TRƯỜNG ĐẠI HỌC SÀI GÒN



KHOA CÔNG NGHỆ THÔNG TIN

**----------****----------**

BÁO CÁO ĐỒ ÁN CUỐI KỲ

HỌC PHẦN: NGÔN NGỮ LẬP TRÌNH PYTHON

ĐỀ TÀI:

**XÂY DỰNG ỨNG DỤNG NHẬN DIỆN BIỂN BÁO GIAO THÔNG**

|  |  |  |
| --- | --- | --- |
| **MÃ HỌC PHẦN:** | **841422** | |
| **NHÓM MÔN HỌC:** | **03** | |
| **HỌC KỲ:** | **02** | |
| **NHÓM BÁO CÁO:** | **05** | |
| **GVHD**: | **ThS.NCS HÀ THANH DŨNG** | |
| **THÀNH VIÊN**: | **3122410149** | **Nguyễn Quốc Huy**  **(Nhóm trưởng)** |
|  | **3122410154** | **Trương Gia Huy** |
|  | **3122410090** | **Nguyễn Văn Vũ Em** |
|  | **3122410098** | **Nguyễn Vũ Hào** |
|  |  |  |

**TP HỒ CHÍ MINH, THÁNG 5 NĂM 2023**

**TRƯỜNG ĐẠI HỌC SÀI GÒN**



KHOA CÔNG NGHỆ THÔNG TIN

**----------****----------**

BÁO CÁO ĐỒ ÁN CUỐI KỲ

HỌC PHẦN: NGÔN NGỮ LẬP TRÌNH PYTHON

ĐỀ TÀI:

**XÂY DỰNG ỨNG DỤNG NHẬN DIỆN BIỂN BÁO GIAO THÔNG**

|  |  |  |
| --- | --- | --- |
| **MÃ HỌC PHẦN:** | **841422** | |
| **NHÓM MÔN HỌC:** | **03** | |
| **HỌC KỲ:** | **02** | |
| **NHÓM BÁO CÁO:** | **05** | |
| **GVHD**: | **ThS.NCS HÀ THANH DŨNG** | |
| **THÀNH VIÊN**: | **3122410149** | **Nguyễn Quốc Huy**  **(Nhóm trưởng)** |
|  | **3122410154** | **Trương Gia Huy** |
|  | **3122410090** | **Nguyễn Văn Vũ Em** |
|  | **3122410098** | **Nguyễn Vũ Hào** |

**TP HỒ CHÍ MINH, THÁNG 5 NĂM 2023**

**PHÂN CÔNG VÀ ĐÁNH GIÁ**

* **Chuẩn bị cho phần thuyết trình:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thứ tự** | **Họ tên** | **Mã số sinh viên** | **Nội dung công việc** | **Mức độ đóng góp** | **Giảng viên đánh giá** |
| 1 | Nguyễn Quốc Huy | 3122410149 | Soạn lý thuyết báo cáo, chuẩn bị Powerpoint | 25% |  |
| 2 | Trương Gia Huy | 3122410154 | Soạn lý thuyết báo cáo, chuẩn bị Powerpoint | 25% |  |
| 3 | Nguyễn Văn Vũ Em | 3122410090 | Soạn lý thuyết báo cáo, chuẩn bị Powerpoint | 25% |  |
| 4 | Nguyễn Vũ Hào | 3122410098 | Soạn lý thuyết báo cáo, chuẩn bị Powerpoint | 25% |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Thứ tự** | **Họ tên** | **Mã số sinh viên** | **Nội dung công việc** | **Mức độ đóng góp** | **Giảng viên đánh giá** |
| 1 | Nguyễn Quốc Huy | 3122410149 | Frame main, train, upload hình ảnh(back-end), scan, nav(back-end), thông tin biển báo | 25% |  |
| 2 | Trương Gia Huy | 3122410154 | Thông tin biển báo, đăng nhập đăng kí, thông tin biển báo, train | 25% |  |
| 3 | Nguyễn Văn Vũ Em | 3122410090 | Đăng nhập đăng kí, upload hình ảnh, admin, train | 25% |  |
| 4 | Nguyễn Vũ Hào | 3122410098 | Nav, thông tin app, VIP, upload hình, admin, thiết kế giao diện cho tất cả frame | 25% |  |

* **Phần kĩ thuật và thực hành đồ án:**

**NHẬN XÉT CỦA GIẢNG VIÊN**

MỤC LỤC

[LỜI MỞ ĐẦU 5](#_Toc165155458)

[Lý do chọn đề tài: 5](#_Toc165155459)

[CHƯƠNG 1: CƠ SỞ LÝ THUYẾT 6](#_Toc165155460)

[1. Giới thiệu về ngôn ngữ Python 6](#_Toc165155461)

[1.1 Ngôn Ngữ Python là gì? 6](#_Toc165155462)

[1.2 Lợi ích của Python 6](#_Toc165155463)

[1.3 Ứng dụng của Python 7](#_Toc165155464)

[2. Các thư viện của Python được sử dụng 7](#_Toc165155465)

[2.1 Back-end 7](#_Toc165155466)

[2.2 Front-end 10](#_Toc165155467)

[4. Lý thuyết CNN 19](#_Toc165155468)

[4.1 Deep learning 19](#_Toc165155469)

[4.2 Convolutional Neural Network (CNN): 20](#_Toc165155470)

[CHƯƠNG 2: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN BIỂN BÁO GIAO THÔNG 22](#_Toc165155471)

[1. Phân tích bài toán cơ bản (Chương trình nhận diện biển báo giao thông) 22](#_Toc165155472)

[2. Phân tích bài toán ứng dụng AI (Nhận diện hình ảnh) 22](#_Toc165155473)

[CHƯƠNG 3: CÀI ĐẶT, DEMO CHƯƠNG TRÌNH 24](#_Toc165155474)

[1. Cài đặt 24](#_Toc165155475)

[2. Demo chương trình 27](#_Toc165155476)

[KẾT LUẬN 40](#_Toc165155477)

[1. ĐÁNH GIÁ NHỮNG NỘI DUNG ĐÃ THỰC HIỆN 40](#_Toc165155478)

[2. ĐÁNH GIÁ NHỮNG NỘI DUNG CHƯA THỰC HIỆN 40](#_Toc165155479)

[3. HƯỚNG PHÁT TRIỂN 40](#_Toc165155480)

# LỜI MỞ ĐẦU

## Lý do chọn đề tài:

Trong thời đại công nghệ phát triển mạnh mẽ như ngày nay, việc ứng dụng trí tuệ nhân tạo vào các lĩnh vực thực tế đã trở thành xu hướng không thể phủ nhận. Trong lĩnh vực giao thông, việc nhận diện và hiểu biển báo giao thông đóng vai trò quan trọng trong việc đảm bảo an toàn cho người tham gia giao thông. Với sự phát triển của công nghệ, các phương pháp nhận diện biển báo giao thông sử dụng mạng neural convolutional (CNN) và ngôn ngữ lập trình Python (Py) đang trở nên ngày càng phổ biến và hứa hẹn mang lại những kết quả đáng kể.

Lý do chúng em chọn đề tài áp dụng CNN và Py để nhận diện biển báo giao thông là vì những ưu điểm mà hai công nghệ này mang lại. Mạng neural convolutional (CNN) là một kiến trúc mạng nơ-ron có khả năng học và nhận dạng các đặc trưng trong hình ảnh một cách hiệu quả. Điều này rất phù hợp trong việc nhận diện biển báo giao thông, vì biển báo giao thông có các đặc trưng đặc biệt và đa dạng. CNN giúp chúng ta xử lý và hiểu các biển báo này một cách tự động và chính xác.

Bên cạnh đó, ngôn ngữ lập trình Python (Py) là một ngôn ngữ linh hoạt, dễ học và dễ sử dụng. Python có nhiều thư viện hỗ trợ xử lý hình ảnh và machine learning, giúp chúng ta triển khai và xây dựng mô hình nhận diện biển báo giao thông một cách nhanh chóng và hiệu quả.

Trong đề tài này, chúng em hy vọng áp dụng CNN và Py vào việc nhận diện biển báo giao thông để tạo ra một hệ thống thông minh, giúp người lái xe nhận biết và tuân thủ quy tắc giao thông một cách tốt nhất. Qua đó, không chỉ nâng cao sự an toàn giao thông mà còn đóng góp vào sự phát triển và ứng dụng của công nghệ trong cuộc sống hàng ngày.

# CHƯƠNG 1: CƠ SỞ LÝ THUYẾT

## 1. Giới thiệu về ngôn ngữ Python

### 1.1 Ngôn Ngữ Python là gì?

Python là một ngôn ngữ lập trình bậc cao, được tạo ra bởi Guido van Rossum vào những năm 1980 và phát triển tiếp theo trong thập niên 1990. Python là một ngôn ngữ lập trình đơn giản, dễ học và dễ sử dụng, có thể được sử dụng để xây dựng các ứng dụng máy tính, trang web, các hệ thống đám mây, trí tuệ nhân tạo, máy học và các ứng dụng phân tích dữ liệu. Python cũng là một ngôn ngữ lập trình đa nền tảng, có thể chạy trên các hệ điều hành khác nhau như Windows, Linux và MacOS. Python có một cộng đồng lớn và đa dạng, với hàng ngàn thư viện và framework hỗ trợ cho việc phát triển ứng dụng.

### 1.2 Lợi ích của Python

+ *Dễ học và sử dụng:* Python có cú pháp đơn giản và dễ hiểu, giúp người mới bắt đầu học lập trình dễ dàng tiếp cận và làm quen với các khái niệm lập trình cơ bản.

+ *Đa năng:* Python có thể được sử dụng để phát triển các loại ứng dụng khác nhau như ứng dụng máy tính, web, trí tuệ nhân tạo, máy học, phân tích dữ liệu, đám mây, hệ thống nhúng và nhiều hơn nữa.

+ *Cộng đồng lớn và đa dạng:* Python có một cộng đồng phát triển lớn và đa dạng, với hàng ngàn thư viện và framework hỗ trợ cho việc hát triển ứng dụng.

+ *Tính tương thích cao:* Python là một ngôn ngữ đa nền tảng, có thể chạy trên nhiều hệ điều hành khác nhau như Windows, Linux và MacOS.

+ *Hiệu suất cao:* Python có thể được tối ưu hóa để đạt được hiệu suất cao, đặc biệt là khi sử dụng các thư viện và framework được viết bằng C/C++.

+ *Dễ dàng mở rộng:* Python có tính mở rộng cao, cho phép người dùng tạo ra các module và thư viện riêng để mở rộng chức năng của Python.

+ *Tương tác tốt với các ngôn ngữ khác:* Python có thể tương tác tốt với các ngôn ngữ khác như C/C++, Java, và .NET, giúp cho việc tích hợp các phần mềm và dịch vụ khác nhau trở nên dễ dàng hơn.

### 1.3 Ứng dụng của Python

Dưới đây là một số ví dụ về các ứng dụng của Python:

+ *Phát triển ứng dụng web*: Python có thể được sử dụng để phát triển các ứng dụng web, với các framework như Django, Flask và Pyramid.

+ *Trí tuệ nhân tạo và máy học*: Python là một trong những ngôn ngữ phổ biến nhất trong lĩnh vực trí tuệ nhân tạo và máy học, với các thư viện như TensorFlow, Keras và PyTorch.

+ *Phân tích dữ liệu*: Python có các thư viện như Pandas và NumPy để xử lý và phân tích dữ liệu, và Matplotlib và Seaborn để tạo đồ thị và biểu đồ.

+ *Game development*: Python có thể được sử dụng để phát triển game với các thư viện như Pygame và PyOpenGL.

+ *Đám mây*: Python có thể được sử dụng để quản lý và triển khai các hệ thống đám mây với các framework như OpenStack và Ansible.

+ *Hệ thống nhúng*: Python có thể được sử dụng để phát triển các hệ thống nhúng, với các framework như MicroPython và CircuitPython.

+ *Ứng dụng máy tính*: Python có thể được sử dụng để phát triển các ứng dụng máy tính trên nhiều nền tảng khác nhau, với các thư viện như PyQt và wxPython.

## 2. Các thư viện của Python được sử dụng

Các thư viện được sử dụng trong đồ án:

* *Import os*
* *Import cv2*
* *Import numpy*
* *Datetime from datetime*
* *Thư viện Google Auth trong Kivy*
* *Import re*
* *Kivy và KivyMD*

### 2.1 Back-end

#### 2.1.1 Import os

Thư viện os cung cấp các phương thức để tương tác với hệ thống tệp và thư mục trong Python. Bạn có thể sử dụng nó để thực hiện các hoạt động như tạo, xóa, di chuyển hoặc đổi tên tệp.

Trong code của nhóm em thì có sử dụng **os:**

+ Để tạo đường dẫn đến thư mục chứa biển báo giao thông trong biến image\_dir.

+ Mở file ảnh biển báo giao thông từ thư mục tạo đường dẫn đến file ảnh có thể là một cách thực tiễn khi cần tương tác với nhiều hệ điều hành khác nhau. Tuy nhiên việc này không được thực hiện trực tiếp.

+ Kiểm tra sự tồn tại của thư mục khi truy cập từng biển báo để tránh lỗi không tồn tại hoặc đã bị xóa đi.

#### 2.1.2 Import cv2

Thư viện OpenCV(cv2) là một thư viện mã nguồn mở chuyên dụng cho xử lý hình ảnh và thị giác máy tính. Nó cung cấp các công cụ để đọc, ghi, xử lý và phân tích hình ảnh, cũng như các chức năng nhận dạng đối tượng, phát hiện khuôn mặt, và nhiều hơn nữa.

Cung cấp các chức năng để thực hiện các bước sau:

+ Đọc và xử lý hình ảnh trong đường dẫn img\_path.

+ Chuyển đổi định dạng màu sắc hình ảnh từ RGBA sang RGB.

+ Phân loại hình ảnh, hình ảnh được truyền vào 1 mô hình đã được huấn luyện để phân loại.

#### 2.1.3 Import numpy

Thư viện numpy là một thư viện Python mạnh mẽ cho tính toán khoa học và toán học. Nó cung cấp một đối tượng mảng nhiều chiều (ndarray), các hàm để làm việc với mảng này, và một loạt các công cụ để làm việc với mảng số.

Cung cấp các chức năng để thực hiện các bước sau:

+ *Chuyển đổi hình ảnh*: NumPy cung cấp các phương pháp để chuyển đổi hình ảnh từ một loạt các định dạng khác nhau thành các mảng NumPy, thuận tiện cho việc xử lý tiếp theo.

+ *Thao tác ma trận*: NumPy cung cấp một loạt các hàm và phương thức để thực hiện các phép toán ma trận, như cộng, trừ, nhân, chia, transpoze, và nhiều phép toán khác. Điều này hữu ích trong việc xử lý dữ liệu hình ảnh, ví dụ như khi áp dụng các bộ lọc hình ảnh hoặc biến đổi hình ảnh.

+ *Xử lý dữ liệu số*: NumPy cung cấp các công cụ mạnh mẽ để thực hiện các phép toán số học và tính toán khoa học trên dữ liệu số.Chúng tôi có thể sử dụng NumPy để xử lý các dữ liệu kết quả từ việc nhận diện biển báo giao thông hoặc xử lý hình ảnh.

#### 2.1.4 From datetime import datetime

Thư viện datetime trong Python cung cấp các lớp để làm việc với ngày tháng và thời gian. Bằng cách này,chúng tôi có thể sử dụng datetime.now() để lấy thời gian hiện tại để lưu vào file khi quét các biển báo.

#### 2.1.5 Thư viện Google Auth trong Kivy (kivyauth.google\_auth)

Cung cấp các công cụ để tích hợp chức năng đăng nhập bằng tài khoản Google vào ứng dụng. Khi sử dụng thư viện này có thể tạo ra một quy trình đăng nhập đơn giản và an toàn cho người dùng của mình, cho phép họ đăng nhập vào ứng dụng bằng tài khoản Google của họ.

Cung cấp các chức năng để thực hiện các bước sau:

+ *Xác thực tài khoản Google*: Thư viện cho phép ứng dụng của bạn gửi yêu cầu xác thực đến Google, và Google sẽ trả về một mã xác thực sau khi người dùng đăng nhập thành công.

+ *Lấy thông tin người dùng*: Sau khi xác thực thành công, thư viện giúp bạn lấy thông tin về người dùng từ tài khoản Google của họ, như tên, hình ảnh đại diện, địa chỉ email, vv. Điều này giúp bạn tạo ra trải nghiệm cá nhân hóa cho người dùng của mình.

+ *Quản lý phiên làm việc*: Thư viện cung cấp cơ chế để quản lý phiên làm việc của người dùng sau khi họ đăng nhập. Điều này bao gồm việc lưu trữ thông tin phiên làm việc và xác thực phiên làm việc để đảm bảo tính bảo mật của ứng dụng.

Tóm lại, việc sử dụng thư viện Google Auth trong Kivy giúp chúng ta tích hợp tính năng đăng nhập bằng tài khoản Google vào ứng dụng của mình một cách dễ dàng và an toàn, cung cấp một trải nghiệm đăng nhập thuận tiện cho người dùng.

#### 2.1.6 Import re

Thư viện re trong Python là một công cụ mạnh mẽ để làm việc với các biểu thức chính quy (regular expressions). Biểu thức chính quy là một chuỗi ký tự mô tả một tập hợp các chuỗi. Khi bạn import re, bạn có thể sử dụng các phương thức của nó để tìm kiếm, thay thế và phân tích các chuỗi dựa trên mẫu đã định sẵn.

Cung cấp các chức năng để thực hiện các bước sau:

+ *Tìm kiếm và so khớp chuỗi*: Bạn có thể sử dụng re.search() hoặc re.match() để tìm kiếm một mẫu trong một chuỗi và trả về vị trí xuất hiện của mẫu nếu có.

+ *Thay thế chuỗi*: re.sub() cho phép bạn thay thế các phần của chuỗi phù hợp với mẫu bằng một chuỗi khác.

+ *Phân tích chuỗi*: Bạn có thể sử dụng re.split() để phân tách chuỗi thành các phần dựa trên một mẫu.

Các phép toán khác với biểu thức chính quy: re.findall(), re.finditer(), re.fullmatch() và các phương thức khác cho phép bạn thực hiện các hoạt động khác nhau với các biểu thức chính quy.

### 2.2 Front-end

*Sử dụng các class thuộc thư viện Kivy và KivyMD. Tổng quan các điều chỉnh thuộc tính khá tương đồng giữa hai module. Có thể sử dụng trực tiếp trong file kivy mà không cần import từ trước.*

#### 2.2.1 MDApp

*“from kivymd.app import MDApp”*: Import lớp MDApp từ module kivymd.app. Lớp này là lớp cơ sở cho ứng dụng KivyMD của bạn. Bạn sẽ tạo một lớp con kế thừa từ MDApp để xây dựng ứng dụng của mình.

#### 2.2.2 Window

Window trong Kivy là một đối tượng toàn cục cung cấp các thuộc tính và phương thức để điều khiển cửa sổ của ứng dụng.

Dưới đây là một số tính năng quan trọng của Window:

+ *Kích thước cửa sổ*: Bạn có thể đặt kích thước cửa sổ của ứng dụng bằng cách sử dụng các thuộc tính như Window.width và Window.height.

+ *Tiêu đề cửa sổ*: Bạn có thể đặt tiêu đề của cửa sổ bằng cách sử dụng thuộc tính Window.title.

+ *Các sự kiện của cửa sổ*: Window cung cấp các sự kiện như on\_close, on\_resize, on\_request\_close, cho phép bạn xử lý các sự kiện liên quan đến cửa sổ của ứng dụng, chẳng hạn như khi người dùng đóng cửa sổ.

+ *Thuộc tính hiển thị*: Bạn có thể điều chỉnh thuộc tính hiển thị của cửa sổ như độ phân giải, tỷ lệ khung hình và có hiển thị toàn màn hình hay không.

#### 2.2.3 ScreenManager, Screen, MDScreen

ScreenManager:

+ ScreenManager là một phần quan trọng của Kivy, cho phép bạn quản lý và chuyển đổi giữa các màn hình trong ứng dụng.

+ Đầu tiên, tạo một đối tượng ScreenManager bằng cách sử dụng ScreenManager().

+ Thêm các màn hình vào ScreenManager bằng cách sử dụng phương thức add\_widget(screen), với screen là một đối tượng Screen mà bạn muốn thêm.

+ Để hiển thị một màn hình cụ thể, sử dụng thuộc tính current của ScreenManager để thiết lập tên của màn hình muốn hiển thị. *Ví dụ*: screen\_manager.current = 'screen1'.

+ Sử dụng các phương thức như switch\_to() để chuyển đổi giữa các màn hình.

Screen:

+ Mỗi Screen đại diện cho một màn hình trong ứng dụng.

+ Tạo một Screen mới bằng cách sử dụng Screen() và chỉ định một số thuộc tính, như name để xác định tên của Screen.

+ Sau khi tạo Screen, bạn có thể thêm các thành phần giao diện người dùng khác nhau vào bên trong Screen, ví dụ như Label, Button, BoxLayout, và nhiều hơn nữa.

+ Một khi bạn đã thêm các thành phần mong muốn vào Screen, bạn cần thêm Screen đó vào ScreenManager để nó có thể được hiển thị và quản lý.

MDScreen:

+ Là một màn hình có thiết kế theo Material Design.

+ Là một phần của KivyMD, được sử dụng để tạo các màn hình trong ứng dụng. Màn hình là không gian hiển thị trên giao diện người dùng của ứng dụng, nơi bạn có thể đặt các thành phần khác nhau như nút, nhãn, hộp thoại, và nhiều hơn nữa.

*MDScreen cung cấp các tính năng như:*

+ Xây dựng và quản lý các màn hình trong ứng dụng.

+ Tích hợp với các thành phần khác của KivyMD để tạo ra giao diện người dùng có phong cách Material Design.

+ Hỗ trợ chuyển đổi giữa các màn hình một cách dễ dàng thông qua ScreenManager.

+ Khi import MDScreen, có thể tạo ra các màn hình và điều chỉnh chúng để hiển thị nội dung mong muốn trong ứng dụng. Có thể thêm các thành phần giao diện khác nhau vào mỗi màn hình và quản lý chúng theo cách thích hợp.

Tích hợp:

+ ScreenManager và Screen(MDScreen) thường được sử dụng cùng nhau để tạo ra giao diện người dùng có nhiều màn hình trong ứng dụng.

+ Có thể tạo một loạt các Screen(MDScreen), mỗi Screen biểu diễn một phần cụ thể của ứng dụng, và sử dụng ScreenManager để chuyển đổi giữa chúng khi cần thiết.

#### 2.2.4 Widget

Widget là lớp cơ sở cho tất cả các thành phần giao diện người dùng (GUI) trong Kivy.

+ Bất kỳ thành phần giao diện nào, chẳng hạn như nút, nhãn, hộp thoại, hoặc bất kỳ thành phần tùy chỉnh nào tạo ra trong Kivy, đều phải kế thừa từ lớp Widget.

+ Widget cung cấp các tính năng cơ bản như định vị, kích thước, và chứa các thành phần con bên trong.

+ Có thể tùy chỉnh và mở rộng lớp Widget để tạo ra các thành phần giao diện phức tạp và đa dạng cho ứng dụng.

#### 2.2.5 Camera

Camera:

+ Lớp Camera trong Kivy được sử dụng để hiển thị hình ảnh từ máy ảnh của thiết bị hoặc từ nguồn video khác.

+ Khi sử dụng Camera trong ứng dụng, nó sẽ kích hoạt máy ảnh của thiết bị và hiển thị hình ảnh trực tiếp từ máy ảnh đó trên giao diện người dùng.

+ Sử dụng Camera để tạo các ứng dụng liên quan đến việc chụp ảnh hoặc quay video, hoặc để thực hiện nhận dạng hình ảnh hoặc các tác vụ xử lý hình ảnh.

#### 2.2.6 Image, AsyncImage

Image:

+ Lớp Image trong Kivy được sử dụng để hiển thị hình ảnh tĩnh.

+ Sử dụng Image để hiển thị hình ảnh từ các tệp tin cục bộ trên thiết bị hoặc từ URL trên Internet.

+ Đối với hình ảnh tĩnh, cần tải hình ảnh hoặc chỉ định đường dẫn đến hình ảnh muốn hiển thị.

AsyncImage:

+ Lớp AsyncImage trong Kivy cung cấp khả năng hiển thị hình ảnh từ URL một cách bất đồng bộ.

+ Khi sử dụng AsyncImage để hiển thị hình ảnh từ một URL, quá trình tải hình ảnh sẽ không làm đóng băng giao diện người dùng, điều này giúp tránh tình trạng giao diện đóng băng khi tải hình ảnh từ Internet.

+ Chỉ cần chỉ định URL của hình ảnh muốn hiển thị, và AsyncImage sẽ tự động tải và hiển thị hình ảnh đó.

#### 2.2.7 Button

Button: là một thành phần giao diện người dùng cho phép người dùng tương tác bằng cách nhấn vào.

Các đặc điểm chính:

+ *Văn bản*: Bạn có thể đặt văn bản trên nút bằng cách sử dụng thuộc tính text.

+ *Hành vi*: Bạn có thể gán một hàm hoặc phương thức để được gọi khi người dùng nhấn vào nút, thông qua sự kiện on\_press hoặc on\_release.

+ *Màu sắc và giao diện*: Button có các thuộc tính cho phép bạn tùy chỉnh màu sắc, kích thước, hình dạng và kiểu chữ.

+ *Trạng thái*: Bạn có thể kích hoạt hoặc vô hiệu hóa một nút, hoặc thay đổi trạng thái của nó, như nút nhấn hoặc nút chức năng thông qua thuộc tính disabled.

Button là một phần cơ bản nhưng quan trọng trong việc xây dựng giao diện người dùng trong Kivy, và nó thường được sử dụng để tạo ra các tương tác người dùng cơ bản như kích hoạt hành động hoặc chức năng.

#### 2.2.8 MDFlatButton, MDFloatingActionButton

MDFlatButton:

+ Là một lớp được cung cấp bởi KivyMD để tạo ra các nút phẳng.

+ Nút phẳng không có hiệu ứng nhấp nháy hoặc độ lồi, chúng có vẻ như phẳng trên giao diện người dùng.

+ Thường được sử dụng cho các hành động phụ hoặc các hành động không quan trọng như "Cancel", "Dismiss", hoặc "OK".

MDFloatingActionButton:

+ Là một lớp được cung cấp bởi KivyMD để tạo ra các nút hành động phản ứng.

+ Nút hành động phản ứng thường có hình dạng hình tròn và nổi bật hơn so với các nút thông thường.

+ Được sử dụng cho các hành động quan trọng hoặc các hành động mà người dùng thực hiện thường xuyên, chẳng hạn như "Add", "Edit", hoặc "Save".

+ Cả hai lớp đều cung cấp một số thuộc tính và phương thức để tùy chỉnh và điều chỉnh hình dạng, màu sắc, kích thước và hành vi của nút.

#### 2.2.9 Label (MDLabel)

Label(MDLabel): Là một thành phần giao diện người dùng được sử dụng để hiển thị văn bản tĩnh trên màn hình.

Các đặc điểm chính:

+ *Văn bản*: Bạn có thể đặt văn bản để hiển thị trên Label thông qua thuộc tính text.

+ *Căn chỉnh và định dạng văn bản*: Label cho phép bạn căn chỉnh văn bản theo chiều ngang và dọc, thay đổi kích thước của văn bản, font chữ, màu sắc và kiểu chữ.

+ *Trạng thái*: Tương tự như Button, bạn cũng có thể vô hiệu hóa hoặc kích hoạt một Label thông qua thuộc tính disabled.

Label thường được sử dụng để hiển thị tiêu đề, thông điệp hoặc nội dung tĩnh khác trên giao diện người dùng, và nó là một thành phần cơ bản nhưng quan trọng trong việc xây dựng giao diện người dùng trong Kivy.

#### 2.2.10 BoxLayout (MDBoxLayout), GridLayout (MDGridLayout)

BoxLayout(MDBoxLayout): là một layout manager được sử dụng để sắp xếp các thành phần giao diện người dùng theo chiều ngang hoặc dọc.

Các đặc điểm chính:

+ *Hướng sắp xếp*: Bạn có thể chọn hướng sắp xếp là theo chiều ngang ('horizontal') hoặc theo chiều dọc ('vertical') bằng cách sử dụng thuộc tính orientation.

+ *Căn chỉnh và định dạng*: Bạn có thể tùy chỉnh cách các thành phần giao diện được căn chỉnh theo hướng sắp xếp, cũng như khoảng cách giữa chúng thông qua các thuộc tính như spacing, padding, và size\_hint.

GridLayout(MDGridLayout): là một layout manager được sử dụng để sắp xếp các thành phần giao diện người dùng vào một lưới có số hàng và cột cố định.

Các đặc điểm chính:

+ *Số hàng và cột*: Bạn có thể xác định số hàng và số cột của lưới thông qua thuộc tính rows và cols.

+ *Sắp xếp tự động*: Các thành phần giao diện được tự động sắp xếp vào các ô của lưới theo thứ tự từ trái qua phải, từ trên xuống dưới.

#### 2.2.11 ScrollView(MDScrollView)

ScrollView(MDScrollView): được sử dụng để tạo ra một vùng hiển thị cuộn (scrollable area) cho nội dung.

ScrollView cho phép:

+ *Cuộn nội dung*: Hiển thị nội dung vượt ra ngoài kích thước của vùng hiển thị bằng cách tạo thanh cuộn hoặc cho phép cuộn bằng cách di chuyển chuột hoặc chạm trên màn hình cảm ứng.

+ *Chứa nhiều thành phần*: Bạn có thể đặt nhiều thành phần con trong ScrollView, chẳng hạn như các nhãn, nút, hình ảnh, hoặc bất kỳ thành phần giao diện nào khác.

+ *Tạo nội dung động*: Nội dung trong ScrollView có thể được tạo ra động dựa trên dữ liệu hoặc điều kiện trong ứng dụng của bạn.

#### 2.2.12 MDList, ThreeLineAvatarIconListItem, OneLineAvatarIconListItem, IconLeftWidget

MDList:

+ Cho phép hiển thị một danh sách các mục theo các hướng khác nhau, bao gồm theo chiều dọc (vertical) hoặc chiều ngang (horizontal). Điều này cho phép tạo ra các loại danh sách khác nhau tùy thuộc vào nhu cầu của ứng dụng.

+ *Cấu trúc đơn giản*: Để thêm mục vào MDList, có thể sử dụng phương thức add\_widget() để thêm các thành phần được hiển thị như các mục trong danh sách.

+ *Tùy chỉnh mục*: Mỗi mục trong MDList có thể được tùy chỉnh để phản ánh nội dung mong muốn. Có thể sử dụng các thành phần như MDListItem, OneLineListItem, TwoLineListItem, ThreeLineListItem và IconLeftWidget để tạo ra các mục với độ phong phú khác nhau, bao gồm cả văn bản, hình ảnh và biểu tượng.

+ *Tương tác người dùng*: MDList hỗ trợ các sự kiện người dùng như nhấp vào mục (on\_press), giữ mục (on\_long\_press), và các sự kiện tương tác khác. Điều này cho phép xử lý các hành động người dùng một cách linh hoạt và tiện lợi.

+ *Thiết kế đẹp mắt*: Với các thành phần thiết kế từ KivyMD, MDList cho phép bạn tạo ra các danh sách với giao diện người dùng đẹp mắt và chuyên nghiệp. Bạn có thể tùy chỉnh màu sắc, kích thước và kiểu dáng của các mục để phù hợp với phong cách thiết kế của ứng dụng.

ThreeLineAvatarIconListItem:

+ Là một lớp được cung cấp bởi KivyMD (Kivy Material Design) để tạo ra một mục danh sách có ba dòng văn bản, biểu tượng và ảnh đại diện.

+ Lớp này thường được sử dụng để hiển thị các mục danh sách phức tạp với thông tin chi tiết.

OneLineAvatarIconListItem: tương tự ThreeLine nhưng chỉ có một dòng văn bản thay vì ba.

IconLeftWidget:

+ Là một lớp widget được sử dụng để hiển thị biểu tượng bên trái của một thành phần giao diện khác.

+ Thường được sử dụng bên trong các mục danh sách hoặc các thành phần giao diện khác như MDList trong KivyMD.

+ IconLeftWidget thường được kết hợp với các lớp khác như MDIconButton, MDIcon, hoặc các lớp widget tùy chỉnh để tạo ra các thành phần giao diện với biểu tượng bên trái.

*2.2.13 MDTopAppBar*

MDTopAppBar: được sử dụng để tạo thanh ứng dụng trên cùng trong bố cục ứng dụng của bạn.

Nó thường chứa tiêu đề, biểu tượng điều hướng, biểu tượng hành động và cũng có thể chứa các tiện ích khác nếu cần.

Đặc trưng:

+ *Tiêu đề*: Nó có thể hiển thị tiêu đề ở giữa thanh ứng dụng.

+ *Biểu tượng điều hướng*: Nó có thể hiển thị một biểu tượng ở phía bên trái của thanh ứng dụng, thường được sử dụng cho mục đích điều hướng (ví dụ: mở menu bên).

+ *Biểu tượng hành động*: Nó có thể hiển thị các biểu tượng bổ sung ở bên phải thanh ứng dụng, thường để thực hiện các hành động như tìm kiếm, cài đặt hoặc bất kỳ hành động tùy chỉnh nào.

+ *Màu nền*: Bạn có thể tùy chỉnh màu nền của thanh ứng dụng để phù hợp với thiết kế ứng dụng của bạn.

Cách sử dụng:

+ Để sử dụng MDTopAppBar, bạn thường tạo một phiên bản của nó và thêm nó vào bố cục ứng dụng của mình.

+ Bạn có thể đặt văn bản tiêu đề bằng thuộc tính title.

+ Bạn có thể đặt biểu tượng điều hướng bằng left\_action\_items.

+ Bạn có thể thêm các biểu tượng hành động bằng right\_action\_items.

#### 2.2.14 NavigationDrawler

Navigation Drawer: là một menu hiển thị như một ngăn xếp ở cạnh viền của màn hình. Nó được ẩn đi khi không sử dụng, nhưng sẽ xuất hiện khi người sử dụng vuốt ngón tay của họ từ mép trái của màn hình, vuốt từ mép trên của màn hình ứng dụng hoặc bằng thanh điều hướng của MDTopAppBar.

Cung cấp các tính năng:

+ *Điều hướng*: Nó cho phép người dùng điều hướng giữa các màn hình hoặc phần khác nhau của ứng dụng. Mỗi mục trong ngăn điều hướng đại diện cho một màn hình hoặc chức năng cụ thể và việc nhấn vào một mục thường sẽ chuyển người dùng sang màn hình đó.

+ *Mục menu*: Ngăn điều hướng chứa danh sách các mục menu đóng vai trò là điểm vào các phần khác nhau của ứng dụng. Các mục menu này thường được thể hiện dưới dạng biểu tượng hoặc nhãn văn bản.

+ *Tiêu đề*: Nó có thể bao gồm phần tiêu đề ở đầu ngăn điều hướng, có thể hiển thị thông tin người dùng, nhãn hiệu ứng dụng hoặc nội dung liên quan khác.

+ *Chân trang*: Tùy chọn, phần chân trang có thể được đưa vào cuối ngăn điều hướng, cung cấp các tùy chọn, cài đặt hoặc liên kết bổ sung.

+ *Chuyển đổi*: Ngăn điều hướng thường được kích hoạt bằng cách nhấn vào nút hoặc biểu tượng chuyển đổi.

## 4. Lý thuyết CNN

### 4.1 Deep learning

Là một lĩnh vực của máy học (machine learning) tập trung vào việc xây dựng và huấn luyện các mạng neuron sâu (deep neural networks) để học biểu diễn dữ liệu phức tạp từ các dạng đầu vào không cấu trúc hoặc dữ liệu cấu trúc có chiều cao. Cùng với CNN, deep learning thường được sử dụng để giải quyết các bài toán như phân loại ảnh, nhận dạng văn bản, dịch máy, và nhiều ứng dụng khác.

Các mạng neuron sâu (deep neural networks) bao gồm nhiều lớp neuron kết nối với nhau thành các cấu trúc phức tạp. Một số loại mạng neuron sâu phổ biến bao gồm:

**+ Mạng nơ-ron Lan truyền tiến (Feedforward Neural Networks):** Đây là một trong những loại mạng neuron đơn giản nhất, trong đó dữ liệu chảy qua từ lớp đầu vào đến lớp đầu ra mà không có kết nối phản hồi (feedback).

**+ Mạng Nơ-ron Hồi quy (Recurrent Neural Networks - RNNs):** Loại mạng này có khả năng xử lý dữ liệu có cấu trúc dạng chuỗi, như ngôn ngữ tự nhiên hoặc dữ liệu chuỗi thời gian. Các lớp trong RNNs có kết nối phản hồi, cho phép chúng nhớ thông tin từ quá khứ.

**+ Mạng Nơ-ron Hồi quy Dài-ngắn (Long Short-Term Memory - LSTM Networks):** Một dạng đặc biệt của RNNs, LSTM có khả năng học và lưu trữ thông tin trong thời gian dài, giúp giải quyết vấn đề triệt tiêu biến mất hoặc bùng nổ đạo hàm trong quá trình huấn luyện.

**+ Mạng Nơ-ron Con đỏ (Convolutional Neural Networks - CNNs)**: Loại mạng này được thiết kế đặc biệt cho việc xử lý và phân tích dữ liệu không gian như ảnh và video, thông qua việc sử dụng các lớp convolutional và pooling.

**+ Mạng Nơ-ron Gần sâu (Deep Belief Networks - DBNs):** Loại mạng này thường sử dụng trong các ứng dụng như phân loại ảnh và nhận dạng âm thanh.

Tuy nhiên trong đồ án này thì chúng tôi sử dụng kỹ thuật **CNN** vìviệc sử dụng deep learning có thể bao gồm sử dụng các kiến trúc mạng neuron sâu để xây dựng mô hình nhận dạng biển báo giao thông từ dữ liệu hình ảnh và huấn luyện mô hình này để có thể phân loại các biển báo trong thời gian thực.

### 4.2 Convolutional Neural Network (CNN):

+ CNN là một dạng của mạng nơ-ron nhân tạo (neural network) được thiết kế đặc biệt cho việc xử lý dữ liệu không gian như hình ảnh.

+ Cấu trúc của CNN bao gồm các lớp chính như lớp convolution (tích chập), lớp pooling (tổng hợp), và các lớp kết nối đầy đủ (fully connected layers).

+ Lớp convolution giúp mô hình học được các đặc trưng cục bộ từ dữ liệu đầu vào bằng cách áp dụng các bộ lọc (kernels) qua ảnh.

+ Lớp pooling giúp giảm kích thước của đầu vào và giữ lại các đặc trưng quan trọng.

Các lớp kết nối đầy đủ được sử dụng để kết nối các đặc trưng đã học được với các lớp đầu ra để thực hiện phân loại.

**+** *Lớp Convolutional (Convolutional Layer):*

* Lớp này chịu trách nhiệm xử lý đầu vào bằng cách áp dụng các bộ lọc (kernels) qua ảnh đầu vào để tạo ra các đặc trưng cục bộ.
* Các bộ lọc này giúp phát hiện các đặc trưng như cạnh, góc, hoặc các hình dạng cụ thể trong ảnh.
* Kết quả của lớp convolution thường được gọi là các feature map.

**+** *Lớp Tổng hợp (Pooling Layer):*

* Lớp này giúp giảm kích thước của feature map, đồng thời giữ lại các thông tin quan trọng.
* Các phương pháp phổ biến trong pooling là max pooling và average pooling, trong đó max pooling giữ lại giá trị lớn nhất trong mỗi vùng và bỏ qua các giá trị khác

.

**+** *Lớp Kích hoạt (Activation Layer):*

* Lớp này thường được đặt sau lớp convolution và pooling để tạo ra các biểu diễn phi tuyến tính từ đầu vào.
* Hàm kích hoạt phổ biến nhất là ReLU (Rectified Linear Activation), nhưng cũng có thể sử dụng các hàm khác như tanh hoặc sigmoid.
* Lớp Kết nối Đầy đủ (Fully Connected Layer):
* Lớp này thường được sử dụng ở cuối mạng để kết nối các đặc trưng đã học được với các lớp đầu ra để thực hiện phân loại.
* Các lớp fully connected được sử dụng để học các mối quan hệ phức tạp giữa các đặc trưng và lớp đầu ra.

**Quy trình Huấn luyện (Training Process):**

- Quy trình huấn luyện của mạng CNN thường bao gồm các bước như sau:

* *Định nghĩa mạng*: Xác định kiến trúc mạng CNN bao gồm số lượng và kiểu của các lớp, kích thước của các bộ lọc, số lượng lớp và kích thước lớp đầu ra.
* *Chuẩn bị dữ liệu*: Tiền xử lý dữ liệu, chia dữ liệu thành các batch, chuẩn hóa dữ liệu nếu cần.
* *Chọn hàm mất mát và thuật toán tối ưu*: Phổ biến nhất là cross-entropy loss function và thuật toán gradient descent hoặc các biến thể như Adam.
* *Huấn luyện mô hình*: Đưa dữ liệu vào mô hình để huấn luyện, điều chỉnh các tham số (weights) sao cho hàm mất mát được giảm thiểu.
* *Đánh giá mô hình*: Sử dụng tập dữ liệu kiểm tra để đánh giá hiệu suất của mô hình, thực hiện đánh giá về độ chính xác hoặc các metric khác.
* Trong mô hình CNN của bạn, quy trình huấn luyện sẽ được thực hiện để học các biểu diễn của các loại biển báo giao thông từ dữ liệu huấn luyện, và sau đó mô hình được sử dụng để phân loại các biển báo mới từ các hình ảnh mới.

Trong đồ án này, mô hình CNN và deep learning được sử dụng để học các mẫu và đặc trưng từ các hình ảnh của các biển báo giao thông, từ đó có thể phân loại chúng thành các loại biển báo khác nhau.

# CHƯƠNG 2: XÂY DỰNG CHƯƠNG TRÌNH NHẬN DIỆN BIỂN BÁO GIAO THÔNG

## 1. Phân tích bài toán cơ bản (Chương trình nhận diện biển báo giao thông)

Chương trình bao gồm các giao diện để người dùng truy cập bằng tài khoản cá nhận để sử dụng các tính năng như: quét ảnh, tải ảnh để nhận diện, hay tra cứu thông tin về các biển báo giao thông.

+ Quét ảnh: sử dụng thư viện OpenCV để mở camera, rồi sử dụng chức năng chụp ảnh. Sau đó, dùng thư viện numpy chuyển ảnh thành ma trận ba chiều để xử lý.

+ Tải ảnh: Ứng dụng truy cập vào kho dữ liệu ảnh có sẵn trên máy của user để tải ảnh lên cho hệ thống xử lý thông qua thư viện os, kết hợp thư viện Image của kivy để hiện thị ảnh.

+ Tra cứu thông tin: Ứng dụng cung cấp những giao diện bắt mắt, dễ tiếp cận, thuận tiện để người dùng sử dụng. Tất cả đều được xây dựng bằng các lớp thuộc thư viện Kivy, KivyMD.

Ngoài những tính năng chính kể trên, chương trình nhận diện biển báo giao thông của chúng em còn đem đến những trải nghiệm thích thú cho người dùng qua những tính năng phụ như: giao diện dễ nhìn, hiển thị thông tin tài khoản với avatar cute, nâng cấp tính năng, lịch sử dùng,… Được viết bằng các lớp cơ bản của Python và Kivy nói chung, truy cập vào database bằng MySQL thông qua Xampp(để truy cập vào localhost) để tải những lưu trữ vào giao diện.

## 2. Phân tích bài toán ứng dụng AI (Nhận diện hình ảnh)

Về bài toán nhận diện hình ảnh, chúng em sử dụng mô hình CNN(Convolutional Neural Network).

Cách sử dụng CNN vào chương trình huấn luyện mô hình nhận diện biển báo giao thông:

Bước 1: Set up

+ Chuẩn bị dataset: các biển báo giao thông thuộc Bộ giao thông vận tải Việt Nam.

+ Chuẩn bị nhãn(label): các tên của biển báo được quy định dựa trên dataset đã chuẩn bị.

Bước 2: Thực hiện

+ Chuyển hóa hình ảnh thành ma trận, kèm theo đó sử dụng thuật toán convolutional layer để tạo ra một ma trận ba chiều(convoled feature).

+ Sử dụng thuật toán Kernel relu, padding: tối ưu hóa dữ liệu đầu vào. Kernel relu làm cho kết quả không âm, còn padding thì làm cho những pixel ở viền ma trận được tính toán nhiều hơn.

+ Max Pooling 2D: tối ưu về kích thước của convoled feature, tăng tốc độ xử lý (giảm dữ liệu một nửa).

+ Drop out: chuyển tham số bằng đơn vị phần trăm để giảm số lượng pixel của convoled feature.

Bước 3: Lặp lại bước 2 bốn lần.

Bước 4: Fully connection (Dense: mạng chằn chịt): Trải dài ma trận được tạo NxMx3 thành ma trận một chiều (N\*M)x1x3. Sử dụng thuật toán trong keras tạo ra 2 ma trận với ma trận thứ 2 có kích thước là 256x1x3, và ma trận thứ 3 là 43x1x3 (bằng với số label đã được chuẩn bị ở bước 1), có xảy ra mất mát. Các ma trận một chiều này được liên kết 1-N (phần tử ma trận phía trước liên kết với N phần tử của ma trận phía sau) theo công thức:

tổng số lượng liên kết ma trận đầu = (N\*M) \* (256) + (N\*M)

tổng số lượng liên kết ma trận thứ 2 = (256) \* (43) + (256) = 11264

Bước 5: Lưu mô hình.

# CHƯƠNG 3: CÀI ĐẶT, DEMO CHƯƠNG TRÌNH

## 1. Cài đặt

Môi trường thực hiện train và các bước thực hiện:

Môi trường: PyCharm, Visual Studio Code

Các bước train:

* Import các thư viện cần thiết và tải dữ liệu:

*+ import numpy as np*: Import thư viện numpy để làm việc với mảng nhiều chiều.

*+ import matplotlib.pyplot as plt*: Import thư viện matplotlib để trực quan hóa dữ liệu.

*+ from PIL import Image*: Import thư viện PIL để làm việc với hình ảnh.

*+ from sklearn.model\_selection import train\_test\_split*: Import hàm train\_test\_split từ thư viện sklearn để chia dữ liệu thành tập huấn luyện và tập kiểm tra.

*+ from keras.utils import to\_categorical*: Import hàm to\_categorical từ thư viện keras.utils để chuyển đổi nhãn thành dạng one-hot encoding.

*+ from keras.models import Sequential, load\_model*: Import lớp Sequential và hàm load\_model từ thư viện keras.models để xây dựng và tải mô hình.

*+ from keras.layers import Conv2D, MaxPool2D, Dense, Flatten, Dropout*: Import các lớp Conv2D, MaxPool2D, Dense, Flatten, Dropout từ thư viện keras.layers để xây dựng kiến trúc mạng nơ-ron.

* Chuẩn bị dữ liệu:

+ Số lượng ảnh train: 31367 ảnh kích thước 30x30x3

+ Số lượng ảnh test: 7842 ảnh kích thước 30x30x3

+ Số lượng nhãn biển báo: 43

+ Các nhãn gồm: Giới hạn tốc độ (20km/h), giới hạn tốc độ (30km/h), giới hạn tốc độ (50km/h), giới hạn tốc độ (60km/h), giới hạn tốc độ (70km/h), giới hạn tốc độ (80km/h), kết thúc giới hạn tốc độ (80km/h), giới hạn tốc độ (100km/h), giới hạn tốc độ (120km/h), cấm vượt, cấm vượt xe trọng tải trên 3.5 tấn, ưu tiên ở giao lộ, đường ưu tiên, nhường đường, dừng lại, cấm xe, cấm xe trọng tải trên 3.5 tấn, cấm đi, chú ý chung, nguy hiểm ngoặc vòng bên trái, nguy hiểm ngoặc vòng bên phải, nguy hiểm ngoặc vòng kép, đường xấu, đường trơn trượt, đường hẹp bên phải, công trường, đèn giao thông, người đi bộ, đường gặp người đi bộ, đường gặp xe đạp, cẩn thận băng/giá lạnh, gặp động vật hoang dã, kết thúc giới hạn tốc độ và cấm vượt, rẽ phải phía trước, rẽ trái phía trước, chỉ được đi thẳng, đi thẳng hoặc rẽ phải, đi thẳng hoặc rẽ trái, luôn đi bên phải, luôn đi bên trái, vòng xuyến bắt buộc, kết thúc cấm vượt, kết thúc cấm vượt xe trọng tải trên 3.5 tấn.

* Chuyển đổi dữ liệu thành mảng numpy:

+ Chuyển đổi data và labels từ danh sách thành mảng numpy bằng cách sử dụng np.array().

* Chia dữ liệu thành tập huấn luyện và tập kiểm tra:

+ Sử dụng hàm train\_test\_split() để chia data và labels thành X\_train, X\_test, y\_train, y\_test với tỷ lệ tập kiểm tra là 0.2 (20%).

+ Kết quả được lưu vào các biến tương ứng.

* Chuyển đổi nhãn thành dạng one-hot encoding:

+ Sử dụng hàm to\_categorical() từ thư viện keras.utils để chuyển đổi y\_train và y\_test thành dạng one-hot encoding với số lượng lớp là 43.

* Xây dựng mô hình:

+ Tạo một đối tượng mô hình Sequential bằng cách sử dụng model = Sequential().

+ Thêm các lớp Conv2D, MaxPool2D, Dropout, Flatten, và Dense vào mô hình theo kiến trúc đã được định nghĩa.

+ Lớp cuối cùng sử dụng softmax activation để đưa ra dự đoán cho 43 lớp.

* Biên dịch mô hình:

+ Sử dụng compile() để biên dịch mô hình với hàm mất mát là categorical\_crossentropy và tối ưu hóa bằng adam optimizer.

+ Đánh giá mô hình dựa trên chỉ số độ chính xác (accuracy).

* Huấn luyện mô hình:

+ Sử dụng fit() để huấn luyện mô hìnhvới dữ liệu huấn luyện X\_train và nhãn y\_train trong epochs epochs.

+ Sử dụng kích thước batch là 32 và sử dụng tập kiểm tra X\_test và y\_test để đánh giá mô hình sau mỗi epoch.

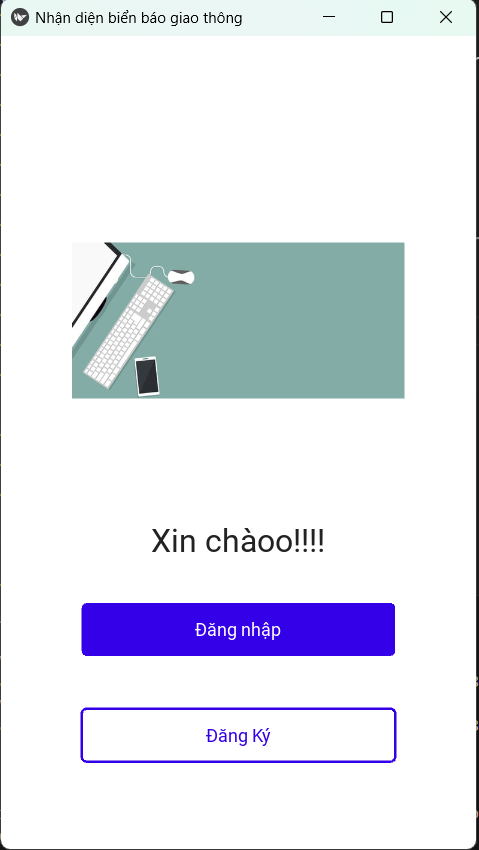
* Lưu mô hình:

+ Sử dụng save() để lưu mô hình vào tệp "my\_model.keras".

* Trực quan hóa kết quả:

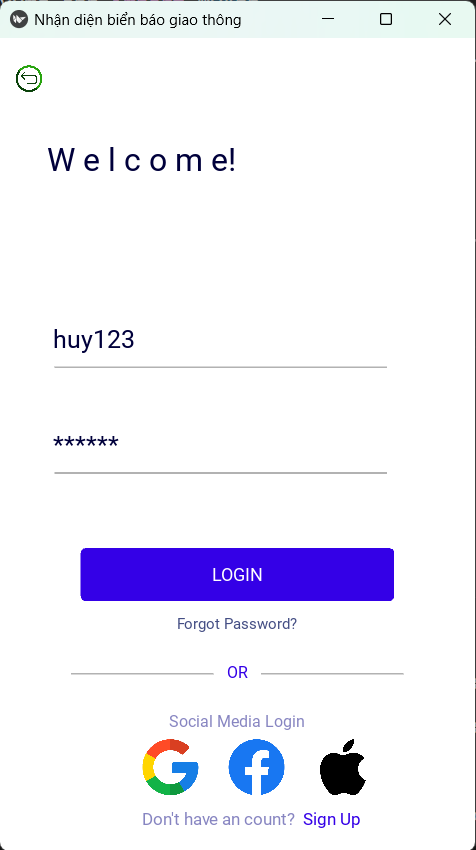
+ Sử dụng matplotlib để vẽ đồ thị cho độ chính xác (accuracy) và hàm mất mát (loss) trên tập huấn luyện và tập kiểm tra.

## 2. Demo chương trình

****

**Hình 1. Main Login**

Khi mở app sẽ hiện ra trang main\_login đầu tiên sau đó chọn 1 trong 2: chọn đăng nhập nếu đã có tài khoản, chọn đăng kí nếu chưa có tài khoản đăng nhập.

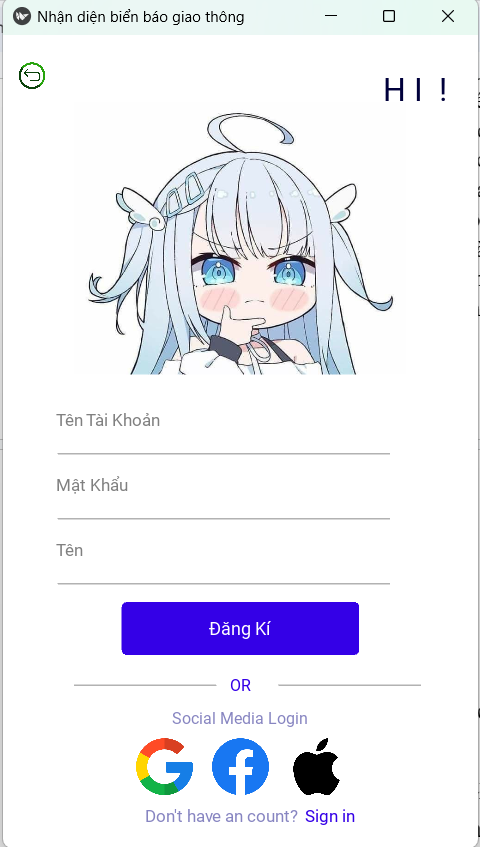
****

**Hình 2. Login**

Trang đăng nhập:

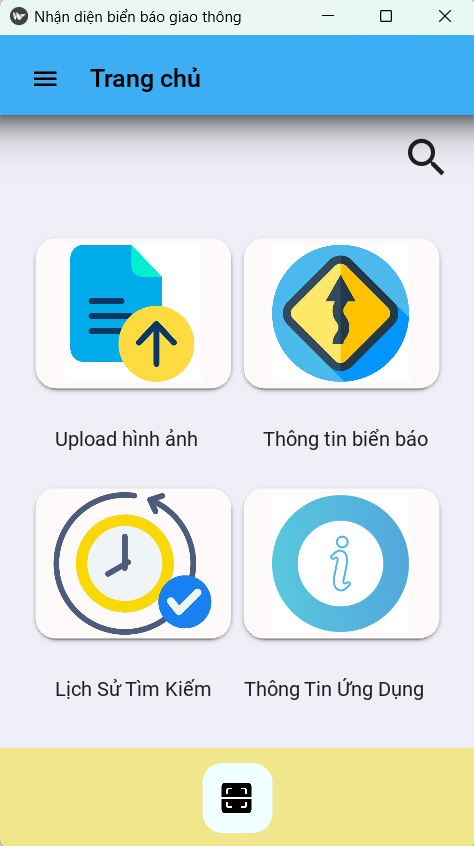
+ Sau khi nhập đúng tài khoản và tên đăng nhập sẽ được chuyển vào frame main của app.

+ Có 2 cách đăng nhập: đăng nhập bằng tài khoản đăng kí hoặc bằng tài khoản google.



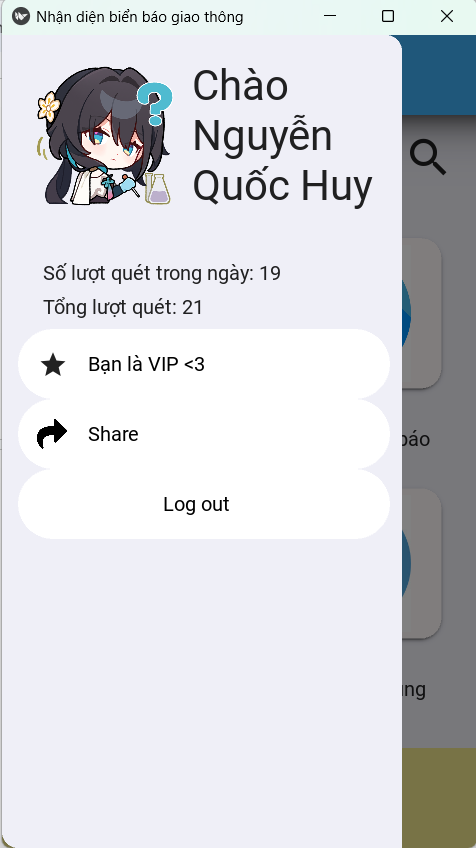
**Hình 3. Sign in**

Điền các thông tin cơ bản của tài khoản vào form đăng kí nếu sau khi kiểm tra thông tin đăng kí hợp lệ thì sẽ được đăng kí.

****

**Hình 4. Trang chủ**

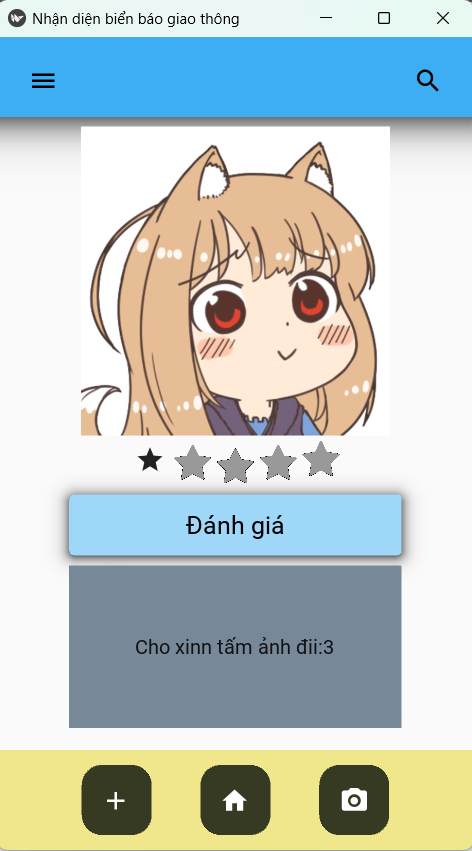
Đây là trang chủ của app hiển thì các chức năng chính: **up load hình để nhận diện**, **list các loại biển báo**, **lịch sử quét**, **thông tin của app**, **quét,** **thông tin cá nhân** nằm bên tay trái phía trên của frame.

****

**Hình 5. Thông tin cá nhân**

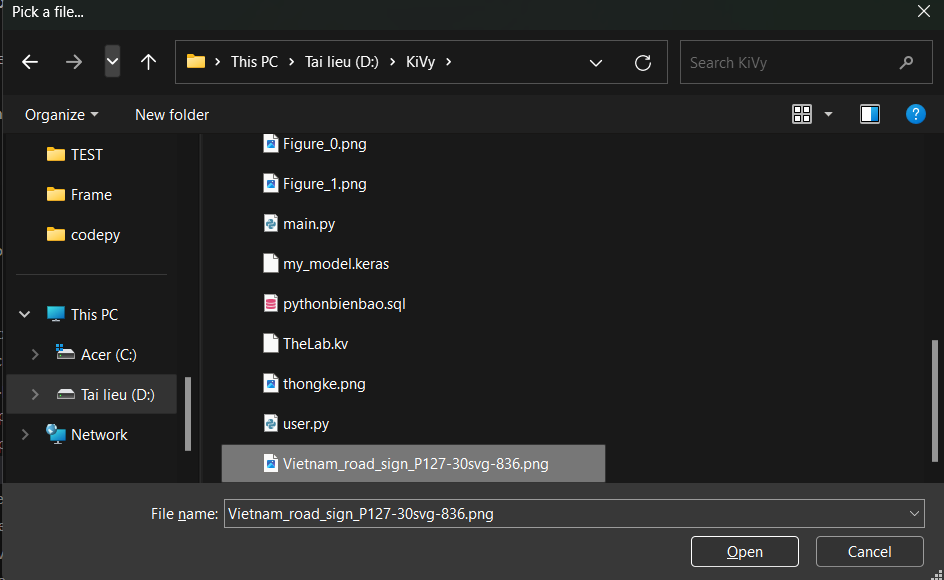
Trang này thề hiện các thông tin cá nhân của user như tên, số lượt quét trong ngày, tổng lượt quét, có phải là khách hàng VIP không; Log Out.

Sau khi nhấn vào nút **Log out** sẽ trở về lại trang Main\_Login (Hình 1).

****

**Hình 6.1 Upload hình**

Khi nhấn Button hình dấu cộng sẽ hiện lên thư mục để chọn ảnh:

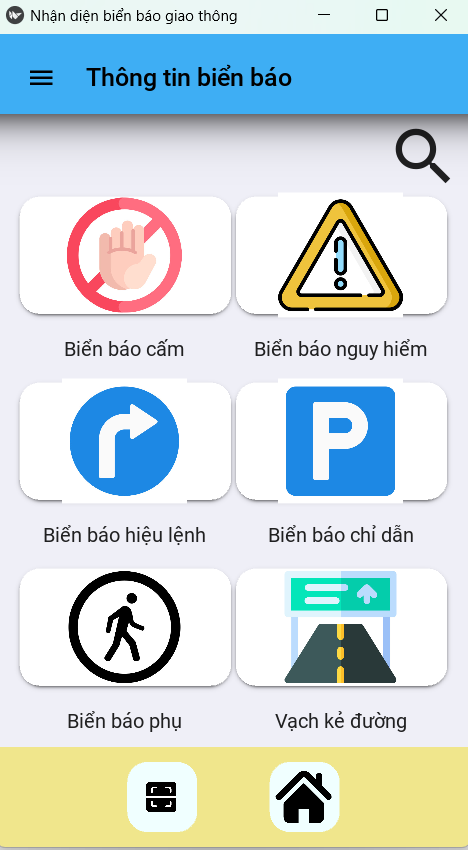


Sau khi chọn ảnh thành công thì sẽ nhận diện file vừa chọn và hiển thị kết quả ra màn hình.

****

**Hình 6.2 Sau khi upload ảnh và nhận diện**

Từ Trang chủ khi click vào Thông tin biển báo:

****

**Hình 7. Thông tin biển báo**

Có 6 nút tương ứng với với các loại biển báo giao thông của Việt Nam: Biển báo cấm, biển báo nguy hiểm, biển báo hiệu lệnh, biển báo chỉ dẫn, biển báo phụ biển báo vạch kẻ đường:





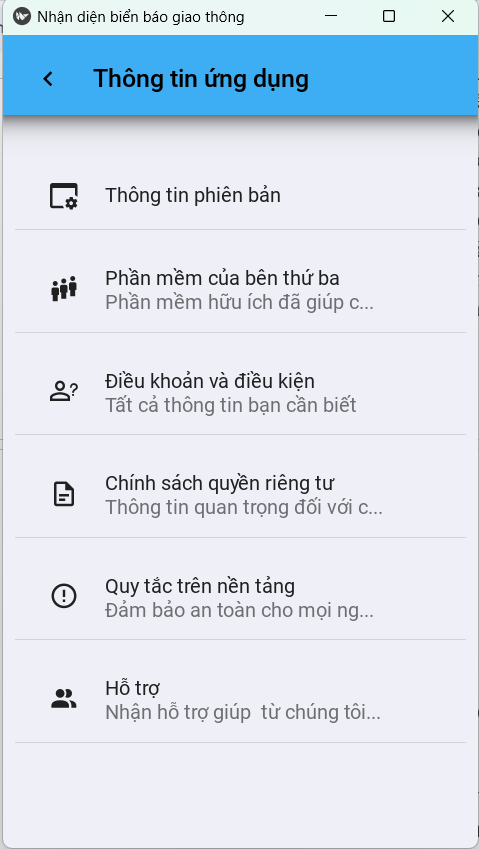
Sau khi click vào Lịch sử tìm kiếm sẽ hiện ra lịch sử quét của tài khoản được hiển thị ra màn hình:

****

**Hình 8. Lịch sử quét**

Ở đây hiển thị hình ảnh đã quét và hiển thị kết quả, thời gian quét.

Sau khi click vào Thông tin ứng dụng sẽ hiện lên một số thông tin cơ bản của App:

****

**Hình 9. Thông tin ứng dụng**

# KẾT LUẬN

## ĐÁNH GIÁ NHỮNG NỘI DUNG ĐÃ THỰC HIỆN

**-**Thiết kế app kivy có giao diện ưa nhìn, dễ dàng sử dụng, trực quan

**-** Nhận diện được các biển báo đã có trong dữ liệu(43 biển)

**-** Độ chính xác hơn 95%

**-**Thống kê người dùng

**-** Đăng nhập được bằng tài khoản gmail

## ĐÁNH GIÁ NHỮNG NỘI DUNG CHƯA THỰC HIỆN

**-** Số lượng thông tin biển báo trong dataset có giới hạn

**-** Giao diện admin còn sơ sài

**-** Độ chính xác chưa tuyệt đối

-Chưa liên kết được với các tài khoản mạng xã hội khác

-Chưa xuất được file apk

## HƯỚNG PHÁT TRIỂN

* + Hoàn thiện chương trình:

+ Liên kết big data để ứng dụng xử lý chính xác hơn.

+ Fix lại giao diện để trông chuyên nghiệp hơn.

+ Update thêm những tính năng nhỏ nhưng hỗ trợ cho người dùng thuận tiện hơn như: đăng nhập bằng tài khoản mạng xã hội, up ảnh, chia sẻ,…

* + Thu lại lợi nhuận từ ứng dụng:

+ Build thành file apk: tải code lên ggcolab, sử dụng các câu lệnh chuyên biệt để build. Sau đó, có thể tải và sử dụng trực tiếp trên thiết bị di động.

+ Tiến hành chạy quảng cáo, hoàn thiện tính năng nạp để thu lợi nhuận.

+ Tích hợp vào xe tự động lái

+Thêm tính năng hỗ trợ