
    cursor.execute('SELECT id, name, dob, face_encoding FROM faces')
    rows = cursor.fetchall()
    conn.close()

    for row in rows:
        db_face_encoding = np.frombuffer(row[3], dtype=np.float32)
        similarity = np.dot(face_encoding, db_face_encoding)
        if similarity > 0.9:  # Ngưỡng tương đồng
            return jsonify({'message': 'Access Granted', 'name': row[1], 'dob': row[2], 'id':
row[0]})

    return jsonify({'message': 'Access Denied'})

if __name__ == '__main__':
    app.run(debug=True)
```

B. API Client

APIClient is the class that handles communication between the Windows Forms application and the Flask API. It makes HTTP requests to send and receive data from the server.

Example code:

```
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace Face
{
    public static class ApiClient
    {
        private static readonly HttpClient client = new HttpClient { BaseAddress = new
Uri("http://localhost:5000/") };

        public static async Task<string> SendImageAsync(byte[] imageBytes)
        {
            using (var content = new ByteArrayContent(imageBytes))
            {
                content.Headers.ContentType = new
System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");
                HttpResponseMessage response = await client.PostAsync("process-image",
content);
                response.EnsureSuccessStatusCode();
                return await response.Content.ReadAsStringAsync();
            }
        }

        public static async Task<string> RegisterFaceAsync(byte[] imageBytes, string name,
string dob, string id)
        {
            using (var form = new MultipartFormDataContent())
            {
                form.Add(new ByteArrayContent(imageBytes), "image", "face.jpg");
                form.Add(new StringContent(name), "name");
                form.Add(new StringContent(dob), "dob");
                form.Add(new StringContent(id), "id");

                HttpResponseMessage response = await client.PostAsync("register-face", form);
                response.EnsureSuccessStatusCode();
                return await response.Content.ReadAsStringAsync();
            }
        }
    }
}
```

SendImageAsync(byte[] imageBytes): Sends an image to the API for face recognition. The API returns the recognition result.

RegisterFaceAsync(byte[] imageBytes, string name, string dob, string id): Sends the user's image and personal information to the API for registering a new face.

2.5 FACE REGISTRATION PROCESS

2.5.1 User Interface Overview

The user interface (UI) in the facial recognition automatic door system plays an important role in providing an easy and efficient way for users to interact with the system. The UI software developed using Visual Studio with Windows Forms provides key functions such as face registration, image capture, and recognition result display.

2.5.2 Structure and Components

Main Form (MainForm):

- PictureBox: Displays the face image captured by ESP32-CAM. PictureBox provides a visual view of the current face image and is an important point for users to verify the image.
  - Buttons (BtnCapture, BtnRegister):
    - BtnCapture: When the user presses this button, the system sends a command to ESP32-CAM to capture and recognize the face. After

capturing, the image will be displayed in the PictureBox and sent to the API for analysis.

- BtnRegister: Allows users to register a new face to the system. This button will send the user's image, name, date of birth, and ID to the API for storage in the database.

- TextBoxes (txtName, txtDOB, txtID): Enter the user's personal information when registering a face. This is where the user enters his/her name, date of birth, and ID to be saved in the database along with the captured face.

UI Functions:

- Capture and Display: When the “Capture” button is pressed, the interface will send a request to ESP32-CAM to capture a photo and then display the photo in the PictureBox. This is an important step for users to view and confirm the image before sending it for recognition or registration.
- Face Registration: After the photo is captured and displayed, users can enter personal information and press the “Register” button to send the information and image to the API for storage and registration in the database.
- Result Display: After sending a recognition or registration request, the system will receive a response from the API and display a message to the user via MessageBox. The message can be a success message, an error message, or a recognition result.

2.5.3 Interaction with Other Parts of the System

Connecting to the API: The user interface communicates with the API (written in Flask) to send and receive image data and personal information. The UI software will use the ApiClient class to make HTTP requests to the server, send image data, and receive responses.

Communicating with the ESP32-CAM: The UI software sends commands to the ESP32-CAM via HTTP protocol to perform functions such as taking a photo or opening a door. These commands are sent through the ESP32Controller class.

2.5.4 User Design and Experience

User-Friendly Design: The interface is designed to be easy to use with clear and easy-to-understand layout elements. Buttons and input fields are clearly labeled so users know what each element does.

User Feedback: The interface provides immediate feedback after users perform actions such as taking a photo or registering, letting users know the system is processing their request and displaying results or error messages.

MainForm.cs:

```
using System;
using System.Drawing;
using System.IO;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Face
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private async void BtnCapture_Click(object sender, EventArgs e)
        {
            byte[] image = await Esp32Controller.CaptureImageAsync();

            if (image == null || image.Length == 0)
            {
                MessageBox.Show("Failed to capture image.");
                return;
            }

            // Hiển thị hình ảnh trong PictureBox
            using (MemoryStream ms = new MemoryStream(image))
            {
                pictureBox.Image = Image.FromStream(ms);
            }

            string result = await ApiClient.SendImageAsync(image);
            MessageBox.Show($"API response: {result}");

            if (result.Contains("unlocked"))
            {
                await Esp32Controller.OpenDoorAsync();
                MessageBox.Show("Door opened!");
            }
            else
            {
                MessageBox.Show("Face not recognized.");
            }
        }

        private async void BtnRegister_Click(object sender, EventArgs e)
        {
            byte[] image = await Esp32Controller.CaptureImageAsync();

            if (image == null || image.Length == 0)
            {
                MessageBox.Show("Failed to capture image.");
                return;
            }

            // Hiển thị hình ảnh trong PictureBox
            using (MemoryStream ms = new MemoryStream(image))
            {
                pictureBox.Image = Image.FromStream(ms);
            }

            string name = txtName.Text;
            string dob = txtDOB.Text;
            string id = txtID.Text;

            string result = await ApiClient.RegisterFaceAsync(image, name, dob, id);
            MessageBox.Show(result);
        }

        private void TxtBox_Enter(object sender, EventArgs e)
        {
            var textBox = sender as TextBox;
            if (textBox != null)
            {
                if (textBox.Text == GetPlaceholderText(textBox))
                {
                    textBox.Text = "";
                    textBox.ForeColor = System.Drawing.Color.Black;
                }
            }
        }

        private void TxtBox_Leave(object sender, EventArgs e)
        {
            var textBox = sender as TextBox;
            if (textBox != null)
            {
                if (string.IsNullOrEmpty(textBox.Text))
                {
                    textBox.Text = GetPlaceholderText(textBox);
                    textBox.ForeColor = System.Drawing.Color.Gray;
                }
            }
        }

        private string GetPlaceholderText(TextBox textBox)
        {
            if (textBox == txtName)
                return "Nhập tên";
            if (textBox == txtDOB)
                return "Nhập DOB";
            if (textBox == txtID)
                return "Nhập ID";
            return "";
        }

        [STAThread]
        public static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new MainForm());
        }
    }
}
```

APIClient.cs:

```
using System;
using System.Net.Http;
using System.Text;
using System.Threading.Tasks;

namespace Face
{
    public static class ApiClient
    {
        private static readonly HttpClient client = new HttpClient { BaseAddress = new
Uri("http://localhost:5000/") };

        public static async Task<string> SendImageAsync(byte[] imageBytes)
        {
            using (var content = new ByteArrayContent(imageBytes))
            {
                content.Headers.ContentType = new
System.Net.Http.Headers.MediaTypeHeaderValue("application/octet-stream");
                HttpResponseMessage response = await client.PostAsync("process-image",
content);
                response.EnsureSuccessStatusCode();
                return await response.Content.ReadAsStringAsync();
            }
        }
        public static async Task<string> RegisterFaceAsync(byte[] imageBytes, string name,
string dob, string id)
        {
            using (var form = new MultipartFormDataContent())
            {
                form.Add(new ByteArrayContent(imageBytes), "image", "face.jpg");
                form.Add(new StringContent(name), "name");
                form.Add(new StringContent(dob), "dob");
                form.Add(new StringContent(id), "id");

                HttpResponseMessage response = await client.PostAsync("register-face", form);
                response.EnsureSuccessStatusCode();
                return await response.Content.ReadAsStringAsync();
            }
        }
    }
}
```

Esp32Controller.cs:

```
using System.Net.Http;
using System.Threading.Tasks;

namespace Face
{
    public static class Esp32Controller
    {
        private static readonly HttpClient client = new HttpClient { BaseAddress = new
Uri("http://esp32-cam-url/") };

        public static async Task<byte[]> CaptureImageAsync()
        {
            HttpResponseMessage response = await client.GetAsync("capture");
            response.EnsureSuccessStatusCode();
            return await response.Content.ReadAsByteArrayAsync();
        }

        public static async Task OpenDoorAsync()
        {
            HttpResponseMessage response = await client.GetAsync("open-door");
            response.EnsureSuccessStatusCode();
        }
    }
}
```

2.6 ALGORITHM FLOWCHARTS

System Overview Diagram:

The system overview diagram describes the main steps from system startup to user request processing, including interactions between the main system components such as ESP32-CAM, Flask server, Windows Forms application and database.

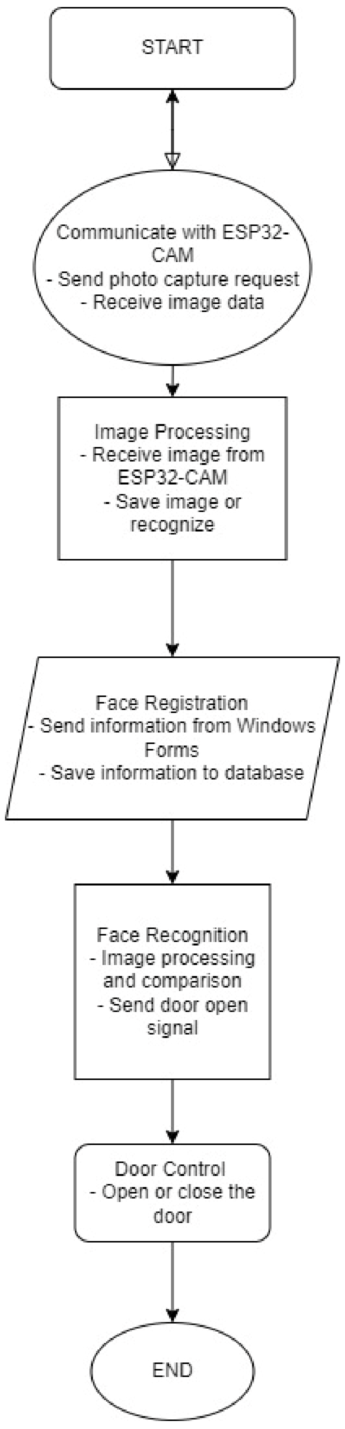

START
Communicate with ESP32-CAM
- Send photo capture request
- Receive image data
Image Processing
- Receive image from ESP32-CAM
- Save image or recognize
Face Registration
- Send information from Windows Forms
- Save information to database
Face Recognition
- Image processing and comparison
- Send door open signal
Door Control
- Open or close the door
END

Figure 10: Overview System Diagram

Detailed Sub-Diagram of Each Process:

- Face Registration Process Sub-diagram:

Description: The face registration process involves the user taking a photo of their face via the Windows Forms application, submitting their personal information, and saving the face information to the database.

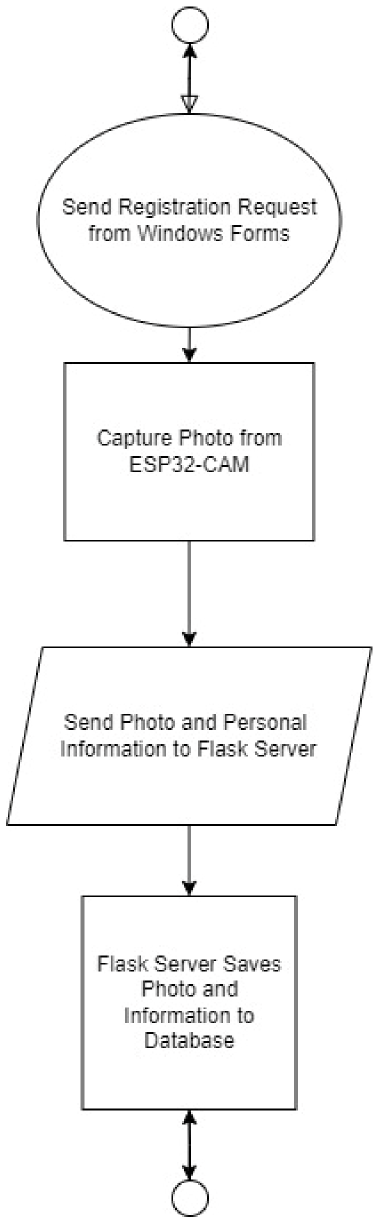

Send Registration Request from Windows Forms
Capture Photo from ESP32-CAM
Send Photo and Personal Information to Flask Server
Flask Server Saves Photo and Information to Database

Figure 11: Face Registration Process Sub-diagram

- Face Recognition Process Sub-diagram:

Description: The face recognition process involves the ESP32-CAM taking a photo and sending a recognition request to the Flask server. The Flask server performs the recognition and returns the result to the ESP32-CAM.

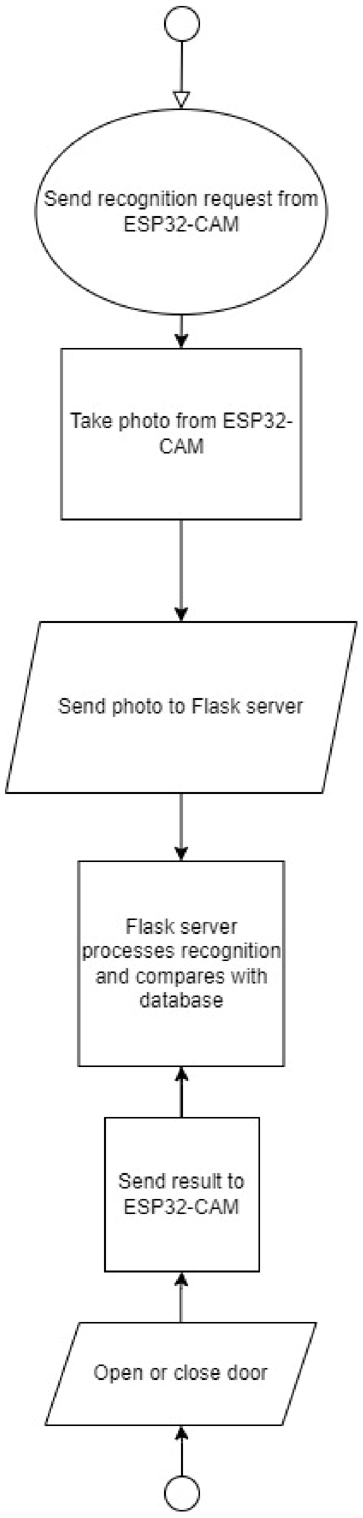

Send recognition request from ESP32-CAM
Take photo from ESP32-CAM
Send photo to Flask server
Flask server processes recognition and compares with database
Send result to ESP32-CAM
Open or close door

Figure 12: Face Recognition Process Sub-diagram

- Door Control Process sub-diagram:

Description: The door control process consists of the Flask server sending a door open signal based on the recognition result to the ESP32-CAM, and the ESP32-CAM controls the electronic lock.

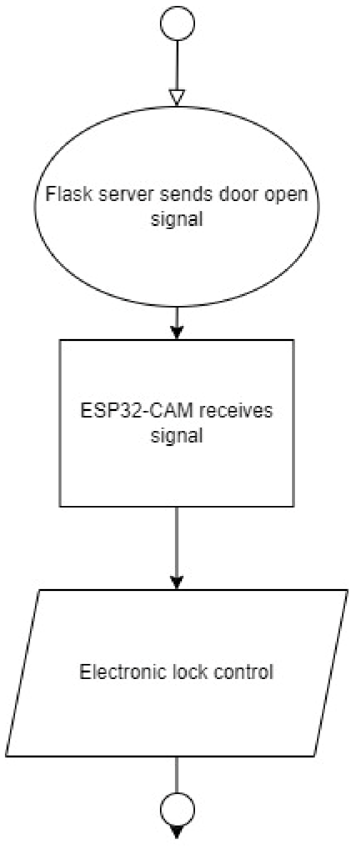
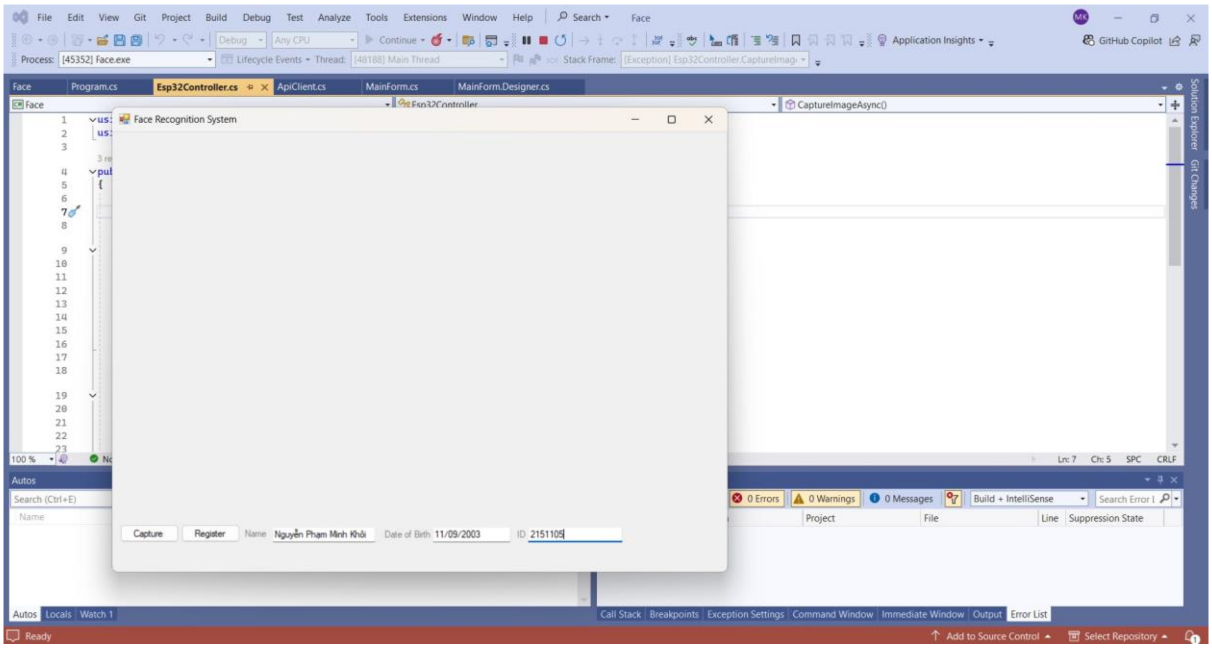

Flask server sends door open signal
ESP32-CAM receives signal
Electronic lock control

Figure 13: Door Control Process sub-diagram

2.7 OVERALL RESULT



Name Nguyễn Phạm Minh Khôi Date of Birth 11/09/2003 ID 2151105

Figure 14: WinForm UI before capture

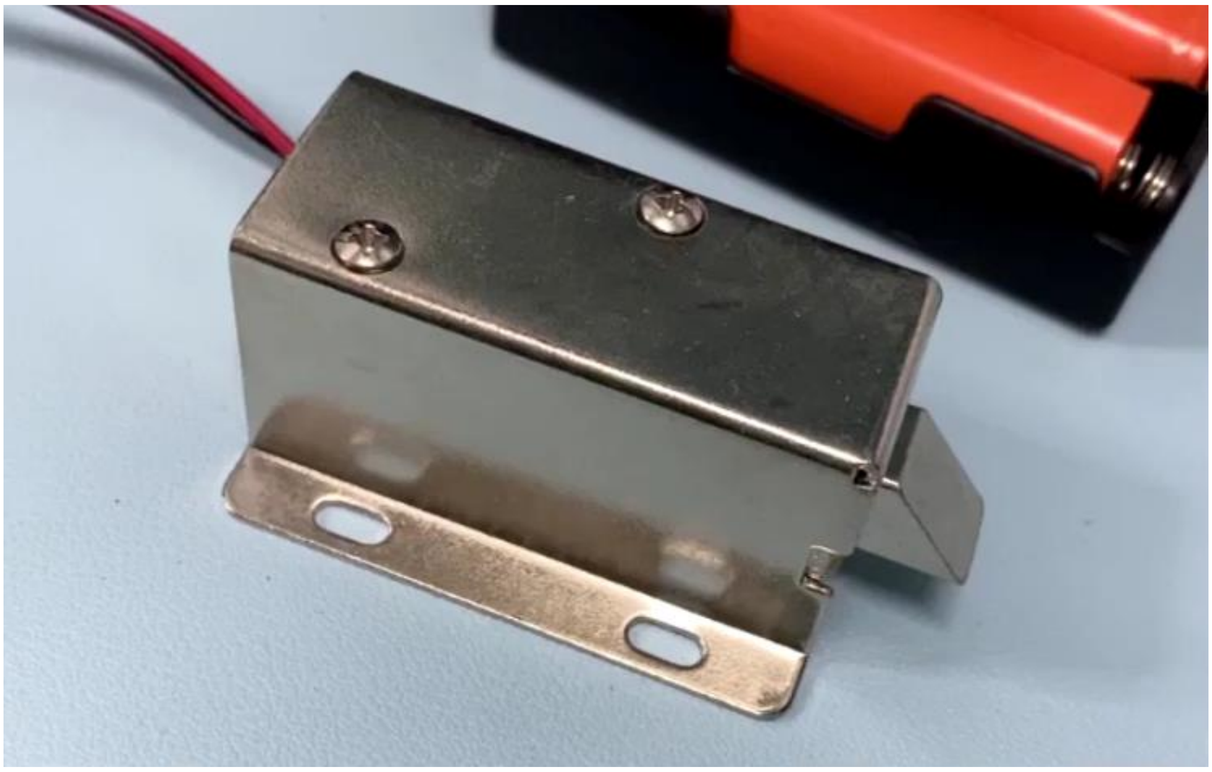
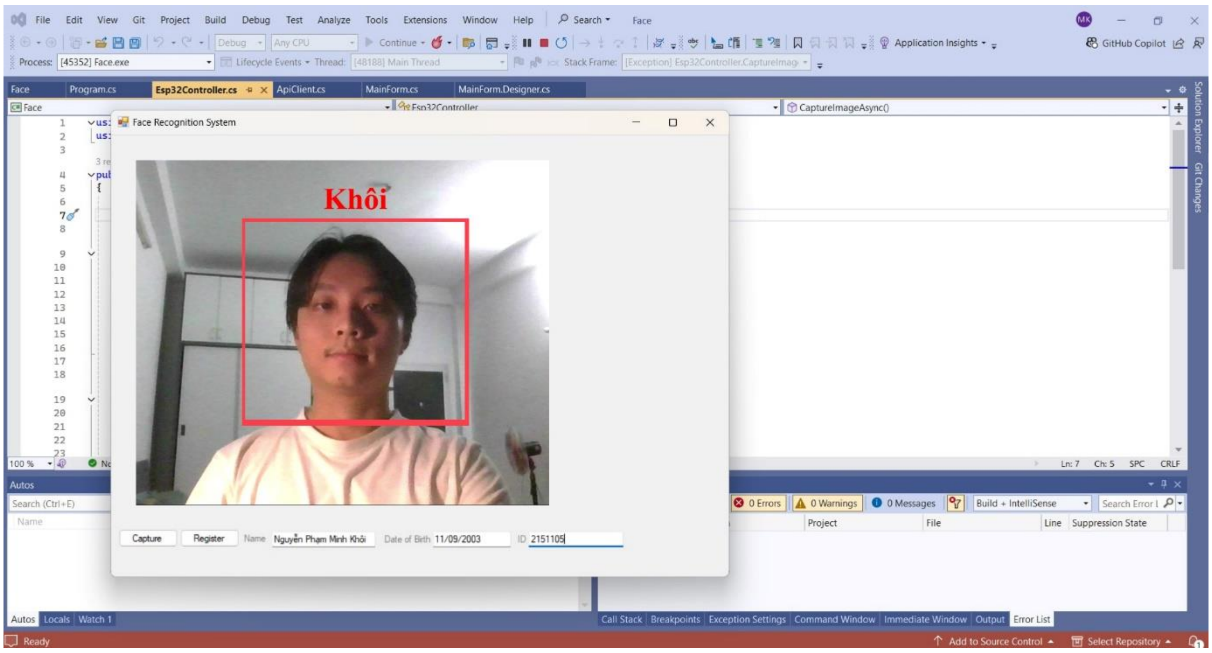


Figure 15: Electric Lock Before Recognition



Name: Nguyễn Phạm Minh Khôi
Date of Birth: 11/09/2003
ID: 2151105

Figure 16: WinForm UI After Captured

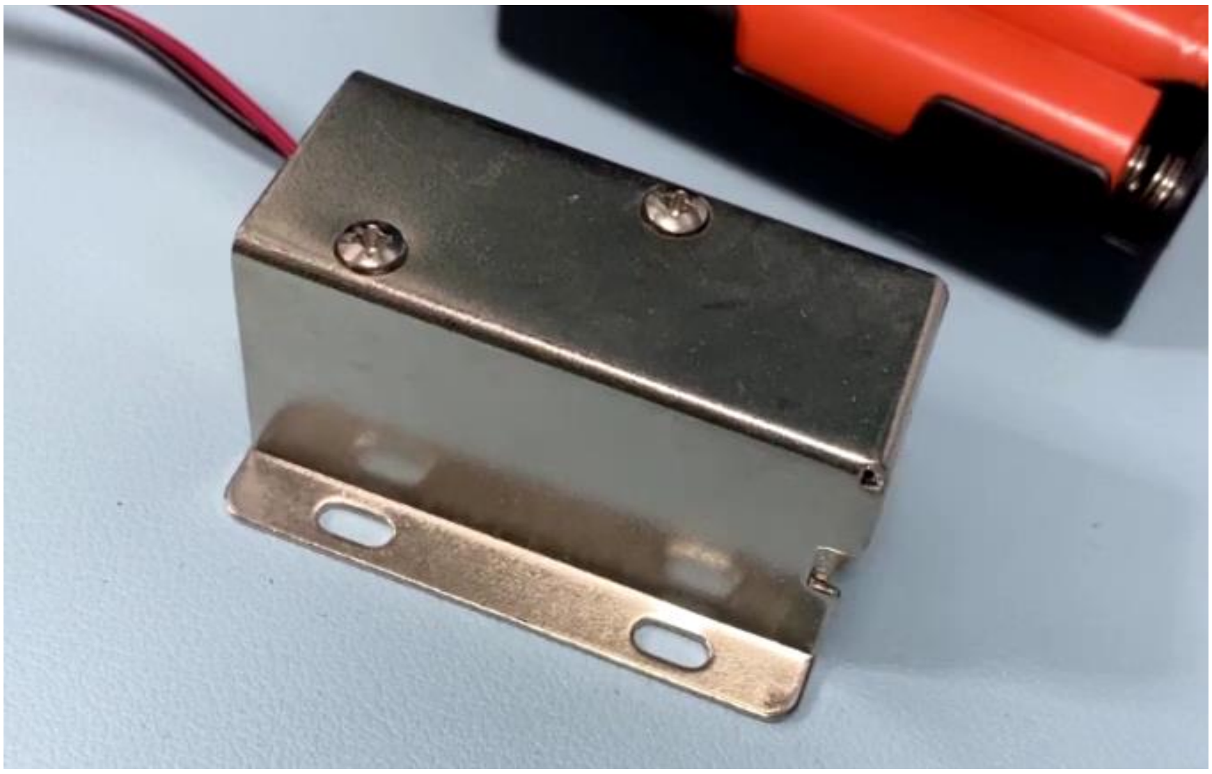


Figure 17: Electric Lock After Recognition

3. SUMMARY

3.1 Internship results

After a period of research and reference from many different sources, with the help of Mr. Vinh and Mr. Quoc, I have basically completed the program. My program has the following advantages:

- First, facial recognition is relatively accurate for many types of faces.
- Second, the door opens almost immediately, without too much delay.
- Third, facial information is automatically stored in the database, convenient for data management.

However, due to my limited knowledge and experience, I realized that my program still has some limitations such as:

- First, the accuracy of the system is still low with lighting conditions or faces with high similarity.
- Second, the program still has many errors when it sometimes fails to recognize faces or recognizes incorrect information.
- Third, the program's database is not really detailed, only storing the most basic information about names and dates of birth.

3.2 Experience gained after internship

During my internship at the company, I learned a lot of experience in how to do the job, clearly understood the future work I could do. I knew what issues and fields my future work would involve. I knew the software and systems related to the industry, and had direct contact with those software. At the same time, I also learned many lessons about skills and work attitudes from my seniors in the company.

Thanks to the experiences learned from my internship at the company, I recognized the shortcomings of the program and proposed some solutions and development directions to overcome them such as:

- Multi-face detection: Improved system to detect multiple faces in the same photo and handle situations where multiple people are in front of the camera.

- Spoof Detection: Adds fake face detection (like fake photos or videos) to improve security.

4. REFERENCES

[1] Trần Đức Trung, "Tìm hiểu về Convolutional Neural Network và làm một ví dụ nhỏ về phân loại ảnh", https://viblo.asia/p/tim-hieu-ve-convolutional-neural-network-va-lam-mot-vi-du-nho-ve-phan-loai-anh-aWj53WXo56m.

[2] Ashish Choudhary, "ESP32-CAM Face Recognition Door Lock System", https://circuitdigest.com/microcontroller-projects/esp32-cam-face-recognition-door-lock-system

[3] CS231n, "CS231n Convolutional Neural Networks for Visual Recognition", https://cs231n.github.io/convolutional-networks/.

[4] Hưng Nguyễn, "Thuật toán CNN là gì? Tìm hiểu về Convolutional Neural Network", https://vietnix.vn/cnn-la-gi/

[5] Sergio Andrés Gutiérrez Rojas, "Multiple Face Detection and Recognition in Real Time", https://www.codeproject.com/Articles/239849/Multiple-Face-Detection-and-Recognition-in-Real-2

5. APPENDIX