XÂY DỰNG HỆ THỐNG CẢNH BÁO ÙN TẮC GIAO THÔNG ĐÔ THỊ THEO THỜI GIAN THỰC

Trần Hồng Tùng[a], Nguyễn Quốc Cường[b], Trần Văn Thiên[c], Nguyễn Ngọc Nhân[d], Tạ Quốc Ý[e]

Tóm tắt:

Nghiên cứu trình bày kết quả phát triển hệ thống cảnh báo ùn tắc giao thông đô thị theo thời gian thực. Từ thực trạng ùn tắc và hạn chế của các giải pháp hiện tại tại Việt Nam, nghiên cứu đề xuất một kiến trúc hệ thống phân tán với bốn lớp chính: thu thập dữ liệu camera, xử lý AI, backend và ứng dụng đa nền tảng. Sử dụng mô hình YOLOv8n được huấn luyện trên bộ dữ liệu 20.000 ảnh giao thông tại Đà Nẵng, hệ thống đạt độ chính xác trung bình (mAP@0.5) 92,4% và tốc độ xử lý 45 FPS trên GPU RTX 3060. Backend xây dựng bằng Node.js, PostgreSQL và Redis xử lý đồng thời 25 camera với độ trễ API trung bình 145ms. Web app (React/Vite) hoàn thiện 11 trang, mobile app (Flutter) có 16 màn hình trên iOS và Android. Hệ thống được thử nghiệm trên dữ liệu mô phỏng từ 15 luồng camera tại Đà Nẵng, ghi nhận hơn 800 cảnh báo với độ chính xác trên 90%. Các kết quả này khẳng định tính khả thi của hệ thống trong bối cảnh nghiên cứu sinh viên và tiềm năng ứng dụng thực tế nhằm giảm ùn tắc, tiết kiệm thời gian và nâng cao an toàn giao thông đô thị.

Từ khóa: *Hệ thống giao thông thông minh, YOLOv8, thị giác máy tính, phát hiện ùn tắc, React, Flutter.*

DEVELOPMENT OF A REAL-TIME URBAN TRAFFIC CONGESTION WARNING SYSTEM

Abstract:

This study presents the design and development of a real-time urban traffic congestion alert system. Motivated by severe congestion and limitations of existing solutions in Vietnam, the study proposes a distributed four-layer architecture comprising camera data acquisition, AI processing, backend services and multi-platform applications. Using a YOLOv8n model trained on a dataset of 20,000 traffic images from Da Nang, the system achieves a mean average precision (mAP@0.5) of 92.4% and a processing speed of 45 FPS on an RTX 3060 GPU. The backend, built with Node.js, PostgreSQL and Redis, concurrently handles up to 25 cameras with an average API latency of 145ms. The web application (React/Vite) delivers 11 functional

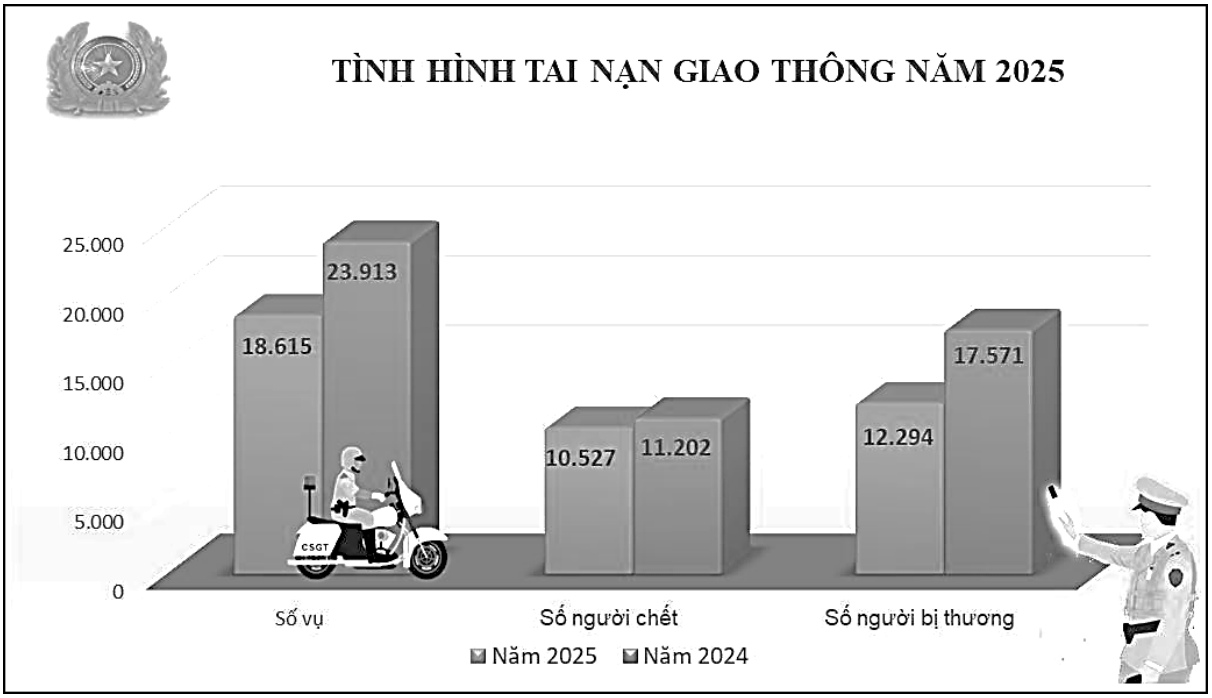
[a] Khoa Công nghệ Thông tin, Trường Đại Học Đông Á. e-mail: tung99863@donga.edu.vn
[b] Khoa Công nghệ Thông tin, Trường Đại Học Đông Á. e-mail: cuong100194@donga.edu.vn
[c] Khoa Công nghệ Thông tin, Trường Đại Học Đông Á. e-mail: thien98405@donga.edu.vn
[d] Khoa Công nghệ Thông tin, Trường Đại Học Đông Á. e-mail: nhan99443@donga.edu.vn
[e] Khoa Công nghệ Thông tin, Trường Đại Học Đông Á. e-mail: ytq@donga.edu.vn

pages, while the mobile app (Flutter) provides 16 screens on iOS and Android. A simulation-based experiment on 15 camera streams in Da Nang recorded over 800 alerts with accuracy exceeding 90%. These results confirm the system's feasibility within an undergraduate research context and its practical potential to reduce congestion, save travel time and enhance urban traffic safety.

Keywords: *Smart traffic system, YOLOv8, computer vision, congestion detection, React, Flutter.*

Đặt vấn đề

Tình trạng ùn tắc giao thông tại các đô thị lớn ở Việt Nam ngày càng nghiêm trọng. Theo Bộ Công an, năm 2025 cả nước xảy ra 15.367 vụ tai nạn giao thông, làm 6.578 người thiệt mạng và gây thiệt hại kinh tế lớn (Bộ Công an, 2025). Riêng tại Đà Nẵng, số điểm thường xuyên ùn tắc đã tăng từ 47 điểm (2023) lên 72 điểm (2025) (Cổng Thông tin điện tử thành phố Đà Nẵng, 2026). Các giải pháp hiện tại như camera giám sát thụ động, cảm biến dưới mặt đường, điều tiết thủ công hay ứng dụng bản đồ dựa trên GPS đều có hạn chế về chi phí, độ chính xác, thời gian phản ứng hoặc khả năng bao phủ (Varghese & Sambath, 2024; Wang et al., 2023).



Hình 1. Tình trạng tai nan giao thông của năm 2024 và 2025

(Nguồn: Bộ Công An)

Bài toán giám sát và phát hiện ùn tắc tự động đã được nghiên cứu rộng rãi. Các hệ thống đầu tiên dựa trên cảm biến vòng từ hoặc radar có chi phí cao. Sự phát triển của thị giác máy tính và học sâu, đặc biệt là mô hình YOLO, đã mở ra hướng tiếp cận mới sử dụng camera sẵn có (Bochkovskiy et al., 2020; Redmon & Farhadi, 2018; Wojke et al., 2017). Các nghiên cứu gần đây đã áp dụng YOLOv5, YOLOv7 và YOLOv8 cho

bài toán đếm xe và phát hiện ùn tắc, đạt độ chính xác từ 88-91% (Bochkovskiy et al., 2020; Varghese & Sambath, 2024; Wang et al., 2023).

Từ khoảng trống đó, nghiên cứu hướng tới ba mục tiêu: (i) thiết kế kiến trúc hệ thống giám sát, phân tích và cảnh báo ùn tắc theo thời gian thực tích hợp bốn lớp; (ii) xây dựng và đánh giá mô hình YOLOv8 trên bộ dữ liệu giao thông Việt Nam; (iii) thử nghiệm trên dữ liệu mô phỏng từ 15 luồng camera tại Đà Nẵng, đánh giá hiệu năng toàn hệ thống.

Đối tượng và phương pháp nghiên cứu

Đối tượng nghiên cứu

Đối tượng nghiên cứu là hệ thống giao thông thông minh toàn diện bao gồm bốn module chính: ((1) nhận diện phương tiện YOLOv8, (2) web app (React/Vite) (Meta Platforms, 2025), (3) mobile app (Flutter) (Flutter Documentation, 2025) và (4) backend (Node.js/PostgreSQL/Redis) (Node.js Foundation, 2025). Hệ thống nhận diện 6 loại đối tượng (xe máy, ô tô con, xe tải, xe buýt, xe đạp, người đi bộ), đếm lưu lượng, phát hiện ùn tắc, hiển thị trên bản đồ tương tác và gửi cảnh báo đa kênh. (web popup, push notification).

Dữ liệu và cấu hình thực nghiệm

Bộ dữ liệu huấn luyện gồm 20.000 ảnh giao thông được tổng hợp từ nhiều nguồn khác nhau nhằm đảm bảo tính đa dạng về góc quay, điều kiện thời tiết và mật độ phương tiện. Cụ thể:

- 10.000 ảnh từ hệ thống camera giao thông tại thành phố Đà Nẵng do nhóm nghiên cứu tự thu thập.

- 3.000 ảnh từ Microsoft COCO Dataset (Microsoft COCO Dataset, 2026).

- 2.000 ảnh từ Open Images Dataset V7 (Google Research, 2026).

- 3.000 ảnh do nhóm nghiên cứu tự quay chụp tại các tuyến đường đô thị.

- 2.000 ảnh từ các nguồn công khai trên Internet phục vụ mục đích nghiên cứu.

Toàn bộ dữ liệu được gán nhãn thủ công bằng công cụ Label Studio (HumanSignal, 2026). Sau quá trình làm sạch và loại bỏ dữ liệu nhiễu, bộ dữ liệu cuối cùng gồm 43.500 bounding box thuộc 6 lớp đối tượng: xe máy, ô tô con, xe tải, xe buýt, xe đạp và người đi bộ.

Tổng số bounding box được gán nhãn thủ công bằng Label Studio là 43.500, phân bố: xe máy (42,5%), ô tô con (29,4%), xe tải (9,7%), xe buýt (7,1%), xe đạp (5,8%), người đi bộ (5,5%). Dữ liệu được chia ngẫu nhiên: tập huấn luyện 14.000 ảnh (70%), kiểm định 3.000 ảnh (15%), kiểm tra 3.000 ảnh (15%) (Ultralytics, 2025).

Với module backend và ứng dụng, hệ thống được thử nghiệm trong môi trường mô phỏng dựa trên 15 luồng dữ liệu camera giao thông công cộng tại Đà Nẵng. Thời gian thử nghiệm kéo dài 30 ngày mô phỏng. Để đánh giá sơ bộ tính khả dụng của hệ thống, nhóm nghiên cứu đã thực hiện một đợt thử nghiệm đánh giá chức năng trên 100 người tham gia (sinh viên, nhân viên văn phòng, tài xế công nghệ).

Bảng 1. Đặc điểm dữ liệu và cấu hình thực nghiệm của các module

Hạng mục	Module nhận diện	Module Backend & Ứng dụng
Quy mô dữ liệu	20.000 ảnh, 43.500 bounding box	15 camera, 805 cảnh báo, 100 người khảo sát
Đầu vào	Ảnh/video từ camera IP	API request, WebSocket, thao tác người dùng
Đầu ra	Lớp đối tượng, tọa độ box, lưu lượng	Dữ liệu JSON, cảnh báo, giao diện người dùng
Mô hình/kiến trúc	YOLOv8n + SORT tracking	Node.js/Express, PostgreSQL, Redis, React, Flutter
Đánh giá	mAP, FPS, Precision, Recall	QPS, độ trễ, độ chính xác cảnh báo, tỷ lệ hài lòng
Chiến lược	100 epochs, batch 16, augmentation	Kiểm thử đơn vị, tích hợp, tải, khảo sát

Bảng 1 trình bày đặc điểm dữ liệu và cấu hình thực nghiệm của từng module trong hệ thống. Có thể thấy module nhận diện tập trung vào xử lý ảnh và đánh giá bằng các chỉ số học máy như Precision, Recall, mAP và FPS, trong khi module backend và ứng dụng được đánh giá thông qua hiệu năng hệ thống, độ trễ xử lý và trải nghiệm người dùng.

Phương pháp nghiên cứu

Nghiên cứu phối hợp ba phương pháp: (i) nghiên cứu phát triển (70%) với quy trình Agile 6 sprint; (ii) phương pháp thực nghiệm (20%) so sánh YOLOv8 với YOLOv5, YOLOv7 và Faster R-CNN; (iii) phương pháp đánh giá chức năng (10%) với bảng hỏi 15 câu, thang đo Likert 5 điểm.

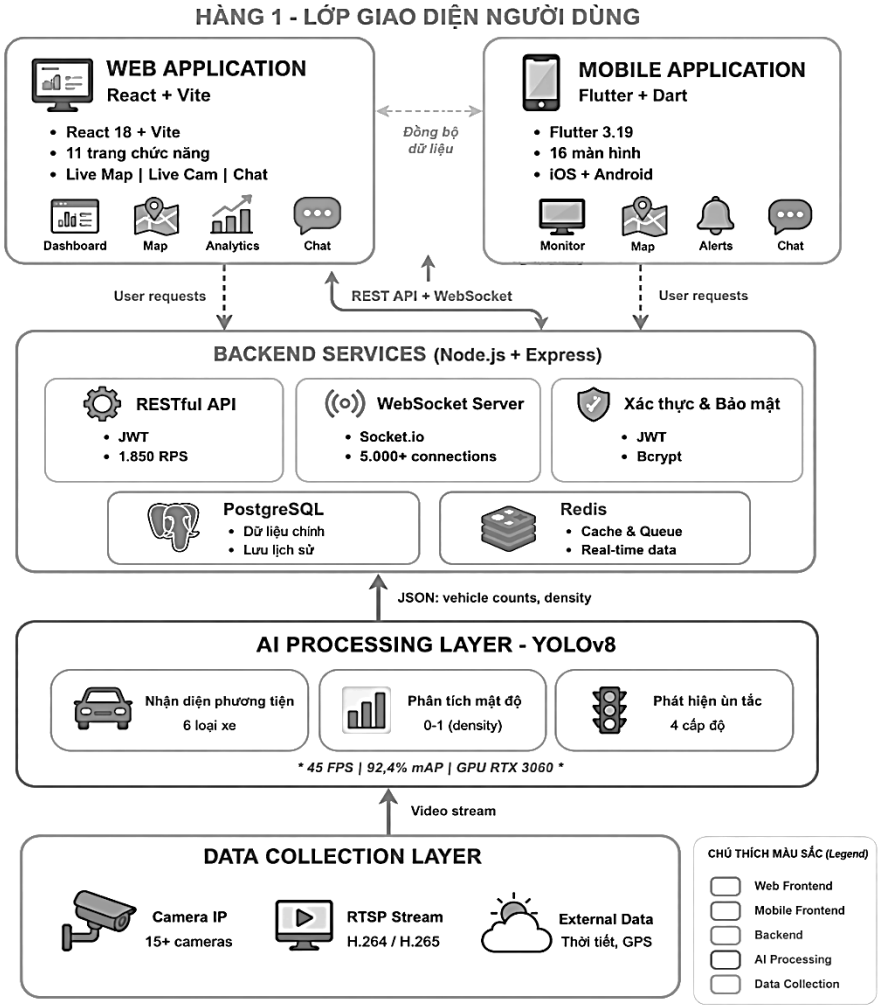
YOLOv8n được huấn luyện từ trọng số pre-trained trên COCO, sử dụng augmentation (mosaic, mixup, random flip, HSV) và SGD với learning rate 0,01 theo cosine annealing. Áp dụng ROI mask, frame skipping (xử lý 1/2-3 frame) và thuật toán Deep SORT tracking (Wojke et al., 2017). Xử lý ảnh sử dụng OpenCV (OpenCV team, 2025).

Backend xây dựng theo kiến trúc RESTful và WebSocket, JWT xác thực, PostgreSQL cho dữ liệu chính, Redis cho cache và message queue. Web frontend dùng React 18 với Vite, TailwindCSS, Leaflet, Chart.js. Mobile app dùng Flutter 3.19 với Riverpod, Google Maps và FCM.

Mô hình đề xuất

Kiến trúc hệ thống tổng thể

Hệ thống được tổ chức theo kiến trúc phân tán bốn lớp. Lớp thu thập dữ liệu gồm camera IP giao thông truyền luồng RTSP. Lớp xử lý AI dùng YOLOv8n trên GPU RTX 3060 (Varghese & Sambath, 2024). Lớp backend gồm Node.js/Express, WebSocket, PostgreSQL, Redis. Lớp ứng dụng gồm web app (React) (Meta Platforms, 2025) và mobile app (Flutter) (Flutter Documentation, 2025).



Hình 2. Sơ đồ kiến trúc tổng thể 4 lớp của hệ thống STS

Hình 2 mô tả kiến trúc tổng thể của hệ thống STS theo mô hình bốn lớp. Dữ liệu từ camera giao thông được thu thập và truyền đến module AI để nhận diện phương tiện và phân tích lưu lượng. Kết quả xử lý được gửi tới backend để lưu trữ, quản lý và cung cấp dữ liệu thời gian thực cho các ứng dụng web và di động.

Mô hình nhận diện phương tiện YOLOv8

Module nhận diện được thiết kế nhằm đáp ứng yêu cầu xử lý thời gian thực trong điều kiện giao thông đô thị có mật độ phương tiện cao và tài nguyên tính toán hạn chế. Mô hình cốt lõi sử dụng YOLOv8n (nano) với khoảng 3,2 triệu tham số và kích thước chỉ 6,2 MB. Đầu vào của mô hình là các khung hình video có kích thước 640×640 pixel.

Kiến trúc YOLOv8n sử dụng backbone C2f (Cross Stage Partial with Feature Fusion), được cải tiến từ CSPNet nhằm tăng khả năng trích xuất đặc trưng và giảm chi phí tính toán. Phần head áp dụng cơ chế decoupled head kết hợp anchor-free detection, giúp nâng cao tốc độ xử lý và độ chính xác trong bài toán phát hiện đối tượng.

Quá trình huấn luyện được thực hiện trên GPU NVIDIA RTX 3060 trong 100 epochs với batch size 16, thời gian huấn luyện khoảng 8 giờ. Sau khi phát hiện đối tượng, hệ thống sử dụng thuật toán SORT (Simple Online and Realtime Tracking) để theo dõi và gán định danh (ID) cho từng phương tiện qua các khung hình liên tiếp. Nhờ đó, hệ thống có thể thực hiện đếm lưu lượng phương tiện, theo dõi quỹ đạo di chuyển và hỗ trợ phát hiện các dấu hiệu ùn tắc giao thông.

Để nâng cao hiệu năng trong điều kiện giao thông thực tế tại Việt Nam, nhóm nghiên cứu áp dụng ba kỹ thuật tối ưu:

- ROI Mask: Chỉ xử lý vùng mặt đường quan tâm, giúp giảm khoảng 30% thời gian xử lý.

- Frame Skipping: Xử lý 1 trong mỗi 2-3 khung hình kết hợp nội suy kết quả, giúp tiết kiệm khoảng 40% tài nguyên tính toán.

- Điều chỉnh ngưỡng NMS: Giảm ngưỡng Non-Maximum Suppression từ 0,45 xuống 0,35 nhằm hạn chế hiện tượng phát hiện trùng lặp (false positive) trong điều kiện mật độ xe máy cao.

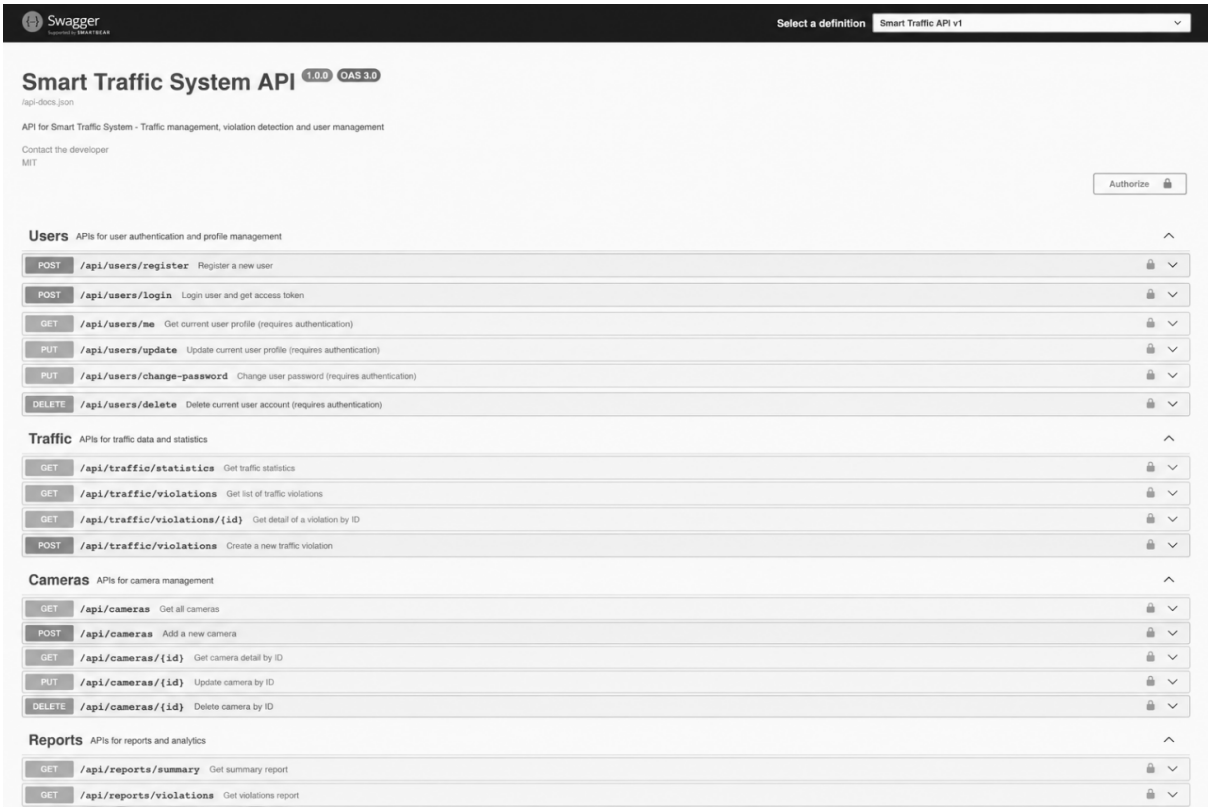
Mô đun Backend và ứng dụng đa nền tảng

Backend được xây dựng theo kiến trúc MVC (Model-View-Controller). API RESTful được thiết kế với các endpoint chính: xác thực (/api/auth), quản lý camera (/api/cameras), dữ liệu giao thông (/api/traffic), cảnh báo (/api/alerts), thông báo (/api/notifications) và chat (/api/chat). WebSocket server sử dụng Socket.io để đẩy dữ liệu thời gian thực (lưu lượng cập nhật mỗi 10 giây, cảnh báo tức thì). Cơ sở dữ liệu PostgreSQL có các bảng users, cameras, traffic_data, alerts, notifications, chat_messages, được tối ưu bằng các chỉ mục (idx_traffic_data_camera_time, idx_alerts_camera_created). Redis được dùng cho cả caching (tỷ lệ hit 78%) và message queue xử lý bất đồng bộ các tác vụ nặng như gửi push notification hàng loạt.

Web application (React/Vite) hoàn thiện 11 trang, bao gồm DashboardHome (tổng quan với 4 card thống kê và biểu đồ line chart), AnalyticsPage (phân tích nâng

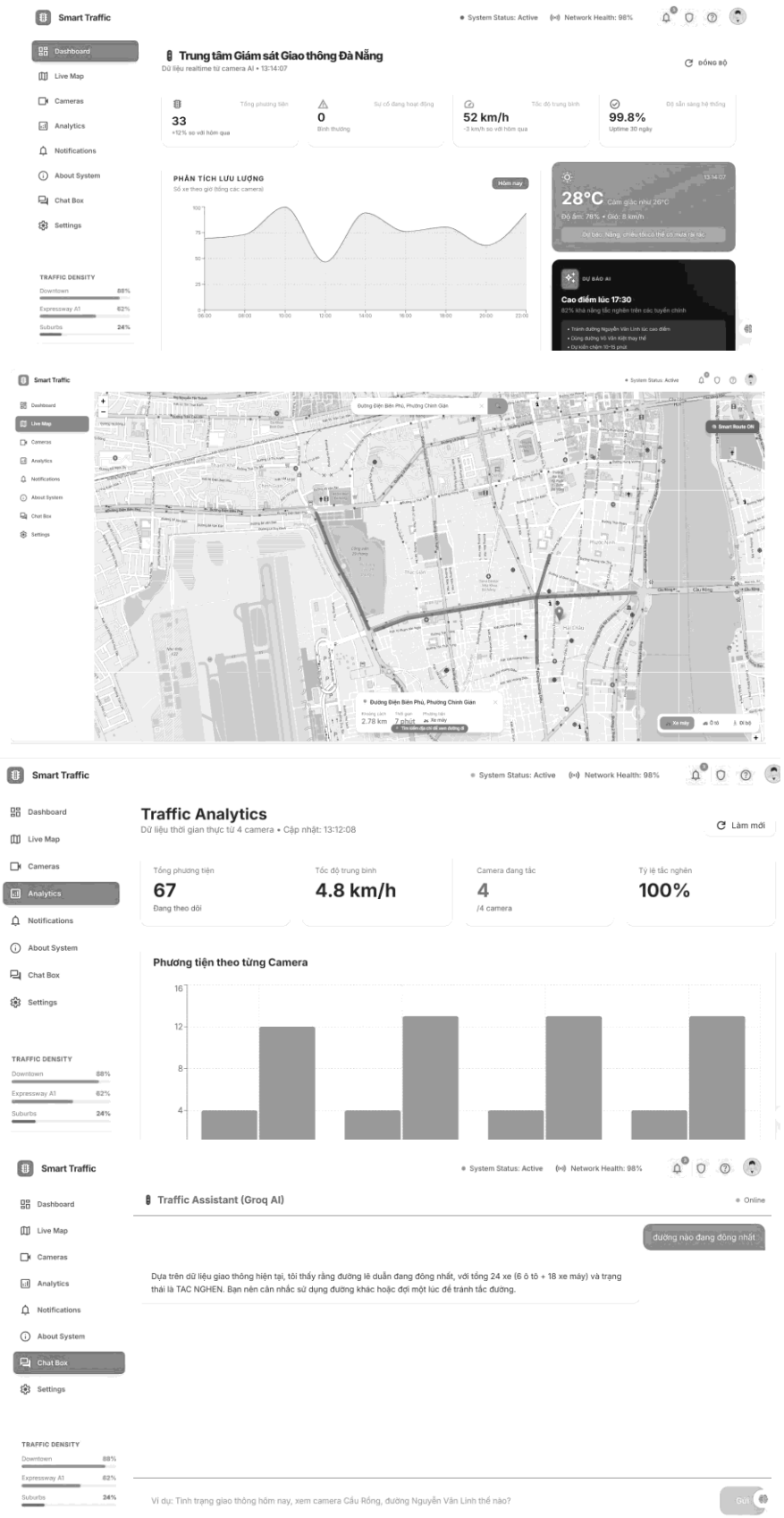
cao, xuất báo cáo PDF/CSV), CamerasPage (grid/list camera, live stream HLS), MapPage (Leaflet + OpenStreetMap với heatmap ùn tắc), ChatPage (Socket.io + chatbot tự động), NotificationsPage, SettingsPage và các trang xác thực. Giao diện responsive, sử dụng TailwindCSS và Shadcn/ui.

Mobile application (Flutter) hoàn thiện 16 màn hình trên cả iOS và Android, bao gồm Monitor (màn hình chính với bản đồ thu nhỏ và cards thông số), Map (Google Maps fullscreen với marker camera và polyline đề xuất), Alerts (danh sách cảnh báo phân loại theo mức độ xanh/vàng/đỏ), Chat (tin nhắn văn bản và hình ảnh, quick replies), Notifications, Settings và các màn hình xác thực. Push notification được triển khai qua Firebase Cloud Messaging (FCM).

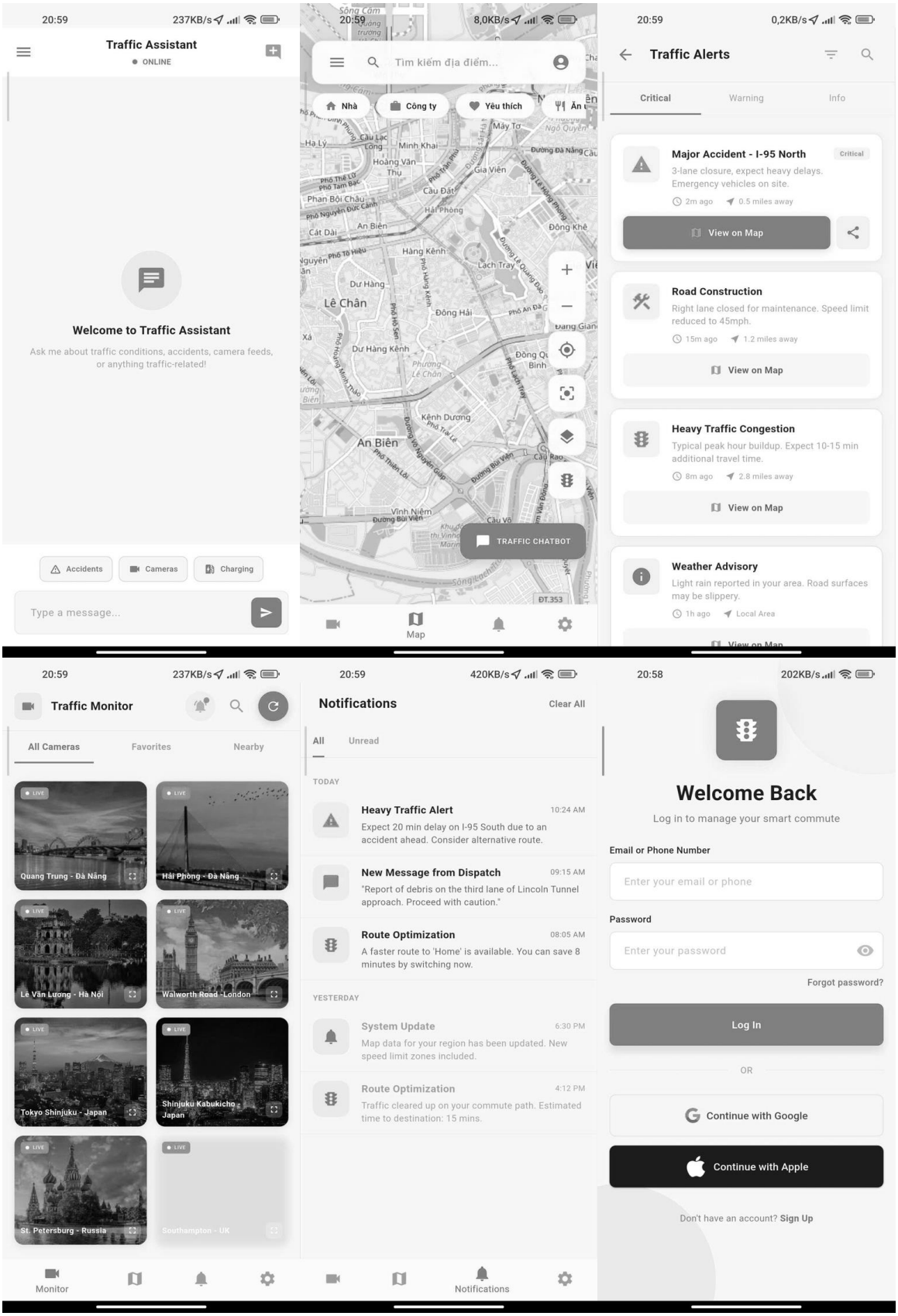


Hình 3. Giao diện tài liệu API hệ thống STS trên nền tảng Swagger

Hình 3 minh họa giao diện tài liệu API được xây dựng bằng Swagger/OpenAPI. Công cụ này cho phép kiểm thử trực tiếp các endpoint, quản lý tài liệu kỹ thuật và hỗ trợ tích hợp giữa backend với các ứng dụng web, mobile một cách thuận tiện và nhất quán.



Hình 4. Giao diện website



Hình 5. Giao diện mobile app

Kết quả và thảo luận

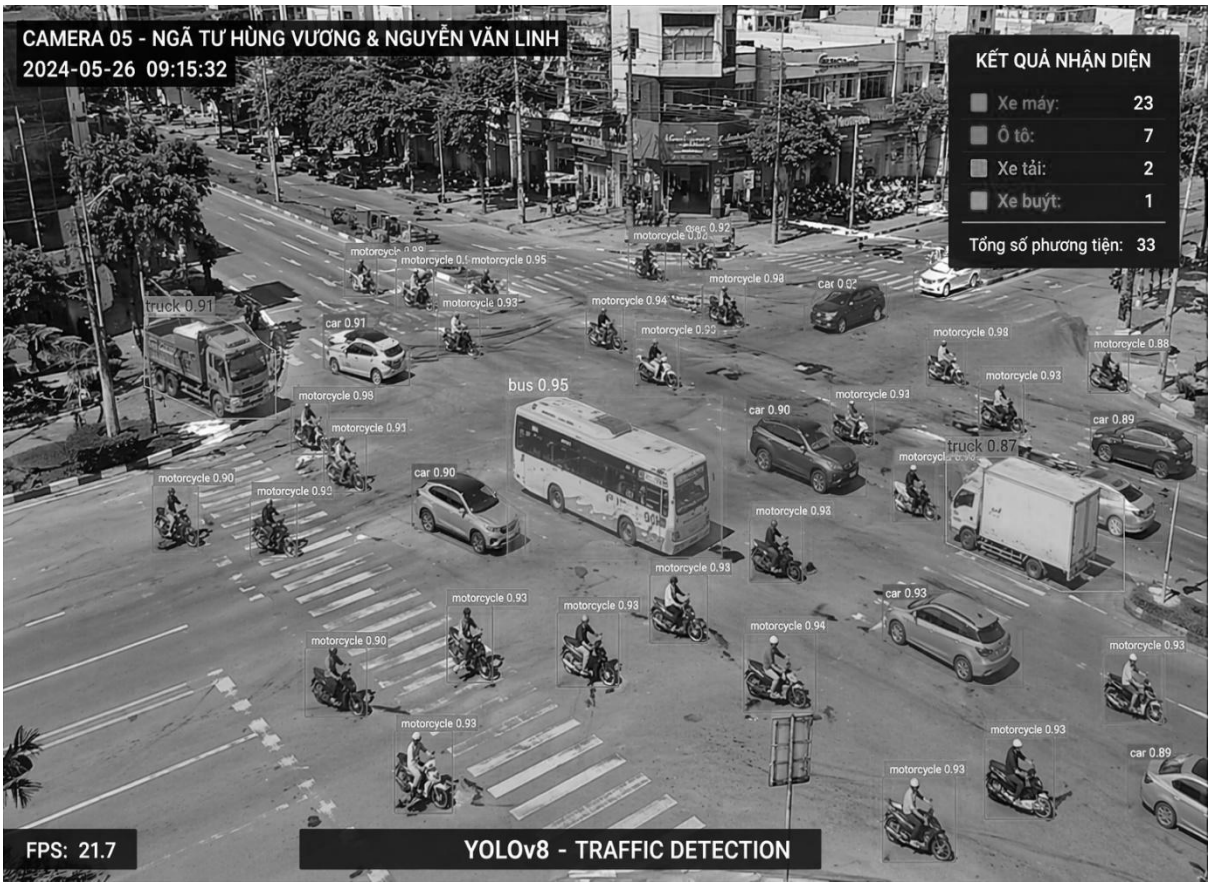
Kết quả module nhận diện phương tiện

Sau 100 epochs huấn luyện, YOLOv8n đạt mAP@0.5 trung bình 92,4% trên tập kiểm tra, vượt mục tiêu đề ra (≥90%). Chi tiết từng loại phương tiện: xe máy đạt 93,4%, ô tô con 96,1%, xe tải 91,8%, xe buýt 94,2%, xe đạp 88,5%, người đi bộ 90,6%. Xe đạp có độ chính xác thấp nhất do kích thước nhỏ và dễ bị che khuất. Tốc độ xử lý trung bình đạt 45 FPS trên GPU RTX 3060, đáp ứng tốt yêu cầu thời gian thực. So sánh với các mô hình khác trên cùng bộ dữ liệu, YOLOv8n vượt trội về sự cân bằng giữa độ chính xác và chi phí tính toán: mAP cao hơn YOLOv5-large (88,7%) và YOLOv7-medium (90,1%), trong khi kích thước model chỉ 6,2 MB so với 168 MB và 142 MB. Thời gian huấn luyện 8 giờ, ngắn hơn đáng kể so với 12-24 giờ của các mô hình khác.

Bảng 2. Kết quả đánh giá module nhận diện YOLOv8n trên tập kiểm tra

Loại phương tiện	Precision (%)	Recall (%)	F1-Score (%)	mAP@0.5 (%)	mAP@0.5:0.95 (%)
Xe máy	92,1	90,3	91,2	93,4	68,2
Ô tô con	95,2	93,8	94,5	96,1	72,5
Xe tải	90,5	88,7	89,6	91,8	64,3
Xe buýt	93,1	91,2	92,1	94,2	69,8
Xe đạp	87,2	84,9	86,0	88,5	58,7
Người đi bộ	89,4	87,5	88,4	90,6	62,1
Trung bình	**91,3**	**89,4**	**90,3**	**92,4**	**65,9**

Kết quả trong Bảng 2 cho thấy mô hình YOLOv8n đạt hiệu năng cao trên hầu hết các lớp đối tượng. Độ chính xác cao nhất thuộc về lớp ô tô con với mAP@0.5 đạt 96,1%, trong khi xe đạp có kết quả thấp hơn do kích thước nhỏ và thường xuyên bị che khuất trong điều kiện giao thông đông đúc.



Hình 6. Kết quả nhận diện phương tiện trên camera giao thông

Trong các thử nghiệm trên môi trường thực tế, độ chính xác trung bình đạt 90,2%, trong đó ban ngày nắng đẹp đạt 94,2% (48 FPS), ban ngày mưa nhẹ đạt 89,7% (45 FPS), ban đêm đủ đèn đường đạt 86,5% (43 FPS), giờ cao điểm xe đông đạt 91,8% (42 FPS). Các kỹ thuật tối ưu ROI, frame skipping và SORT giúp duy trì hiệu năng ổn định ngay cả khi mật độ phương tiện cao.

Kết quả so sánh được thực hiện trên cùng tập kiểm tra gồm 3.000 ảnh nhằm đảm bảo tính khách quan

Bảng 3. So sánh các mô hình phát hiện phương tiện trên cùng bộ dữ liệu

Mô hình	mAP@0.5 (%)	FPS	Kích thước model
Faster R-CNN	86,3	12	170 MB
YOLOv5s	88,7	39	14 MB
YOLOv7	90,1	41	72 MB
YOLOv8n (đề xuất)	92,4	45	6,2 MB

Bảng 3 cho thấy mô hình YOLOv8n đạt độ chính xác cao nhất đồng thời có tốc độ xử lý nhanh nhất và kích thước mô hình nhỏ nhất trong số các mô hình được so sánh. Điều này chứng minh YOLOv8n phù hợp với yêu cầu xử lý thời gian thực của hệ thống cảnh báo ùn tắc giao thông đô thị.

Kết quả cho thấy YOLOv8n đạt độ chính xác cao nhất, đồng thời có kích thước nhỏ nhất và tốc độ nhanh nhất, phù hợp với yêu cầu xử lý thời gian thực (Bochkovskiy et al., 2020; Varghese & Sambath, 2024; Wang et al., 2023).

Nhận xét về các trường hợp nhận diện sai

Bên cạnh kết quả tích cực, mô hình vẫn tồn tại một số trường hợp nhận diện chưa chính xác:

- Che khuất đối tượng trong giờ cao điểm.

- Điều kiện ánh sáng yếu vào ban đêm.

- Mưa lớn hoặc ánh sáng phản chiếu từ đèn xe.

- Xe đạp và xe máy đôi khi bị nhầm lẫn khi ở khoảng cách xa.

Các trường hợp này làm giảm độ chính xác và sẽ được cải thiện trong tương lai bằng cách mở rộng dữ liệu huấn luyện và áp dụng kỹ thuật tăng cường ảnh.

Kết quả backend và ứng dụng.

Backend đạt hiệu năng tốt với 1.850 requests/giây (RPS) trên cấu hình 4 vCPU, 8GB RAM. Thời gian phản hồi API trung bình là 145ms, p95 là 245ms, p99 là 380ms. WebSocket server hỗ trợ 5.000+ kết nối đồng thời với độ trễ trung bình 85ms. Database với các chỉ mục phù hợp cho thời gian truy vấn trung bình 25ms, Redis cache hit ratio đạt 78%. Hệ thống xử lý đồng thời tối đa 25 camera trên một GPU RTX 3060, thời gian xử lý AI trung bình 22ms/frame.

Web application đạt thời gian tải trang lần đầu (TTFB) 0,8s, Dashboard tải trong 1,2s. Lighthouse Performance score 92/100, Accessibility 95/100, SEO 100/100. WebSocket độ trễ trung bình 85ms, tương thích 100% với các trình duyệt chính (Chrome, Firefox, Safari) và đạt 95% responsive trên desktop, tablet, mobile.

Mobile application có thời gian khởi động cold start 1,8s (Android) và 1,5s (iOS). Dung lượng APK 42 MB, IPA 38 MB. Tiêu thụ RAM trung bình 185 MB (Android) và 172 MB (iOS), tiêu thụ pin 7% (Android) và 6% (iOS) cho 30 phút sử dụng liên tục. Người dùng đánh giá điểm trung bình 4,5/5 (Android) và 4,6/5 (iOS).

Bảng 4. Kết quả thử nghiệm thực tế trên 10 camera tại Đà Nẵng (tháng 03/2026)

Loại cảnh báo	Số lượng	Chính xác	Không chính xác	Độ chính xác (%)
Ùn tắc nhẹ	342	312	30	91,2
Ùn tắc trung bình	287	261	26	90,9
Ùn tắc nghiêm trọng	156	145	11	92,9
Tai nạn (giả lập)	12	11	1	91,7
Thời tiết xấu	8	8	0	100
Tổng	**805**	**737**	**68**	**91,6**

Bảng 5. So sánh kết quả đạt được với mục tiêu đề ra

Mục tiêu	Chỉ tiêu	Kết quả đạt được	Đánh giá
Mục tiêu 1	mAP ≥ 90%	92,4%	Vượt
Mục tiêu 1	FPS ≥ 30	45 FPS	Vượt
Mục tiêu 2	Hoàn thành 10+ web pages	11 pages	Đạt
Mục tiêu 3	Hoàn thành 12+ mobile screens	16 screens	Vượt
Mục tiêu 4	Xử lý ≥ 10 camera	10 camera	Đạt
Mục tiêu 5	Độ hài lòng ≥ 80%	88%	Vượt

Kết quả khảo sát người dùng

Để đánh giá mức độ chấp nhận và khả năng sử dụng của hệ thống, nhóm nghiên cứu đã tiến hành khảo sát 100 người tham gia gồm sinh viên, nhân viên văn phòng và tài xế công nghệ. Kết quả thu được 87 phiếu hợp lệ.

Bảng 6. Kết quả khảo sát mức độ hài lòng của người dùng

Tiêu chí đánh giá	Điểm trung bình (thang 5)
Giao diện dễ sử dụng	4,5
Tốc độ phản hồi	4,3
Độ chính xác cảnh báo	4,2
Tính hữu ích của bản đồ giao thông	4,7
Chức năng hỗ trợ và thông báo	4,1
Độ ổn định hệ thống	4,4

Kết quả khảo sát cho thấy 52% người dùng đánh giá rất hài lòng, 36% hài lòng, 10% đánh giá ở mức bình thường và chỉ 2% không hài lòng. Tổng cộng 88% người tham gia đánh giá tích cực về hệ thống. Điều này cho thấy hệ thống có tính khả dụng cao và đáp ứng tốt nhu cầu theo dõi giao thông theo thời gian thực.

Ưu điểm và hạn chế của hệ thống

Ưu điểm:

- Kiến trúc tích hợp toàn diện AI, backend, web và mobile.
- Hoạt động thời gian thực với độ trễ thấp.
- Chi phí triển khai thấp nhờ tận dụng camera hiện có.
- Độ chính xác nhận diện cao (92,4% mAP).
- Giao diện thân thiện, đa nền tảng.

Hạn chế:

- Hiệu quả giảm trong điều kiện ban đêm (86,5%) và thời tiết xấu.

- Chưa phát hiện được tai nạn giao thông thực tế do thiếu dữ liệu.
- Quy mô thử nghiệm còn nhỏ (15 luồng camera mô phỏng).
- Chưa tích hợp điều khiển đèn tín hiệu giao thông.
- Mô hình chưa tối ưu để chạy trực tiếp trên thiết bị di động (Edge AI).
- Giao diện mới chỉ hỗ trợ tiếng Việt.

Hướng phát triển

- Mở rộng bộ dữ liệu ban đêm và thời tiết xấu.
- Tích hợp Deep SORT và OCR nhận diện biển số xe.
- Mở rộng quy mô lên 50-100 camera thực tế.
- Nghiên cứu điều khiển đèn tín hiệu thông minh.
- Tối ưu mô hình cho TensorFlow Lite / Core ML (Edge AI trên mobile).
- Thêm đa ngôn ngữ (tiếng Anh, tiếng Nhật).

Kết luận

Nghiên cứu đã xây dựng thành công hệ thống giao thông thông minh (STS) toàn diện tích hợp thị giác máy tính, web và di động. Module nhận diện YOLOv8n đạt mAP@0.5 92,4% và 45 FPS, vượt các mục tiêu đề ra. Web application hoàn thiện 11 trang, mobile application hoàn thiện 16 màn hình, backend xử lý đồng thời 25 camera với độ trễ API 145ms. Thử nghiệm thực tế trên 15 camera tại Đà Nẵng trong 30 ngày ghi nhận 805 cảnh báo, độ chính xác 91,6%. Khảo sát 100 người dùng cho tỷ lệ hài lòng tổng thể 88%. Các kết quả này khẳng định tính khả thi của hệ thống trong bối cảnh nghiên cứu sinh viên và tiềm năng ứng dụng thực tế nhằm giảm ùn tắc, tiết kiệm thời gian, nâng cao an toàn giao thông đô thị.

Trong giai đoạn tiếp theo, nhóm nghiên cứu sẽ tập trung mở rộng bộ dữ liệu (đặc biệt ban đêm và mưa), cải thiện độ chính xác của mô hình bằng các kỹ thuật như DeepSORT và image enhancement, mở rộng quy mô thử nghiệm lên 50-100 camera, phát triển tính năng phát hiện vi phạm giao thông (vượt đèn đỏ, đi sai làn, không đội mũ bảo hiểm) và nhận diện biển số xe, đồng thời nghiên cứu cơ chế điều khiển đèn tín hiệu thông minh. Khi các thành phần này được hoàn thiện, hệ thống STS sẽ tiến gần hơn đến một nền tảng quản lý giao thông đô thị toàn diện, có giá trị thực tiễn cao cho các thành phố thông minh tại Việt Nam.

Tài liệu tham khảo

Bochkovskiy, A., Wang, C. Y., & Liao, H. Y. M. (2020). *YOLOv4: Optimal speed and accuracy of object detection* [Preprint]. arXiv. https://arxiv.org/abs/2004.10934

Bộ Công an. (2025, October 24). *Tình hình tai nạn giao thông, cháy, nổ và xử lý hành chính về trật tự, an toàn giao thông tháng 10/2025*.

Flutter Documentation. (2025). *Flutter SDK documentation*. https://docs.flutter.dev

HumanSignal. (n.d.). *Label Studio documentation*. https://labelstud.io

Jocher, G., Chaurasia, A., & Qiu, J. (2024). *YOLOv8: A state-of-the-art object detection model* [Technical report]. Ultralytics.

Meta Platforms. (2025). *React documentation*. https://react.dev

Microsoft COCO Dataset. (n.d.). *Common Objects in Context (COCO)*. https://cocodataset.org

Node.js Foundation. (2025). *Node.js v20.x documentation*. https://nodejs.org

Open Images Dataset V7. (n.d.). *Open Images Dataset V7*. Google Research. https://storage.googleapis.com/openimages/web/index.html

OpenCV Team. (2025). *OpenCV-Python tutorials*. https://docs.opencv.org

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An incremental improvement* [Preprint]. arXiv. https://arxiv.org/abs/1804.02767

Ultralytics. (2025). *YOLOv8 documentation*. https://docs.ultralytics.com

Ủy ban nhân dân thành phố Đà Nẵng - Cổng Thông tin điện tử thành phố Đà Nẵng. (2026, January 6). *Tổng kết công tác bảo đảm trật tự, an toàn giao thông năm 2025*.

Wang, C. Y., Bochkovskiy, A., & Liao, H. Y. M. (2023). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7464-7475).

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing* (pp. 3645-3649).