



TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA CÔNG NGHỆ THÔNG TIN



BÁO CÁO PBL5
DỰ ÁN KỸ THUẬT MÁY TÍNH

HỆ THỐNG PHÁT HIỆN BIỂN BÁO GIAO
THÔNG VÀ ĐƯA RA CẢNH BÁO

Giảng viên hướng dẫn: TS.Huỳnh Hữu Hưng

SINH VIÊN THỰC HIỆN	MÃ HỌC PHẦN	MSSV
Lê Hữu Minh Vũ	22.13B	102210336
Nguyễn Phạm Phúc Tân	22.13B	102210326
Trương Nguyễn Hoàng Phát	22.13B	102210322
Dương Võ Hoàng Hùng	22.13B	102210315

LỜI MỞ ĐẦU

Trong bối cảnh giao thông ngày càng phát triển và phức tạp, việc đảm bảo an toàn giao thông trở thành ưu tiên hàng đầu. Biển báo giao thông đóng vai trò quan trọng trong việc hướng dẫn, cảnh báo và điều phối các phương tiện, giúp giảm thiểu tai nạn và ùn tắc. Tuy nhiên, việc nhận diện và xử lý thông tin biển báo thủ công gặp nhiều hạn chế về độ chính xác, thời gian và chi phí.

Nhờ sự phát triển nhanh chóng của trí tuệ nhân tạo và học sâu, đặc biệt là các mô hình phát hiện đối tượng như YOLO, việc xây dựng hệ thống phát hiện biển báo giao thông tự động trở nên khả thi và hiệu quả. Hệ thống này không chỉ nâng cao độ chính xác và tốc độ nhận dạng trong nhiều điều kiện khác nhau mà còn góp phần phát triển các ứng dụng giao thông thông minh như hỗ trợ lái xe tự động và quản lý giao thông.

Báo cáo này nhằm nghiên cứu và triển khai hệ thống phát hiện biển báo giao thông sử dụng mô hình YOLO, từ đó đánh giá hiệu quả và khả năng ứng dụng của hệ thống trong việc nhận diện biển báo và đưa ra cảnh báo kịp thời cho người tham gia giao thông, góp phần nâng cao an toàn và ý thức lái xe tại Việt Nam, đồng thời thúc đẩy sự phát triển của công nghệ giao thông thông minh trong nước.

BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên thực hiện	Nhiệm vụ	Tự đánh giá theo 3 mức (Đã hoàn thành/Chưa hoàn thành/Không triển khai)
Lê Hữu Minh Vũ	<ul style="list-style-type: none"> - Huấn luyện mô hình EfficienDet - Viết báo cáo 	Đã hoàn thành
Nguyễn Phạm Phúc Tân	<ul style="list-style-type: none"> - Thiết lập phần cứng - Phát triển ứng dụng - Viết báo cáo - Gán nhãn dữ liệu 	Đã hoàn thành
Trương Nguyễn Hoàng Phát	<ul style="list-style-type: none"> - Thu thập dữ liệu - Tăng cường dữ liệu - Huấn luyện mô hình YOLO V5 - Viết báo cáo 	Đã hoàn thành
Dương Võ Hoàng Hùng	<ul style="list-style-type: none"> - Thiết lập phần cứng - Gán nhãn dữ liệu - Phát triển ứng dụng - Viết báo cáo 	Đã hoàn thành

MỤC LỤC

1. Giới thiệu.....	7
1.1. Vấn đề cần giải quyết.....	7
1.2. Các giải pháp hiện có.....	7
1.3. Giải pháp đề xuất.....	7
2. Giải pháp.....	7
2.1. Giải pháp phần cứng và truyền thông.....	7
2.2.1 Cấu hình phần cứng.....	8
2.2.2 Sơ đồ khối phần cứng và giao tiếp.....	8
2.2.4 Sơ đồ quy định xử lý Camera.....	10
2.2.5 Danh sách linh kiện, thông số kỹ thuật và chi phí.....	11
2.2. Giải pháp AI/KHDL.....	12
2.2.1 Mô hình AI sử dụng.....	12
2.2.2 Quy trình xử lý dữ liệu và huấn luyện.....	13
2.2.4 Lý do chọn giải pháp.....	23
2.3. Giải pháp phần mềm.....	24
3. Kết quả.....	25
3.1. Ngôn ngữ và thư viện sử dụng.....	25
3.2 Cấu trúc xử lý phần mềm.....	26
4. Xử lý và hiển thị kết quả:.....	27
4. Kết luận.....	30
4.1. Khó khăn.....	30
4.2. Hướng phát triển.....	30
5. Tài liệu tham khảo.....	31

DANH SÁCH HÌNH ẢNH

Hình 1. Sơ đồ phân cứng và giao tiếp

Hình 2. Quy trình xử lý – sơ đồ giải thuật

Hình 3. Ảnh ở các class.

Hình 4. Số lượng ảnh ở mỗi class.

Hình 5. Số lượng ảnh ở tập train ở mỗi class trước và sau khi tăng cường mỗi.

Hình 6. Ảnh sau khi áp dụng các phương pháp agumentation ở mỗi class.

Hình 7. Code train model YOLOv5.

Hình 8. Model YOLOv5 sau khi train lưu ở folder weights

Hình 9. Precision, Recall và mAP theo từng biến báo.

Hình 10. Biểu đồ Loss và Precision-Recall trên tập huấn luyện và đánh giá.

Hình 11. Ma trận nhầm lẫn.

Hình 12. Quy trình xử lý mô hình AI

Hình 13. Sơ đồ khởi quy trình xử lý GUI

Hình 14. Giao diện chính hệ điều Raspberry 3

Hình 15. Giao diện điều chỉnh file client chạy nền mỗi khi khởi động Raspberry 3

Hình 16. Giao diện chính của phần mềm

Hình 17. Dự đoán các biến báo với chức năng Real Time

Hình 18. Dự đoán các biến báo với chức năng Real Time

Hình 19. Dự đoán các biến báo với chức năng Real Time

Hình 20. Dự đoán các biến báo với chức năng Load Image

Hình 21. Dự đoán các biến báo với chức năng Load Video

DANH SÁCH CÁC BẢNG

Bảng 1. Bảng liệt kê linh kiện và thông số kỹ thuật.

Bảng 2. Bảng kê chi phí linh kiện.

1. Giới thiệu

1.1. Vấn đề cần giải quyết

Trong bối cảnh giao thông hiện đại, vấn đề nhận diện biển báo giao thông trở nên ngày càng quan trọng, đặc biệt là trong các hệ thống hỗ trợ lái xe tự động và các công nghệ giao thông thông minh. Hiện tại, nhiều hệ thống giao thông và phương tiện chưa có khả năng nhận diện biển báo giao thông trong thời gian thực, dẫn đến tiềm ẩn nguy cơ gây tai nạn. Vì vậy, hệ thống nhận diện biển báo giao thông và đưa ra cảnh báo là một giải pháp cần thiết, nhằm giúp nâng cao an toàn cho người tham gia giao thông, đặc biệt là lái xe.

1.2. Các giải pháp hiện có

Trên thế giới, nhiều nghiên cứu và hệ thống đã được triển khai để nhận diện biển báo giao thông bằng cách sử dụng các mô hình học sâu (deep learning) và các phương pháp xử lý hình ảnh. Các mô hình như YOLO (You Only Look Once), Faster R-CNN và SSD (Single Shot Multibox Detector) đã chứng minh hiệu quả trong việc phát hiện đối tượng, bao gồm cả biển báo giao thông. Tuy nhiên, vẫn còn nhiều thách thức về độ chính xác, tốc độ xử lý và khả năng nhận diện trong các điều kiện môi trường khác nhau như ánh sáng yếu, thời tiết xấu.

1.3. Giải pháp đề xuất

Hệ thống mà chúng tôi đề xuất sử dụng mô hình YOLO v5, một phiên bản cải tiến của YOLO, để phát hiện và nhận diện biển báo giao thông trong thời gian thực. Hệ thống sẽ kết hợp với một cảnh báo âm thanh, giúp người lái xe nhận được thông tin cảnh báo ngay lập tức khi phát hiện biển báo giao thông. Hệ thống này sẽ sử dụng camera để thu thập hình ảnh, sau đó mô hình AI sẽ xử lý và nhận diện biển báo, gửi tín hiệu cảnh báo..

2. Giải pháp

2.1. Giải pháp phần cứng và truyền thông

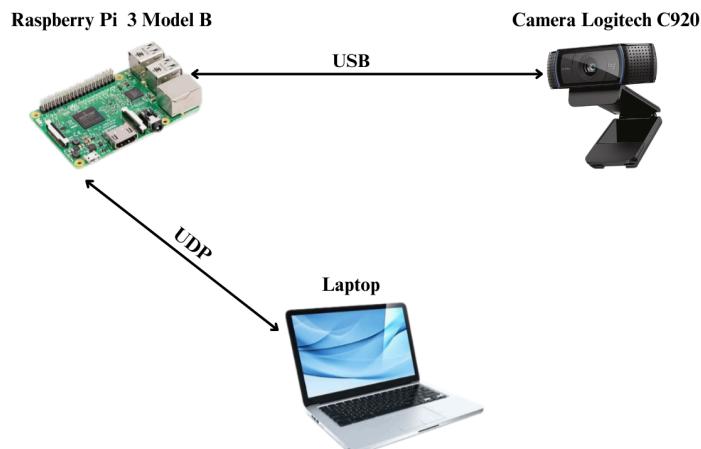
Hệ thống sử dụng một camera HD được lắp đặt trên xe để ghi lại hình ảnh liên tục trong quá trình di chuyển. Dữ liệu hình ảnh được truyền qua cổng USB hoặc Wifi đến một đơn vị xử lý (Máy tính tích hợp). Mô hình YOLO v5 sẽ được cài đặt trên hệ thống xử lý để nhận diện biển báo trong thời gian thực.

2.2.1 Cấu hình phần cứng

Hệ thống phát hiện biển báo giao thông được thiết kế nhằm đảm bảo tính đơn giản, chi phí thấp và khả năng triển khai thực tế cao. Các thành phần phần cứng chính bao gồm:

- Raspberry Pi 3: Đóng vai trò là trung tâm xử lý, thực thi mô hình học sâu và điều khiển các thiết bị ngoại vi. Raspberry Pi là nền tảng phổ biến trong các ứng dụng nhúng nhờ kích thước nhỏ gọn, tiêu thụ điện năng thấp và khả năng xử lý đủ mạnh cho các mô hình AI nhẹ [1].
- Camera C922 logitech USB: Thu hình ảnh đầu vào phục vụ quá trình nhận diện theo thời gian thực.
- Laptop: Hiển thị GUI, là server nhận thông tin từ Raspberry, phát âm thanh cảnh báo

2.2.2 Sơ đồ khái niệm phần cứng và giao tiếp



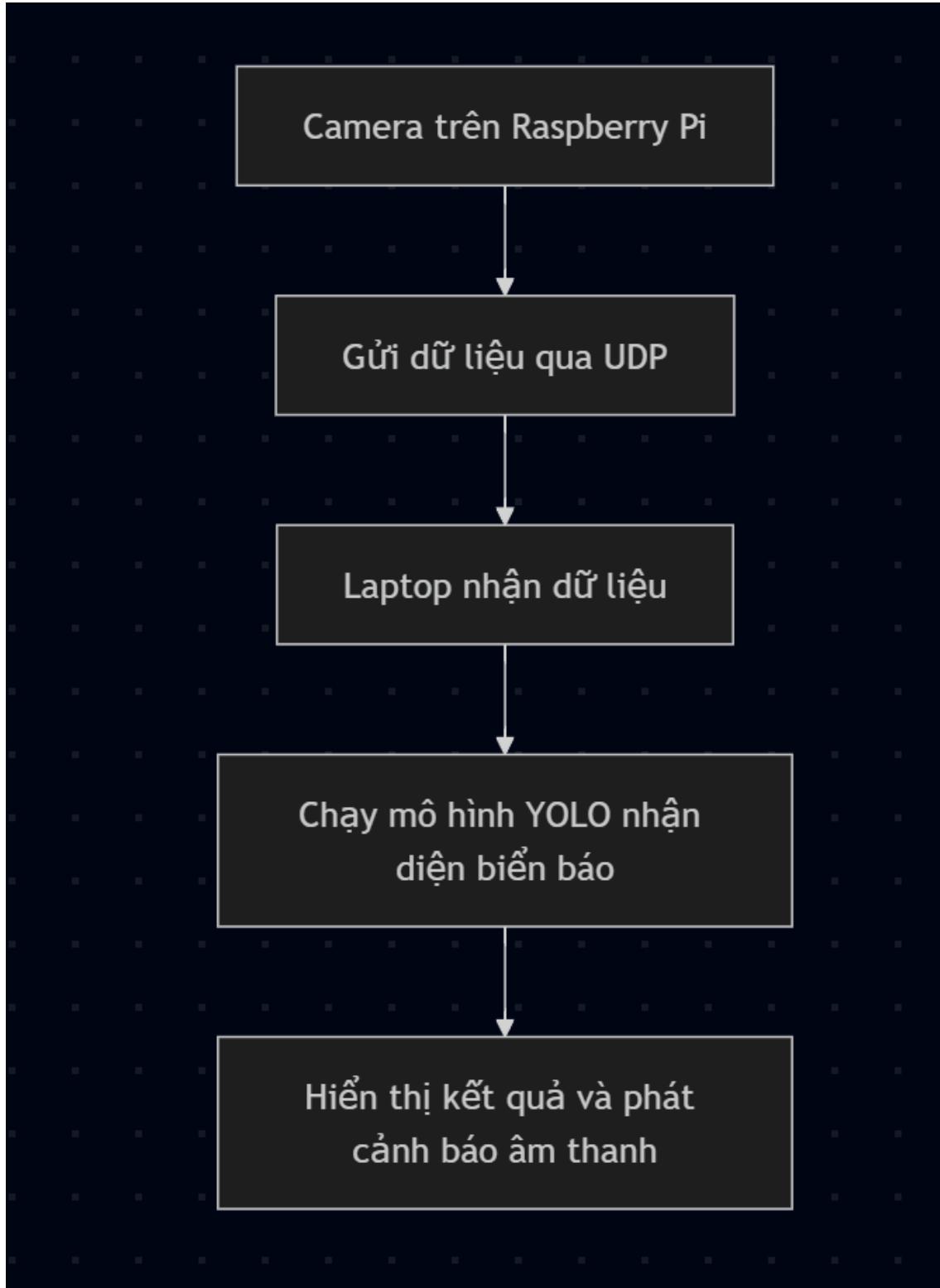
Hình 1. Sơ đồ phần cứng và giao tiếp

2.2.3 Nguyên lý hoạt động:

- Laptop đóng vai trò là một server, Raspberry pi 3 đóng vai trò là client gửi thông tin sang cho server thông qua phương thức UDP (Wifi hoặc cổng ethernet)

- Raspberry Pi 3 Model B: Đọc từng khung hình (FPS) từ camera sau đó gửi về laptop bằng giao thức UDP.
- Camera Logitech C920 (USB): Thu hình ảnh đầu vào phục vụ quá trình nhận diện theo thời gian thực.
- Laptop thực thi đọc mô hình học sâu best.pt , sau đó hiển thị kết quả dự đoán cho GUI để hiển thị cho người dùng

2.2.4 Sơ đồ quy trình xử lý Camera



Hình 2. Sơ đồ quy trình xử lý Camera

2.2.5 Danh sách linh kiện, thông số kỹ thuật và chi phí

2.2.5.1 Bảng liệt kê linh kiện và thông số kỹ thuật

Linh kiện	Thông số kỹ thuật	Nguyên lý hoạt động
 Raspberry Pi 3 Model B	CPU: Cortex-A53 1.2GHz, 64-bit RAM: 1GB Wi-Fi: 802.11 b/g/n Ethernet: 10/100Mbps Nguồn: 5V/2.5A Đầu ra: GPIO, HDMI, USB	Thiết bị nhúng nhỏ gọn, chạy hệ điều hành Raspbian, điều khiển các thiết bị ngoại vi qua USB hoặc TCP.
 Camera Logitech C920	Độ phân giải 1080p, 30FPS, giao tiếp qua USB	Được điều khiển bởi Raspberry để ghi hình và truyền ảnh trực tiếp về laptop để xử lý.
 Thẻ nhớ MicroSD 16GB	Loại Class 10, dung lượng ≥ 16GB	Lưu hệ điều hành, mã nguồn và tạm lưu kết quả nhận diện.

 Laptop	Lenovo Legion 5 2023: CPU AMD Ryzen 7/9 RAM: 16-32GB Đầu vào: 220V Đầu ra: USB-C/USB-A, HDMI	Cung cấp điện năng ổn định cho Raspberry Pi và các thiết bị ngoại vi đi kèm. Lưu trữ mã nguồn và thực thi mô hình học sâu
--	---	--

Bảng 1. Bảng liệt kê linh kiện và thông số kỹ thuật.

2.2.5.2 Bảng kê chi phí linh kiện

STT	Tên linh kiện	Số lượng	Đơn giá (VNĐ)	Thành tiền (VNĐ)
1	Raspberry Pi 3 Model B (1GB RAM)	1	1.800.000	1.800.000
2	Camera Logitech C920 (720p/1080p)	1	1.700.000	1.700.000
5	Thẻ nhớ MicroSD 16GB	1	100.000	100.000
Tổng cộng				3.600.000

Bảng 2. Bảng kê chi phí linh kiện.

2.2. Giải pháp AI/KHDL

2.2.1 Mô hình AI sử dụng

Trong đồ án này, nhóm sử dụng mô hình YOLO v5 (You Only Look Once version 5) – một mạng nơ-ron tích chập (CNN) nổi bật trong bài toán nhận diện đối tượng theo thời gian thực. YOLO v5 được phát triển bởi Ultralytics, cải tiến từ các phiên bản trước như YOLO v3 và YOLO v4, sử dụng nền tảng PyTorch và dễ dàng triển khai trên thiết bị nhúng như Raspberry Pi. Mô hình này hoạt động bằng cách chia ảnh đầu vào thành nhiều lối và đưa ra dự đoán trực tiếp các bounding box và nhãn lớp trong một lần suy luận duy nhất.

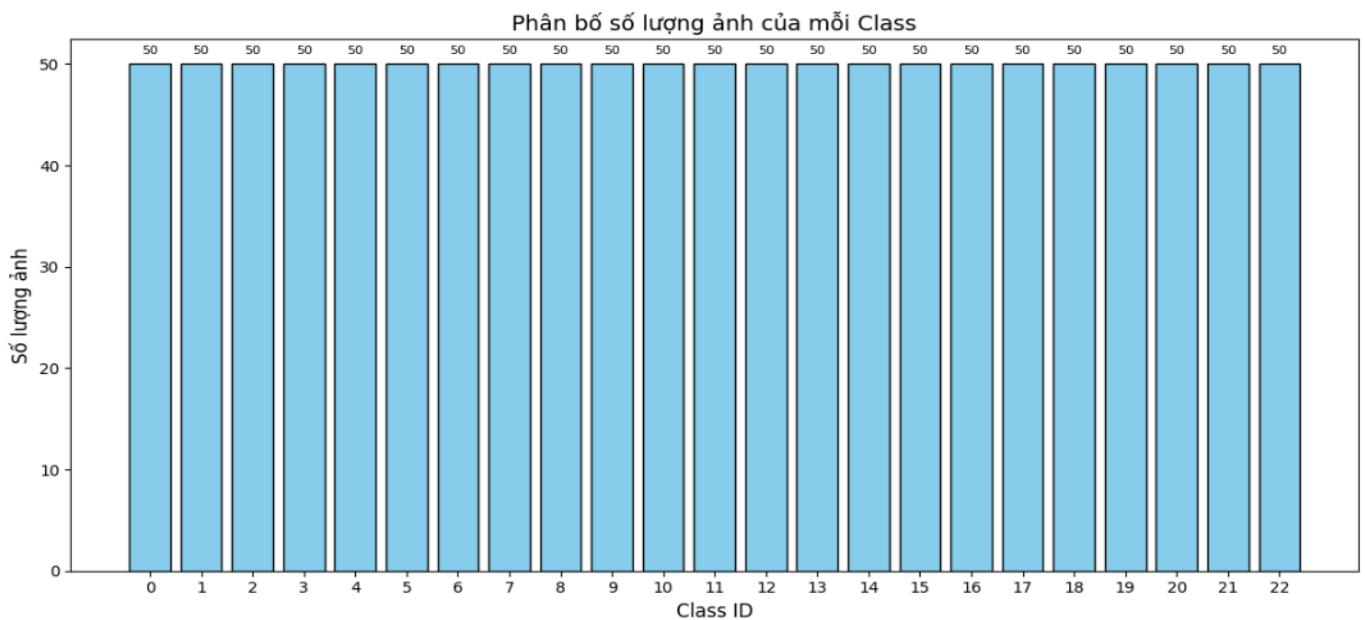
2.2.2 Quy trình xử lý dữ liệu và huấn luyện

2.2.2.1. Thu thập dữ liệu:

- Sử dụng điện thoại để chụp hình các biển báo giao thông thực tế tại địa phương.



Hình 3. Ảnh ở các class.



Hình 4. Số lượng ảnh ở mỗi class.

- Biểu đồ thể hiện số lượng biển báo có trong dataset. Trục tung biểu thị số lượng ảnh và trục hoành biểu thị các loại biển báo theo class id. Cụ thể có 23 class:

- 0 - Cam re trai
- 1 - Giao nhau voi duong uu tien
- 2 - Cam do xe ngay le
- 3 - Cam do xe ngay chan
- 4 - Duong giao nhau
- 5 - Cam di nguoc chieu
- 6 - Nguoi giao nhau theo vong xuyen
- 7 - Giao nhau voi duong khong uu tien
- 8 - Cam do xe
- 9 - Toc do toi da 40 km/h
- 10 - Tre em qua duong
- 11 - Cam re phai
- 12 - Cam dung xe va do xe
- 13 - Cam bam coi
- 14 - Toc do toi da 80 km/h
- 15 - Toc do toi da 60 km/h
- 16 - Cam re trai
- 17 - Cho ngoac nguy hiem phia ban phai
- 18 - Huong di thang phai theo
- 19 - Duong di bo cat ngang
- 20 - Cam quay dau
- 21 - Cho ngoac nguy hiem phia ben trai
- 22 - Cam xe may 2 banh

2.2.2.2. Gán nhãn dữ liệu:

- Dữ liệu ảnh được tải lên Roboflow, nền tảng hỗ trợ gán nhãn, resize, tăng cường dữ liệu (augmentation), và xuất dữ liệu về đúng định dạng YOLO v5.

2.2.2.3. Phân chia tập dữ liệu:

- Tiến hành chia tập dữ liệu thành hai phần: train và validation với tỷ lệ lần lượt là 80-20
- Tập train (80%): Được sử dụng để huấn luyện mô hình, bao gồm các mẫu đã qua tăng cường dữ liệu để đảm bảo mô hình có thể học được các đặc trưng phong phú và đa dạng
- Tập validation (20%): Được sử dụng để đánh giá hiệu năng của mô hình trong quá trình huấn luyện, giúp điều chỉnh siêu tham số và phát hiện hiện tượng overfitting

2.2.2.4. Tăng cường dữ liệu:

Áp dụng 2 phương pháp augmentation : Rotation và Brightness

- Rotation (xoay ảnh):
 - + Xoay ảnh ngẫu nhiên trong khoảng góc từ -5 đến +5.
ROTATION_RANGE = (-5, 5)
 - + Đồng thời điều chỉnh lại tọa độ bounding box sao cho đúng vị trí mới sau khi xoay ảnh.
- Brightness (điều chỉnh độ sáng)
 - + Thay đổi độ sáng của ảnh theo hệ số ngẫu nhiên trong khoảng từ 0.9 đến 1.1.
BRIGHTNESS_RANGE = (0.9, 1.1)
 - + Độ sáng được thay đổi bằng cách nhân giá trị pixel với hệ số này rồi cắt xén giá trị trong khoảng 0-255.

2.2.2.5 Cách áp dụng augmentation:

- Đếm số lượng mẫu cho từng class trong dataset.
- Nếu số lượng mẫu của class nào chưa đủ TARGET COUNT (200 mẫu), thì thực hiện augmentation cho ảnh thuộc class đó.
- Ảnh được chọn ngẫu nhiên từ nhóm ảnh có 1 class hoặc nhiều class.
- Ảnh được tạo ra với augmentation ngẫu nhiên: xoay hoặc điều chỉnh sáng.
- Ảnh và file nhãn tương ứng được lưu vào thư mục output.

👉 Số lượng class trước augmentation:

```
Class 0: 40  
Class 1: 40  
Class 2: 40  
Class 3: 40  
Class 4: 40  
Class 5: 40  
Class 6: 40  
Class 7: 40  
Class 8: 40  
Class 9: 40  
Class 10: 40  
Class 11: 40  
Class 12: 40  
Class 13: 40  
Class 14: 40  
Class 15: 40  
Class 16: 40  
Class 17: 40  
Class 18: 40  
Class 19: 40  
Class 20: 40  
Class 21: 40  
Class 22: 40
```

👉 Số lượng class sau augmentation:

```
Class 0: 200  
Class 1: 200  
Class 2: 200  
Class 3: 200  
Class 4: 200  
Class 5: 200  
Class 6: 200  
Class 7: 200  
Class 8: 200  
Class 9: 200  
Class 10: 200  
Class 11: 200  
Class 12: 200  
Class 13: 200  
Class 14: 200  
Class 15: 200  
Class 16: 200  
Class 17: 200  
Class 18: 200  
Class 19: 200  
Class 20: 200  
Class 21: 200  
Class 22: 200
```

Hình 5. Số lượng ảnh ở tập train ở mỗi class trước và sau khi tăng cường mỗi.



Hình 6. Ảnh sau khi áp dụng các phương pháp augmentation ở mỗi class.

2.2.2.7 Vấn đề:

Sau khi tăng cường dữ liệu xong nhóm nhận thấy sự chênh lệch tỷ lệ giữa 2 tập train và validation ở mỗi class. Điều này có thể sẽ làm giảm chất lượng đánh giá, dẫn đến kết luận sai về khả năng của model và làm model dễ bị overfit, cụ thể:

- train : 200 ảnh.

- validation : 10 ảnh.

Vì vậy nhóm giải quyết bằng cách gộp 2 tập train và validation lại và thực hiện phân chia dữ liệu thêm 1 lần nữa cho 2 tập train và validation theo tỷ lệ lần lượt 80-20.

Sau khi gộp và phân chia lại số lượng ảnh ở 2 tập ở mỗi class như sau:

- train : 170 ảnh.
- validation : 40 ảnh.

Nhìn chung, 170 train và 40 validation là tỉ lệ thường được xem là hợp lý trong nhiều bài toán học máy. Có thể huấn luyện và đánh giá mô hình vì:

- validation đủ ảnh để đánh giá model chính xác hơn,
- train vẫn có nhiều ảnh để học đặc trưng.
- Tỉ lệ này giúp tránh overfitting và có đánh giá ổn định

2.2.2.8 Huấn luyện mô hình:

Bộ dữ liệu đã xử lý và tăng cường sau đó đưa vào huấn luyện trên nền tảng Kaggle Notebook, sử dụng GPU miễn phí với môi trường PyTorch tích hợp sẵn.

```
!python train.py \
  --img 640 \
  --batch 16 \
  --epochs 90 \
  --data /kaggle/working/data.yaml \
  --cfg models/yolov5m.yaml \
  --weights yolov5m.pt \
  --name traffic_sign_yolo \
  --project traffic_sign_classification_version2 \
  --exist-ok
```

Hình 7. Code train model YOLO v5.

--img 640: Kích thước ảnh đầu vào được resize về 640x640 pixel trước khi đưa vào mạng.

--batch 16:

Kích thước batch = 16, số ảnh xử lý trong 1 bước huấn luyện.

Batch lớn giúp ổn định gradient nhưng tốn nhiều RAM.

--epochs 90:

Số vòng huấn luyện (epoch) là 90 lần toàn bộ dữ liệu được học.

--data /kaggle/working/data.yaml:

File cấu hình dữ liệu, định nghĩa đường dẫn dataset, số class và nhãn,

--cfg models/yolov5m.yaml:

File cấu hình mạng, ở đây dùng model YOLO v5 Medium (m), cấu trúc mạng trung bình phù hợp với số lượng dataset

--weights yolov5m.pt:

Trọng số pretrained được dùng để khởi tạo mạng, giúp tăng tốc huấn luyện.

2.2.2.9. Triển khai mô hình: Sau khi huấn luyện, mô hình được lưu dưới dạng .pt và chép vào Raspberry Pi để phục vụ nhận diện trực tiếp.

- ▼ traffic_sign_classification
- ▼ traffic_sign_yolo
- ▼ weights
 - best.pt
 - last.pt

Hình 8. Model YOLO v5 sau khi train lưu ở folder weights.

2.2.2.10. Kết quả sau khi train mô hình:

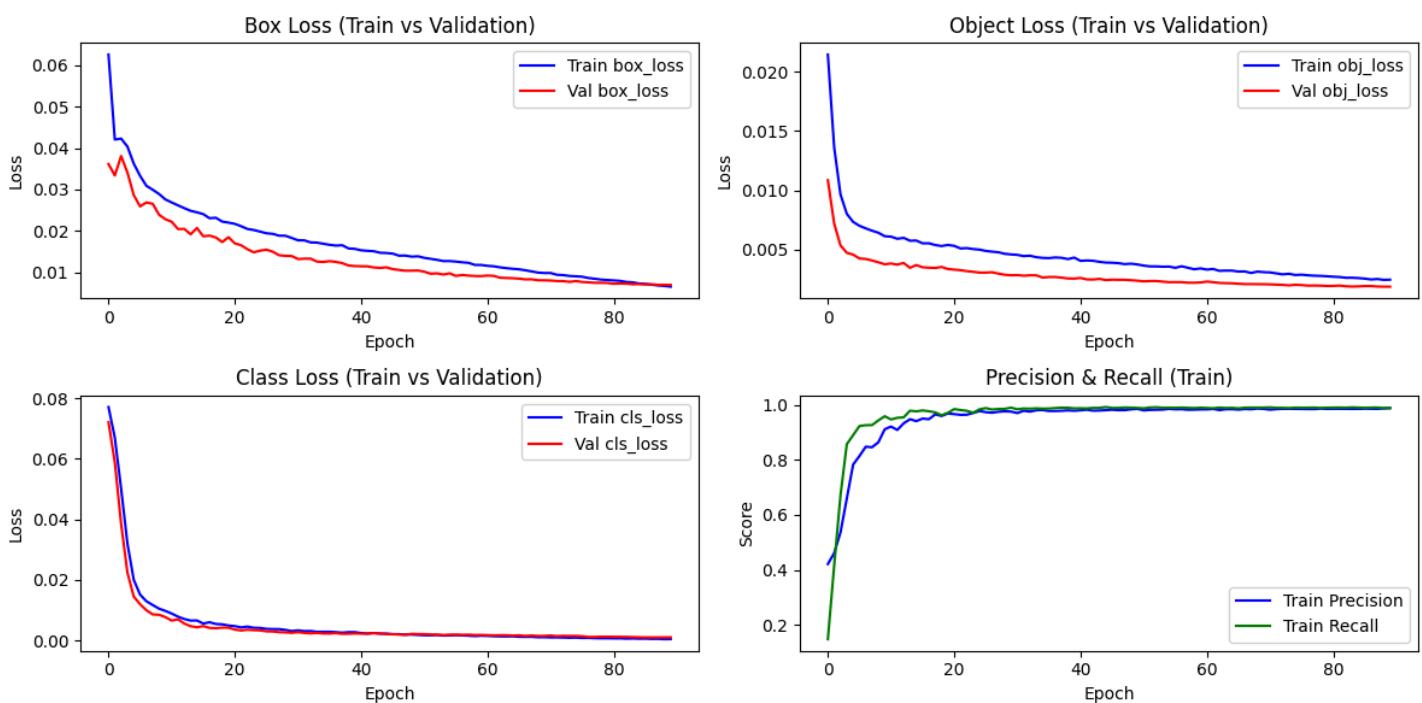
Các thông số đánh giá mô hình:

- Precision (P): Tỷ lệ phần trăm các dự đoán được mô hình cho là đúng so với tổng số dự đoán dương
- Recall (R): Tỷ lệ phần trăm các mẫu thực sự đúng mà mô hình dự đoán đúng (khả năng phát hiện đúng các trường hợp thực).
- mAP@0.5: Giá trị trung bình của độ chính xác (Average Precision) khi ngưỡng IoU (Intersection over Union) là 0.5, tức là dự đoán được xem là đúng nếu vùng dự đoán và vùng thực có độ chồng lấn $\geq 50\%$.
- mAP@0.5:0.95: Giá trị trung bình của độ chính xác tính trên nhiều ngưỡng IoU từ 0.5 đến 0.95 (bước 0.05), đánh giá mô hình toàn diện hơn trên các mức độ chồng lấn khác nhau.

Class	Images	Instances	P	R	mAP50	mAP50-95
all	920	920	0.987	0.988	0.989	0.944
Cam re trai	920	40	0.997	1	0.995	0.979
Giao nhau voi duong uu tien	920	40	0.997	1	0.995	0.945
Cam do xe ngay le	920	40	0.997	1	0.995	0.952
Cam do xe ngay chan	920	40	0.997	1	0.995	0.944
Duong giao nhau	920	40	0.997	1	0.995	0.931
Cam di nguoc chieu	920	40	0.971	0.955	0.976	0.908
Noi giao nhau theo vong xuyen	920	40	0.997	1	0.995	0.914
Giao nhau voi duong khong uu tien	920	40	0.998	1	0.995	0.933
Cam do xe	920	40	0.997	1	0.995	0.939
Toc do toi da 40 km/h	920	40	0.974	0.975	0.973	0.963
Tre em qua duong	920	40	0.998	1	0.995	0.959
Cam re phai	920	40	0.928	1	0.994	0.969
Cam bam coi	920	40	0.997	0.95	0.971	0.959
Toc do toi da 80 km/h	920	40	0.997	0.975	0.995	0.947
Toc do toi da 60 km/h	920	40	0.971	0.95	0.947	0.9
Cam do re trai	920	40	0.997	1	0.995	0.985
Cho ngoac nguy hiem phia ben phai	920	40	0.974	1	0.982	0.876
Huong di thang phai theo	920	40	0.976	0.95	0.993	0.951
Duong nguoi di bo cat ngang	929	40	0.999	1	0.995	0.929
Cam quay dau	920	40	0.997	1	0.995	0.963
Cho ngoac nguy hiem phia ben trai	920	40	0.994	0.975	0.992	0.951
Cam xe may 2 banh	920	40	0.927	1	0.992	0.949

Hình 9. Precision, Recall và mAP theo từng biến báo.

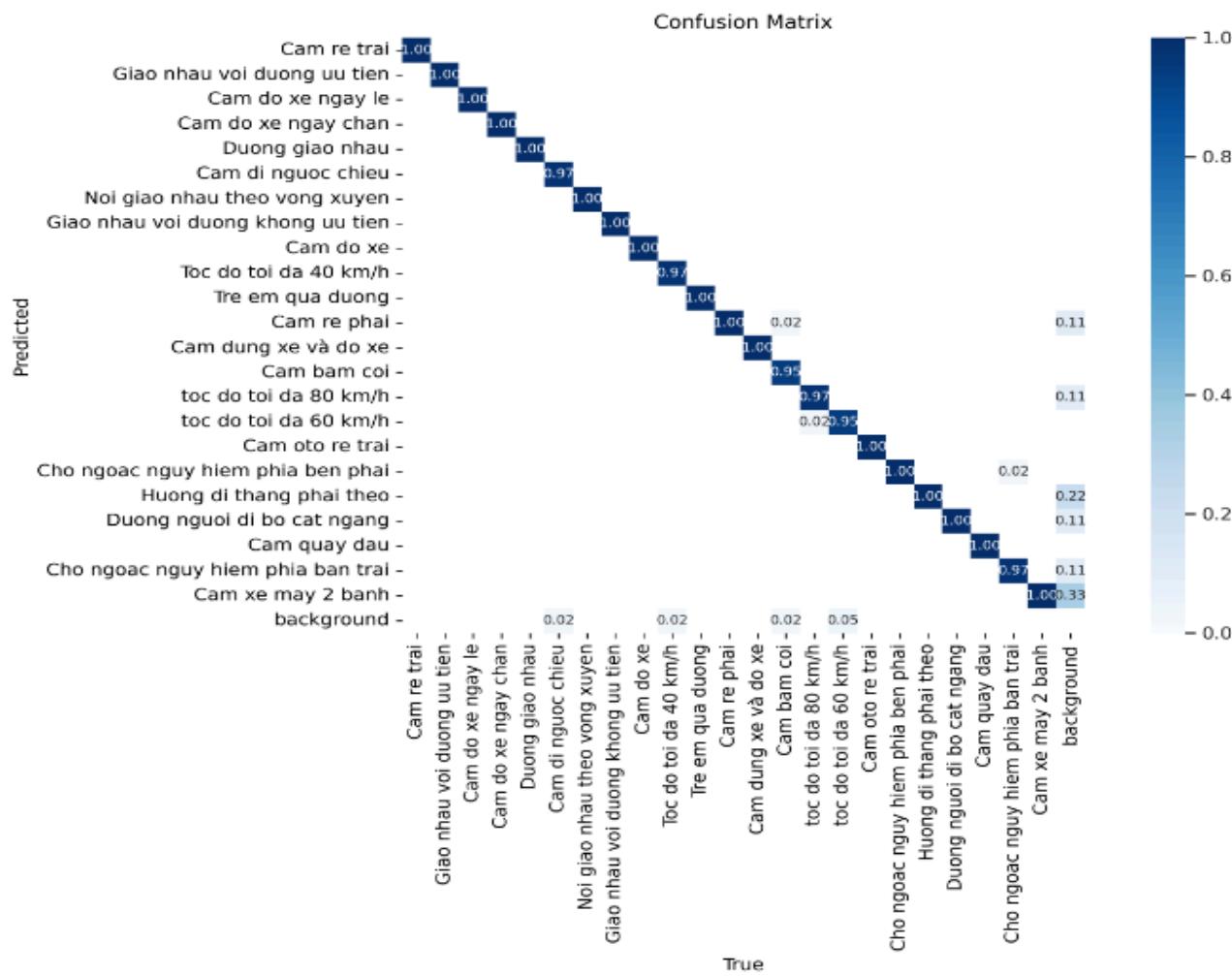
- P (Precision): Độ chính xác rất cao, hầu hết các biến báo đều trên 0.9, nghĩa là mô hình dự đoán chính xác rất tốt, ít báo sai.
- R (Recall): Đa số gần 1 hoặc trên 0.95, nghĩa là mô hình phát hiện hầu hết các đối tượng thật sự.
- mAP50 và mAP50-95: Chỉ số trung bình về độ chính xác, đều rất cao (gần hoặc trên 0.9), cho thấy mô hình có hiệu suất xuất sắc ở các ngưỡng IoU khác nhau.
- Các một số biến báo có recall thấp hơn một chút (vd: "Cam re phai" recall = 1 nhưng precision thấp hơn hay "Cam xe hai banh" recall = 1 nhưng precision thấp hơn), thể hiện độ khó khác nhau của từng lớp.



Hình 10. Biểu đồ Loss và Precision-Recall trên tập huấn luyện và đánh giá.

- Ba biểu đồ loss (Box Loss, Object Loss, Class Loss) trên tập huấn luyện và validation giảm đều, cho thấy mô hình dần hội tụ tốt.

- Validation loss thấp hơn hoặc bằng train loss, biểu hiện mô hình không bị overfitting rõ ràng.
- Biểu đồ Precision & Recall trên tập train đều tăng nhanh và ổn định ở giá trị gần 1, chứng tỏ mô hình phân loại chính xác và thu hồi đầy đủ các đối tượng.
- Mô hình có khả năng học tốt, cân bằng giữa Precision và Recall .

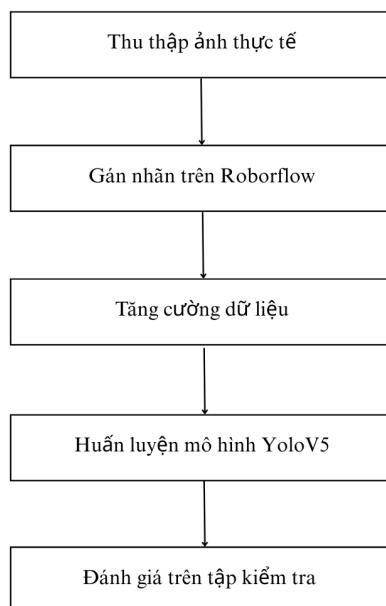


Hình 11. Ma trận nhầm lẫn.

- Ma trận thể hiện mô hình rất chính xác, chỉ một số ít nhầm lẫn giữa các biến báo tương tự nhau hoặc khó phân biệt.
- Một số giá trị nhỏ (như 0.02, 0.05, 0.11, 0.22) cho thấy có vài trường hợp nhầm lẫn giữa các lớp nhất định, ví dụ:
 - + "Cam re phai" có vài dự đoán nhầm sang lớp khác.

- + "Cho ngoat nguy hiem phia ben phai" có mức nhầm cao hơn (0.22) sang lớp khác.
- + "background" bị dự đoán nhầm sang một vài lớp nhỏ.

2.2.3 Sơ đồ quy trình mô hình AI

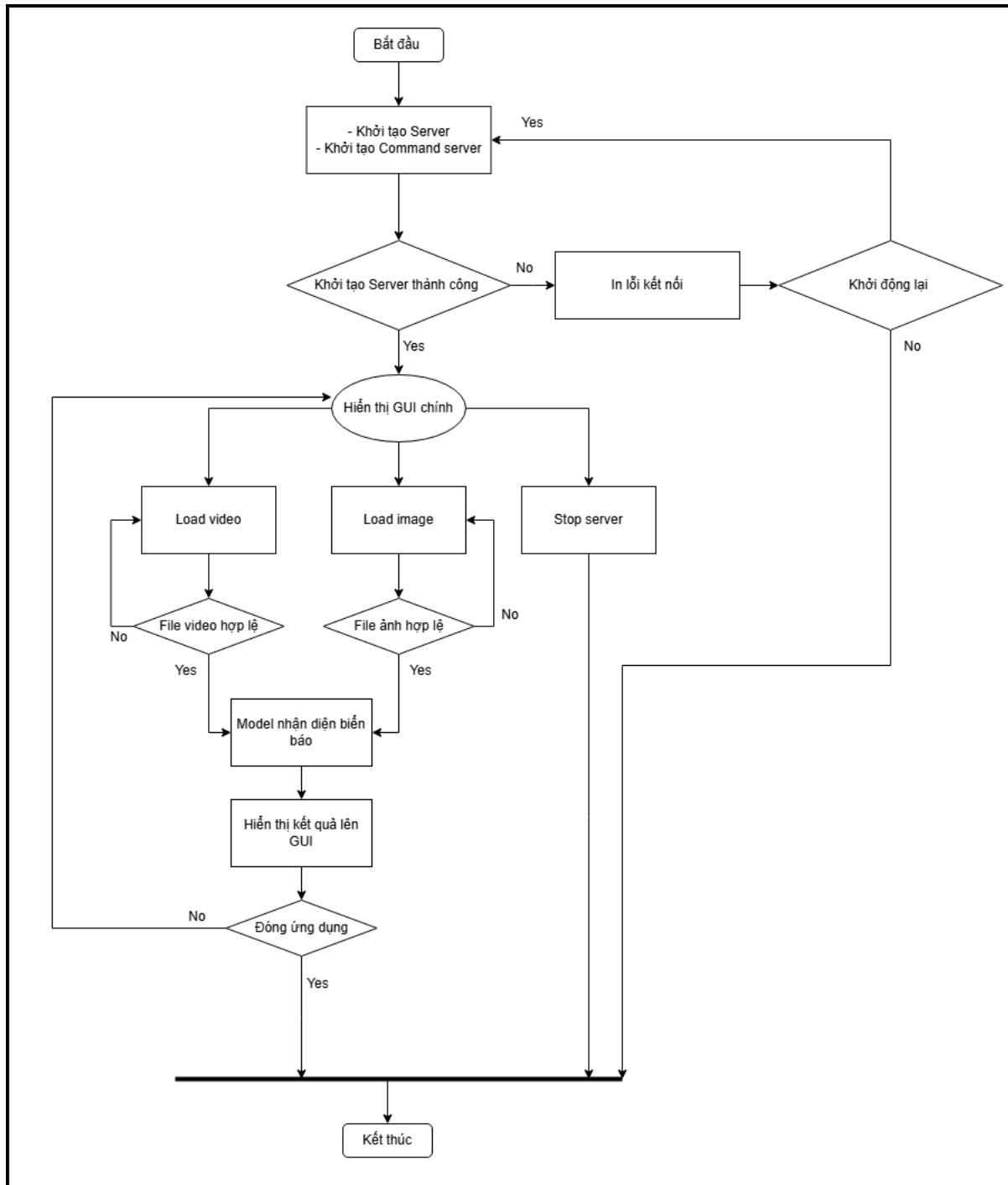


Hình 12. Sơ đồ quy trình xử lý mô hình AI

2.2.4 Lý do chọn giải pháp

- YOLO v5 nhanh và chính xác, tối ưu tốt cho thiết bị nhúng.
- Roboflow giúp rút ngắn thời gian gán nhãn, đảm bảo định dạng đồng bộ và hỗ trợ augmentation dễ dàng [4].
- Kaggle cung cấp tài nguyên GPU miễn phí, môi trường huấn luyện ổn định, giúp tiết kiệm chi phí so với Google Colab.

2.3. Giải pháp phần mềm

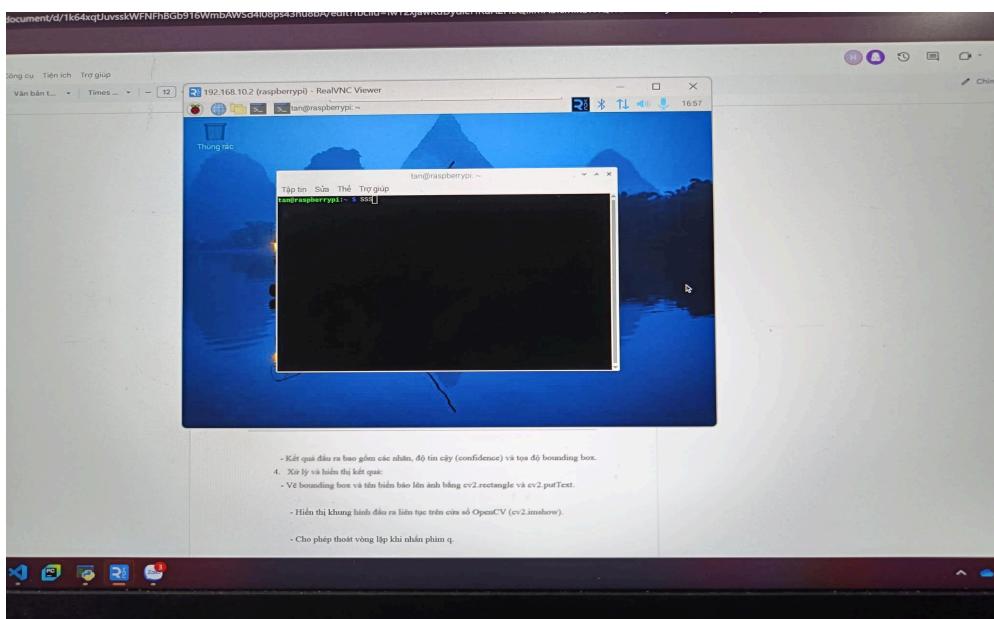


Hình 13. Sơ đồ khái quát trình xử lý GUI

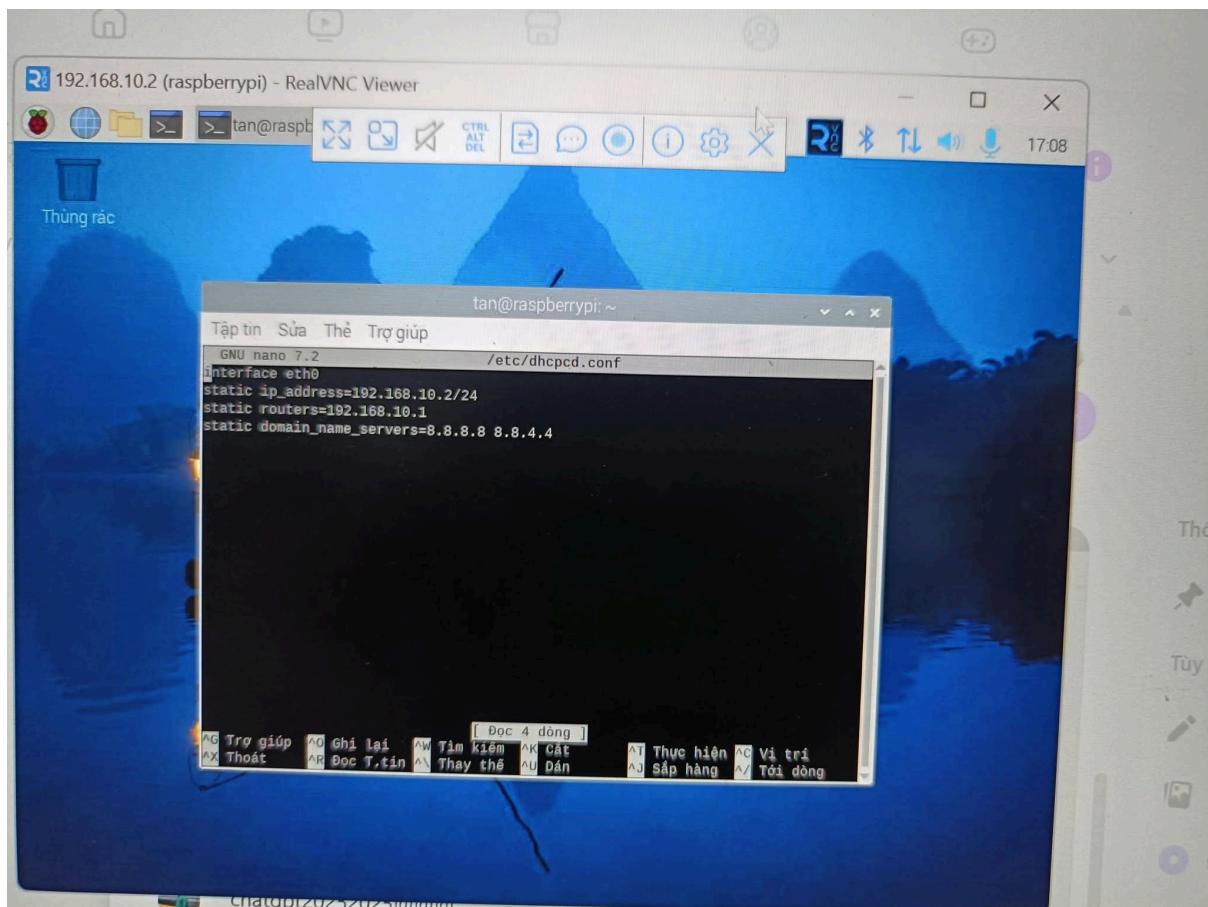
3. Kết quả

3.1. Ngôn ngữ và thư viện sử dụng

- Trên Laptop cá nhân (Server):
 - PyTorch: Thư viện chính để nạp và thực thi mô hình YOLO v5 đã huấn luyện (.pt).
 - OpenCV (cv2): Xử lý hình ảnh, thu nhận khung hình từ camera và hiển thị kết quả nhận diện.
 - Torchvision / numpy: Tiền xử lý dữ liệu, chuyển ảnh về tensor và chuẩn hóa đầu vào.
 - Yolov5 Ultralytics : Hỗ trợ load model để thực hiện dự đoán
-Socket: Thư viện dùng để lập trình socket sử dụng phương thức UDP
- Trên Raspberry (Client) :
 - Hệ điều hành : Linux raspbianpi 6.12.25+rpt-rpi-v8
 - Công cụ điều khiển: RealVNC kết hợp với dây mạng LAN kết nối qua cổng Ethernet nhằm điều khiển raspberry trên Laptop
 - IDE: Thonny nhẹ dễ triển khai



Hình 14 : Giao diện chính hệ điều Raspberry 3



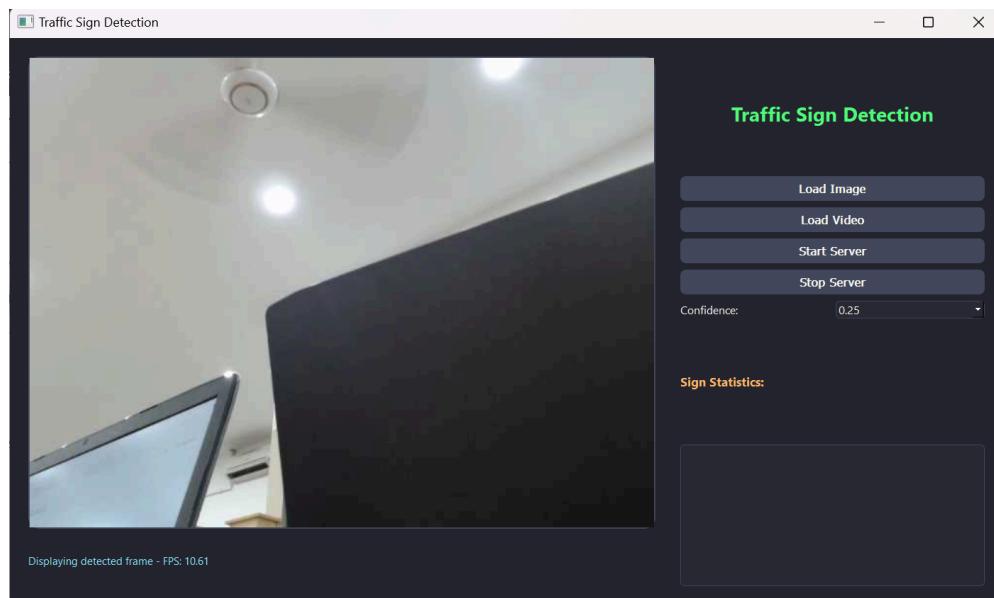
Hình 15 : Giao diện điều chỉnh file client chạy nền mỗi khi khởi động Raspberry 3

3.2 Cấu trúc xử lý phần mềm

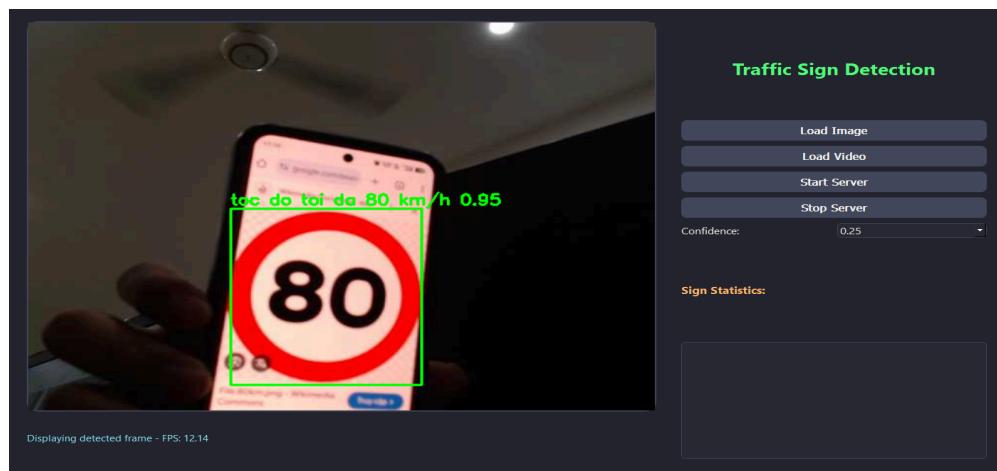
1. Nạp mô hình:
Mô hình YOLO v5 đã được huấn luyện từ Kaggle, lưu dưới định dạng .pt, được nạp bằng torch.hub.load(...). Trọng số mô hình được tải từ local để tiết kiệm thời gian truy xuất [3].
 2. Mở camera và thu nhận ảnh liên tục:
 - Dùng cv2.VideoCapture(0) để mở webcam USB.
 3. Gửi ảnh qua cho server thông qua phương thức UDP
 3. Dự đoán với YOLO v5:

- Ảnh đầu vào được đưa vào mô hình để suy luận.
 - Kết quả đầu ra bao gồm các nhãn, độ tin cậy (confidence) và tọa độ bounding box.
4. Xử lý và hiển thị kết quả:
- Vẽ bounding box và tên biển báo lên ảnh bằng cv2.rectangle và cv2.putText.
 - Hiển thị khung hình đầu ra liên tục trên cửa sổ OpenCV (cv2.imshow).
 - Phát âm thanh cảnh báo

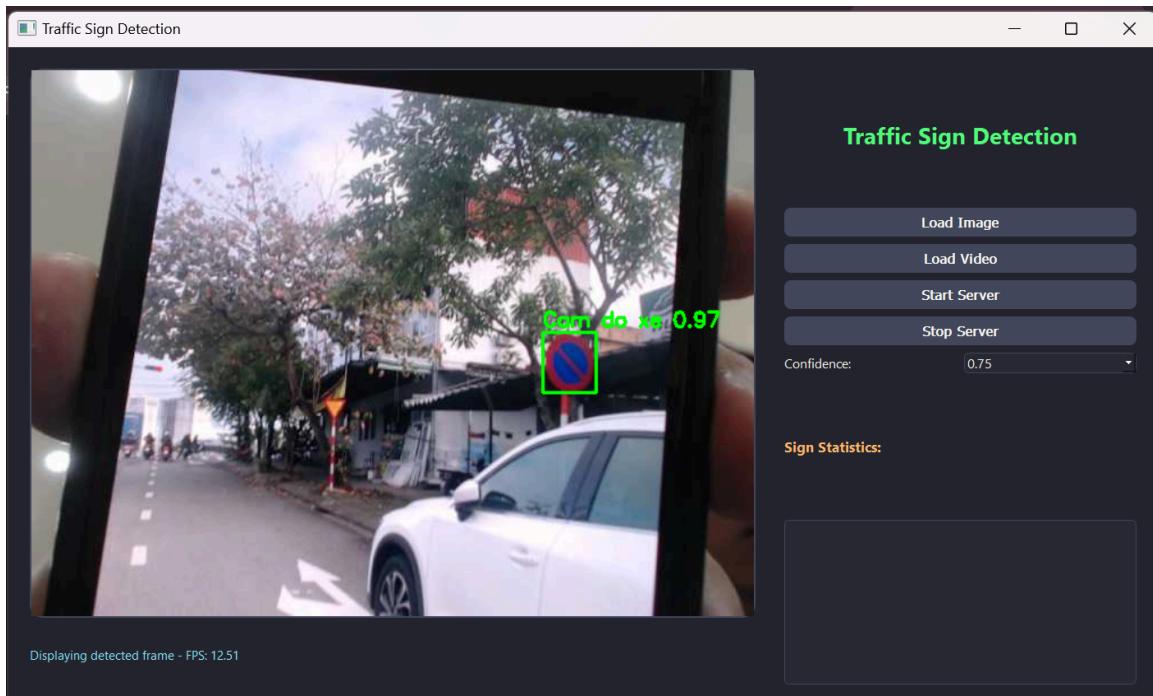
3.3. Giao diện của phần mềm



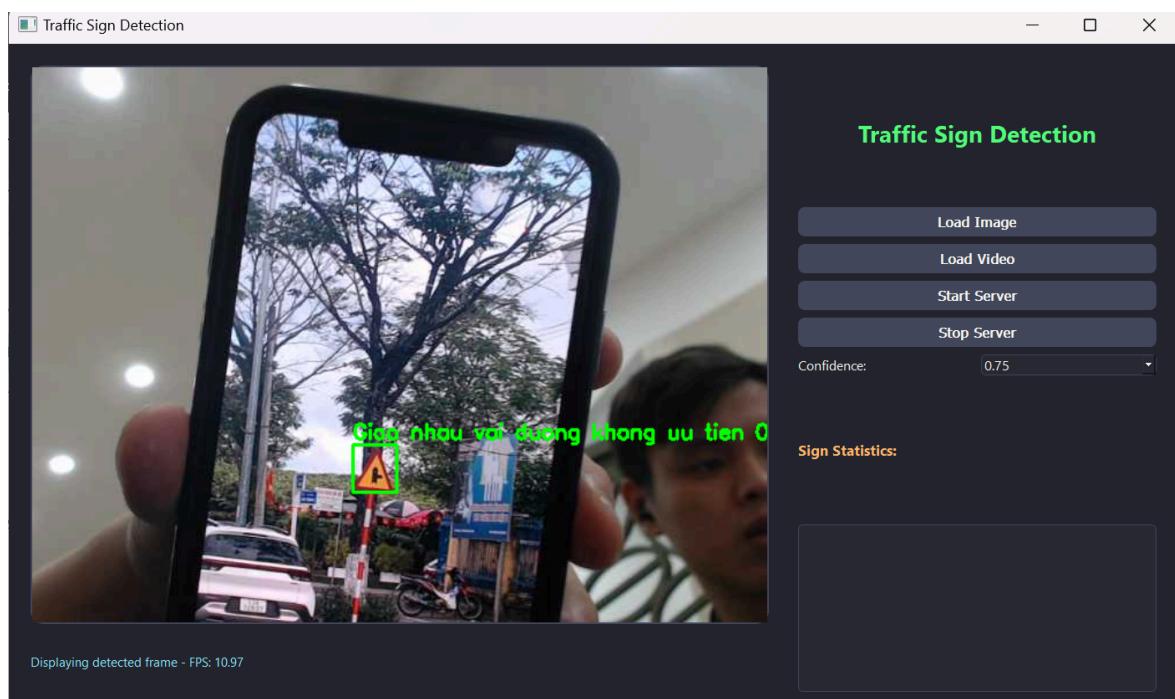
Hình 16 : Giao diện chính của phần mềm



Hình 17 : Dự đoán các biển báo với chức năng Real Time



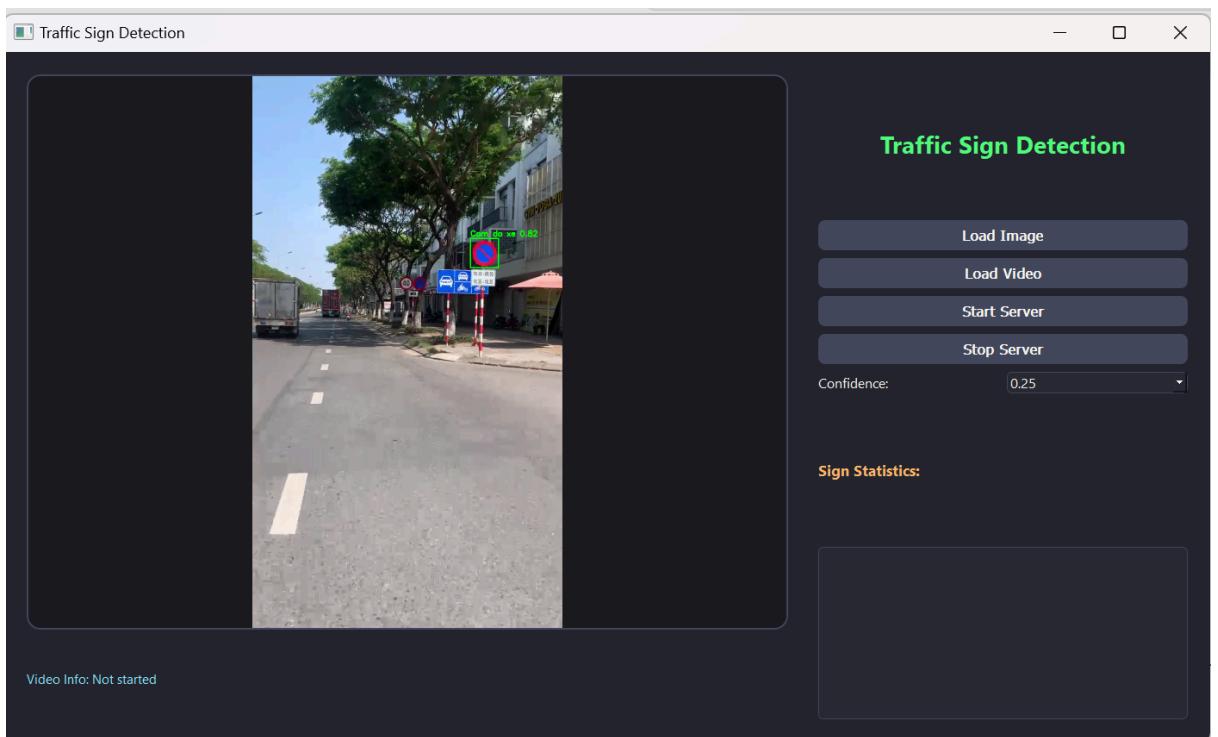
Hình 18 :Dự đoán các biển báo với chức năng Real Time



Hình 19 :Dự đoán các biển báo với chức năng Real Time



Hình 20 :Dự đoán các biển báo với chức năng Load Image



Hình 21 :Dự đoán các biển báo với chức năng Load Video

4. Kết luận

Nhóm đã xây dựng thành công một hệ thống phát hiện biển báo giao thông theo thời gian thực với kiến trúc phân tán, trong đó laptop được sử dụng như một server xử lý trung tâm, còn Raspberry Pi đóng vai trò lấy dữ liệu từ camera và gửi thông tin hình ảnh sang server qua giao thức UDP. Cách tiếp cận này giúp tận dụng hiệu năng xử lý mạnh mẽ của laptop để chạy mô hình YOLO v5 nhận diện biển báo, đồng thời sử dụng phần cứng nhỏ gọn và linh hoạt của Raspberry Pi để thu thập dữ liệu từ camera.

4.1. Khó khăn

- Raspberry Pi có tài nguyên hạn chế, không thể xử lý mô hình trực tiếp nên phải gửi dữ liệu camera qua UDP sang laptop server.
- Giao thức UDP không đảm bảo truyền dữ liệu an toàn, gây khó khăn trong việc đồng bộ và xử lý dữ liệu thời gian thực.
- Việc tối ưu hiệu năng xử lý trên laptop để đảm bảo tốc độ nhận diện nhanh và ổn định là thách thức lớn.
- Thu thập dữ liệu biển báo đa dạng, đủ chất lượng để huấn luyện mô hình mất nhiều thời gian.

4.2. Hướng phát triển

- Tích hợp cảnh báo biển báo nguy hiểm theo mức độ ưu tiên.
- Mở rộng lưu trữ và phân tích dữ liệu biển báo phục vụ nghiên cứu.
- Xây dựng giao diện người dùng trực quan hơn (GUI hoặc web).
- Đầu tư về phần cứng tốt hơn nhằm đáp ứng nhu cầu nhận diện với nhiều biển báo hơn

5. Tài liệu tham khảo

- [1] E. Upton and G. Halcrow, *Raspberry Pi User Guide*, 4th ed. Chichester, UK: Wiley, 2016.
- [2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 779–788.
- [3] Ultralytics, “YOLOv5 by Ultralytics,” GitHub Repository, 2020. [Online]. Available: <https://github.com/ultralytics/yolov5>
- [4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection,” *arXiv preprint arXiv:2004.10934*, 2020
- [5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 779–788.
- [6] Roboflow, “Roboflow: Annotate, Augment, and Export Computer Vision Datasets,” 2021. [Online]. Available: <https://roboflow.com/>
- [7] A. Paszke et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library,” in *Advances in Neural Information Processing Systems 32 (NeurIPS)*, 2019.
- [8] G. Bradski, “The OpenCV Library,” *Dr. Dobb’s Journal of Software Tools*, 2000.