TÊN ĐỀ TÀI: HỆ THỐNG AI PHÁT HIỆN VI PHẠM GIAO THÔNG TỪ CAMERA

Nguyễn Thanh Thiệt, 225990, thietnguyenthanh300@gmail.com

Hồng Phước Thịnh, 224366, phuocthinh232006@gmail.com

Dương Lý Cử, 223650, duonglycu99@gmail.com

**Phụ trách chính: Nguyễn Thanh Thiệt*

TÓM TẮT

Đề tài nhằm phát triển một hệ thống phát hiện vi phạm giao thông theo thời gian thực, tập trung vào hai hành vi phổ biến: vượt đèn đỏ và không đội mũ bảo hiểm. Phương pháp được sử dụng bao gồm mô hình phát hiện đối tượng YOLOv11 kết hợp thuật toán theo dõi DeepSORT, cùng kỹ thuật xử lý ảnh để xác định tín hiệu đèn giao thông trong không gian màu HSV. Kết quả cho thấy mô hình YOLOv11 đạt hiệu suất cao khi nhận diện các phương tiện chính như ô tô và xe máy, với mAP@0.5 trung bình đạt 0.866. Mô hình phát hiện mũ bảo hiểm đạt F1-score trung bình 0.71. Hệ thống hoạt động ổn định trong môi trường thực tế và có khả năng phân tích hành vi vi phạm dựa trên chuyển động và đặc trưng trực quan. Nghiên cứu cho thấy tiềm năng ứng dụng của thị giác máy tính trong giám sát giao thông tự động, góp phần nâng cao hiệu quả quản lý và đảm bảo an toàn cho người tham gia giao thông.

1. ĐẶT VẤN ĐỀ

1.1 Bối cảnh

Sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI), đặc biệt trong lĩnh vực thị giác máy tính (Computer Vision), đã mở ra những tiềm năng to lớn trong việc nâng cao hiệu quả quản lý đô thị, trong đó giao thông là một lĩnh vực trọng tâm. Tại Việt Nam, với đặc điểm giao thông phức tạp ở các đô thị lớn như Hà Nội, TP. Hồ Chí Minh và Đà Nẵng, các hành vi vi phạm như vượt đèn đỏ và không đội mũ bảo hiểm khi điều khiển xe máy là những nguyên nhân hàng đầu dẫn đến tai nạn giao thông nghiêm trọng.

Theo báo cáo của Ủy ban An toàn Giao thông Quốc gia, mỗi năm nước ta ghi nhận hàng nghìn vụ tai nạn liên quan đến các hành vi trên, gây thiệt hại nghiêm trọng về người và tài sản. Dù hệ thống camera giám sát và xử phạt nguội đã được triển khai tại nhiều đô thị, nhưng hiệu quả thực tế còn hạn chế do một số nguyên nhân như độ chính xác thấp,

thiếu khả năng xử lý thời gian thực, và sự phụ thuộc lớn vào con người trong quá trình phân tích dữ liệu.

Trong bối cảnh đó, các công nghệ thị giác máy tính tiên tiến như YOLO và DeepSort mang đến cơ hội xây dựng một hệ thống giám sát thông minh, có khả năng tự động phát hiện hành vi vi phạm giao thông tại thời điểm xảy ra, đồng thời hỗ trợ lực lượng chức năng trong công tác quản lý giao thông một cách hiệu quả và kịp thời.

1.2 Thực trạng và tầm quan trọng của nghiên cứu

Hiện nay, tại Việt Nam, các hệ thống giám sát giao thông chủ yếu vẫn dựa trên camera quan sát truyền thống, kết hợp với phương pháp phân tích hình ảnh thủ công hoặc bán tự động. Tuy đã có đóng góp nhất định trong việc ghi nhận vi phạm giao thông, nhưng những hệ thống này vẫn bộc lộ nhiều hạn chế đáng kể khi áp dụng trong môi trường giao thông đô thị phức tạp:

Độ chính xác còn thấp: Việc nhận diện các hành vi vi phạm như vượt đèn đỏ hoặc không đội mũ bảo hiểm thường bị ảnh hưởng bởi các yếu tố như góc quay không tối ưu, điều kiện ánh sáng yếu, hình ảnh bị mờ, hoặc đối tượng bị che khuất, dẫn đến kết quả nhận diện không đáng tin cậy.

Thiếu khả năng xử lý theo thời gian thực: Phần lớn các hệ thống hiện tại chỉ thu thập dữ liệu và xử lý sau khi vi phạm đã xảy ra (xử phạt nguội), làm giảm hiệu quả răn đe cũng như khó có khả năng can thiệp kịp thời để ngăn chặn vi phạm ngay tại hiện trường.

Không phù hợp với môi trường giao thông phức tạp: Mật độ phương tiện cao, loại hình phương tiện đa dạng (ô tô, xe máy, xe đạp...), cùng với hành vi tham gia giao thông không đồng nhất tại các thành phố lớn như TP. Hồ Chí Minh, Hà Nội, khiến việc áp dụng các mô hình giám sát truyền thống trở nên thiếu hiệu quả.

Trong khi đó, các nghiên cứu ứng dụng thị giác máy tính vào giám sát giao thông tại Việt Nam vẫn còn hạn chế, chủ yếu mới chỉ dừng lại ở quy mô thử nghiệm nhỏ, trong môi trường mô phỏng hoặc điều kiện đơn giản. Chưa có một hệ thống nào có khả năng tích hợp đồng bộ và xử lý đồng thời nhiều hành vi vi phạm theo thời gian thực, đặc biệt là trong điều kiện thực tế phức tạp của giao thông Việt Nam.

Chính vì vậy, việc phát triển một hệ thống giám sát giao thông thông minh, ứng dụng các công nghệ thị giác máy tính tiên tiến như YOLO (dùng để phát hiện đối tượng) và DeepSort (dùng để theo dõi đối tượng), có thể hoạt động theo thời gian thực, là một nhu cầu cấp thiết. Hệ thống này không chỉ góp phần nâng cao hiệu quả phát hiện và xử lý vi phạm, giảm thiểu tai nạn giao thông, mà còn tạo tiền đề cho việc tích hợp dữ liệu vào các nền tảng giao thông thông minh (Intelligent Transportation Systems – ITS) trong tương lai. Qua đó, nghiên cứu mang lại giá trị thực tiễn cao, đồng thời giải quyết khoảng trống còn tồn tại trong lĩnh vực ứng dụng AI cho giao thông tại Việt Nam.

1.3 Tổng quan tài liệu

Trong những năm gần đây, các mô hình thị giác máy tính như YOLO (You Only Look Once) và DeepSort đã được áp dụng rộng rãi trong lĩnh vực giám sát giao thông, đặc biệt trong các bài toán như nhận diện phương tiện, xác định người điều khiển hoặc phát hiện hành vi không đội mũ bảo hiểm. Các phiên bản cải tiến như YOLOv3 và YOLOv5 đã chứng minh khả năng nhận diện chính xác cao khi triển khai trong môi trường giao thông tại các đô thị lớn như Thành phố Hồ Chí Minh và Hà Nội, đặc biệt trong điều kiện ánh sáng và góc quay thuận lợi. Bên cạnh đó, DeepSort với khả năng theo dõi đối tượng theo thời gian liên tục đã góp phần nâng cao hiệu quả trong việc phân tích luồng di chuyển và hành vi phương tiện tại các giao lộ.

Tuy nhiên, các nghiên cứu ứng dụng tại Việt Nam vẫn chủ yếu tập trung vào từng khía cạnh riêng lẻ, chẳng hạn chỉ phát hiện hành vi không đội mũ bảo hiểm hoặc chỉ giám sát tình huống vượt đèn đỏ. Việc thiếu một hệ thống tích hợp có khả năng xử lý đồng thời nhiều loại vi phạm làm hạn chế tính hiệu quả và khả năng triển khai thực tế. Ngoài ra, nhiều nghiên cứu quốc tế tuy đạt được kết quả ấn tượng nhưng thường được thử nghiệm trong môi trường giao thông đã được chuẩn hóa, chưa phản ánh đầy đủ những đặc thù như mật độ phương tiện cao, tính đa dạng về phương tiện và hành vi giao thông phức tạp như tại Việt Nam.

Từ thực tế đó, đặt ra yêu cầu cần thiết phải phát triển một hệ thống giám sát giao thông thông minh, có khả năng xử lý theo thời gian thực, phù hợp với điều kiện hạ tầng tại Việt Nam, và có thể triển khai trên các thiết bị phần cứng nhúng tại các nút giao thông. Hệ thống này không chỉ giúp nâng cao năng lực giám sát mà còn mở ra hướng tiếp cận hiệu quả hơn trong việc ứng dụng trí tuệ nhân tạo vào quản lý giao thông hiện đại.

1.4 Xác định vấn đề nghiên cứu

Trên cơ sở phân tích thực trạng hiện nay và những khoảng trống trong lĩnh vực giám sát giao thông bằng công nghệ thị giác máy tính, nghiên cứu này tập trung vào việc giải quyết một câu hỏi trọng tâm như sau:

Làm thế nào để xây dựng một hệ thống phát hiện vi phạm giao thông theo thời gian thực, tích hợp các công nghệ thị giác máy tính hiện đại như YOLO và DeepSort, nhằm nhận diện hành vi vượt đèn đỏ và không đội mũ bảo hiểm, đồng thời đảm bảo khả năng triển khai hiệu quả trong điều kiện giao thông phức tạp tại Việt Nam?

Câu hỏi này không chỉ đặt ra yêu cầu về mặt kỹ thuật, bao gồm việc lựa chọn mô hình học sâu phù hợp, tối ưu hoá quy trình xử lý ảnh và theo dõi đối tượng, mà còn liên quan đến các yếu tố thực tế trong triển khai và vận hành hệ thống. Cụ thể, nghiên cứu cần giải quyết một số vấn đề chính sau:

Làm thế nào để hệ thống duy trì được độ chính xác cao trong môi trường có nhiều yếu tố gây nhiễu như điều kiện ánh sáng thay đổi, phương tiện bị che khuất hoặc chồng lấn, cũng như hành vi giao thông thiếu tuân thủ quy chuẩn?

Hệ thống có thể xử lý và phản hồi theo thời gian thực hay không, từ đó đáp ứng được yêu cầu giám sát liên tục tại các giao lộ đông đúc?

Giải pháp đưa ra có đủ tính linh hoạt và khả năng tích hợp vào hạ tầng hiện có của các đô thị, chẳng hạn như hệ thống camera quan sát, trung tâm điều hành giao thông, hay các nền tảng dữ liệu đô thị khác?

Việc triển khai hệ thống trên phần cứng nhúng có đáp ứng được yêu cầu về hiệu năng, chi phí và khả năng mở rộng hay không?

Qua việc trả lời những câu hỏi này, nghiên cứu hướng đến xây dựng một hệ thống giám sát giao thông thông minh, hoạt động hiệu quả trong môi trường thực tế của Việt Nam, từ đó góp phần nâng cao năng lực xử lý vi phạm, hỗ trợ cơ quan chức năng và tiến tới phát triển đô thị thông minh trong tương lai.

1.5 Tính mới của nghiên cứu

Nghiên cứu này mang tính đột phá ở chỗ không chỉ áp dụng các công nghệ thị giác máy tính tiên tiến mà còn kết hợp chúng trong một hệ thống tích hợp hoàn chỉnh, đáp ứng các yêu cầu thực tế của giao thông đô thị Việt Nam.

Trước hết, điểm mới nổi bật của nghiên cứu là việc kết hợp đồng bộ giữa YOLO và DeepSort trong cùng một hệ thống để thực hiện đồng thời hai nhiệm vụ: phát hiện vi phạm và theo dõi đối tượng vi phạm. Trong khi YOLO đóng vai trò nhận diện các đối tượng và hành vi như người điều khiển phương tiện, tình trạng đội mũ bảo hiểm hay trạng thái đèn tín hiệu giao thông, thì DeepSort đảm nhiệm việc theo dõi liên tục các đối tượng qua từng khung hình, giúp xác định chính xác quá trình vi phạm (chẳng hạn hành vi vượt đèn đỏ diễn ra như thế nào, từ vị trí nào và kết thúc ở đâu). Việc xử lý đồng thời cả hai hành vi vi phạm phổ biến nhất hiện nay – vượt đèn đỏ và không đội mũ bảo hiểm – trong một hệ thống thống nhất là một điểm mới mang tính thực tiễn cao, chưa được khai thác đầy đủ trong các nghiên cứu trước đây.

Bên cạnh đó, hệ thống được thiết kế tối ưu để phù hợp với điều kiện giao thông đặc thù tại Việt Nam, vốn có nhiều thách thức như mật độ phương tiện cao, không gian quan sát hạn chế, điều kiện ánh sáng thay đổi liên tục (sáng – tối – mưa – nắng), cũng như sự đa dạng và khó kiểm soát trong hành vi tham gia giao thông của người dân. Việc xử lý hình ảnh và dữ liệu trong thời gian thực đòi hỏi hệ thống không chỉ chính xác mà còn phải ổn định và hiệu quả trong môi trường khắc nghiệt, điều mà ít nghiên cứu trước đây đạt được.

1.6 Mục tiêu nghiên cứu

Nghiên cứu hướng đến việc phát triển một giải pháp toàn diện nhằm nâng cao hiệu quả giám sát và xử lý vi phạm giao thông trong bối cảnh thực tế tại Việt Nam. Cụ thể, các mục tiêu chính của nghiên cứu bao gồm:

Thứ nhất, nghiên cứu tập trung vào việc thiết kế và xây dựng một hệ thống phát hiện vi phạm giao thông hoạt động theo thời gian thực, ứng dụng các mô hình thị giác máy tính tiên tiến như YOLO và DeepSort. Mục tiêu là tạo ra một hệ thống có khả năng

nhận diện chính xác hai hành vi vi phạm phổ biến và nguy hiểm nhất hiện nay là vượt đèn đỏ và không đội mũ bảo hiểm khi điều khiển xe máy. Hệ thống cần đảm bảo độ chính xác cao trong điều kiện môi trường thực tế, bao gồm ánh sáng thay đổi, mật độ phương tiện cao và các yếu tố gây nhiễu thường gặp trong đô thị Việt Nam.

Thứ hai, nghiên cứu đặt mục tiêu đánh giá hiệu quả thực tiễn của hệ thống thông qua các thử nghiệm thực địa tại môi trường giao thông thật. Thông qua các phương pháp phân tích định lượng, nghiên cứu sẽ xác định các chỉ số như độ chính xác (accuracy), tốc độ xử lý (latency), khả năng theo dõi (tracking consistency), và tỷ lệ phát hiện đúng (detection rate). Ngoài ra, nghiên cứu cũng xem xét mức độ khả thi trong triển khai thực tế và tiềm năng ứng dụng rộng rãi, từ đó đưa ra các đề xuất cải tiến và hướng phát triển trong tương lai, góp phần vào việc xây dựng hệ thống giao thông thông minh và nâng cao mức độ an toàn giao thông tại Việt Nam.

2. CÁC NGHIÊN CỨU LIÊN QUAN

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. ***arXiv preprint arXiv:1804.02767:*** Redmon và Farhadi (2018) giới thiệu YOLOv3, một mô hình phát hiện đối tượng thời gian thực được thiết kế nhằm cải thiện tốc độ và độ chính xác so với các phiên bản trước đó. Mô hình này sử dụng kiến trúc mạng nơ-ron tích chập (CNN) kết hợp với mạng kim tự tháp đặc trưng (Feature Pyramid Network – FPN) để phát hiện vật thể ở nhiều kích thước khác nhau, đồng thời áp dụng anchor boxes và hồi quy logistic để dự đoán các bounding box. Hệ thống được huấn luyện trên tập dữ liệu COCO (Common Objects in Context), bao gồm 80 lớp đối tượng phổ biến như ô tô, xe máy, xe tải và người đi bộ. Kết quả cho thấy YOLOv3 đạt được mAP (mean Average Precision) 33.0 trên COCO, với tốc độ xử lý khoảng 22 mili giây mỗi khung hình, thể hiện khả năng cân bằng giữa hiệu năng và độ chính xác. Mô hình đã đặt nền móng cho các ứng dụng giám sát giao thông và xe tự hành trong môi trường thực tế. Tuy nhiên, YOLOv3 vẫn còn một số hạn chế như hiệu suất giảm khi xử lý các vật thể nhỏ, chồng lấp hoặc trong điều kiện giao thông phức tạp, và thường cần tinh chỉnh thêm để đạt hiệu quả tốt nhất trong từng bối cảnh cụ thể.

Liu, W., et al. (2016). SSD: Single Shot MultiBox Detector. ***ECCV:*** Liu et al. (2016) đề xuất SSD (Single Shot MultiBox Detector) – một mô hình phát hiện đối tượng hiệu quả, cho phép nhận diện phương tiện giao thông với độ chính xác cao và tốc độ xử lý nhanh. Phương pháp này tận dụng các lớp đặc trưng tích chập (convolutional feature maps) ở nhiều tỷ lệ để đồng thời dự đoán bounding boxes và lớp đối tượng trong một lần duy nhất (single shot), từ đó cải thiện tốc độ mà vẫn duy trì được độ chính xác. SSD được huấn luyện trên các tập dữ liệu PASCAL VOC và COCO, tập trung vào các lớp phương tiện phổ biến như xe hơi, xe buýt và xe đạp. Kết quả cho thấy mô hình đạt mAP 74.3% trên PASCAL VOC và 46.5% trên COCO, với tốc độ 59 FPS khi chạy trên GPU, giúp cân bằng tốt giữa hiệu năng và độ chính xác. SSD đã được ứng dụng rộng rãi trong

các hệ thống camera giao thông thông minh. Tuy nhiên, hạn chế của phương pháp là khả năng phát hiện các vật thể nhỏ chưa cao so với các mô hình như Faster R-CNN.

Geiger, A., et al. (2012). Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite. ***CVPR:*** Geiger et al. (2012) giới thiệu bộ dữ liệu KITTI – một benchmark toàn diện dành cho các nghiên cứu về xe tự hành và nhận diện phương tiện giao thông trong điều kiện thực tế. Nhóm tác giả đã thu thập dữ liệu từ nhiều cảm biến khác nhau như camera, LiDAR và GPS, được gắn trên xe di chuyển qua các khu vực đô thị và đường cao tốc. Bộ dữ liệu KITTI gồm 7481 hình ảnh huấn luyện với hơn 80.000 đối tượng được gán nhãn, bao gồm các phương tiện như xe hơi, người đi bộ và xe đạp. KITTI đã trở thành tiêu chuẩn đánh giá phổ biến cho nhiều mô hình như Faster R-CNN (với độ chính xác AP đạt 70% cho lớp "car"). Tuy nhiên, một điểm hạn chế của bộ dữ liệu này là nguồn thu thập chủ yếu tại Châu Âu, nên chưa bao phủ được sự đa dạng về địa lý và văn hóa giao thông toàn cầu.

Nguyen, T., Le, M., Pham, V., & Tran, H. (2020). Comparative Analysis of YOLOv3 and YOLOv5 for Traffic Violation Detection in Vietnam: Nguyen, Le, Pham và Tran (2020) tiến hành so sánh hiệu suất giữa hai mô hình YOLOv3 và YOLOv5 trong việc phát hiện vi phạm giao thông tại Việt Nam. Nghiên cứu sử dụng video thực tế từ camera giao thông tại ba thành phố lớn là TP. Hồ Chí Minh, Hà Nội và Đà Nẵng để đánh giá hiệu quả qua các chỉ số như F1-score, độ trễ xử lý và tốc độ khung hình (FPS). Kết quả cho thấy YOLOv5 vượt trội với F1-score đạt 93% và độ trễ dưới 0.2 giây trong việc phát hiện hành vi không đội mũ bảo hiểm, so với YOLOv3 có F1-score là 92%. Điều này khẳng định tiềm năng ứng dụng YOLOv5 trong các hệ thống giám sát giao thông thời gian thực. Tuy nhiên, nghiên cứu cũng chỉ ra rằng mô hình gặp khó khăn khi áp dụng trong điều kiện thời tiết xấu hoặc ở các khu vực ngoài thành phố lớn do hạn chế về sự đa dạng trong tập dữ liệu huấn luyện.

Bharati, A., Singh, R., & Sharma, P. (2021). Intelligent Traffic Control System using YOLOv4 and XGBoost: Bharati, Singh và Sharma (2021) đề xuất một hệ thống điều khiển giao thông thông minh nhằm tối ưu hóa thời gian đèn tín hiệu bằng cách đếm và phân loại phương tiện. Phương pháp tiếp cận bao gồm việc sử dụng mô hình YOLOv4 để nhận diện và phân loại phương tiện từ video, kết hợp với XGBoost để dự đoán thời gian đèn xanh, từ đó giảm thiểu tình trạng ùn tắc giao thông. Tập dữ liệu sử dụng gồm 5000 ảnh thu thập từ camera giao thông và các nguồn dữ liệu mở. Kết quả cho thấy hệ thống đạt mAP 63.4% trên Pascal VOC, độ chính xác phân loại phương tiện là 85% và giúp giảm 15% thời gian chờ tại giao lộ. Nghiên cứu này góp phần nâng cao hiệu quả quản lý tại các nút giao thông phức tạp. Tuy nhiên, hệ thống chưa hoạt động tốt trong điều kiện ánh sáng yếu, không hỗ trợ nhận diện biển số và đòi hỏi phần cứng tính toán cao.

Raj, D., Karthik, K., & Kumar, R. (2021). Detection of Traffic Violations using ResNet-50 and RGB Analysis: Raj, Karthik và Kumar (2021) phát triển một hệ thống phát hiện vi phạm giao thông tập trung vào hai hành vi chính: không đội mũ bảo hiểm và vượt đèn đỏ tại các giao lộ. Nghiên cứu sử dụng mô hình ResNet-50 kết hợp với phân tích màu sắc RGB để nhận diện các hành vi vi phạm trong video giao thông. Tập dữ liệu được thu thập từ nhiều tuyến đường tại các thành phố lớn ở Ấn Độ. Hệ thống đạt độ chính xác 95% trong việc phát hiện hành vi không đội mũ bảo hiểm và 91% đối với hành vi vượt đèn đỏ, cho thấy hiệu quả cao trong môi trường kiểm soát giao thông đô thị. Tuy nhiên, mô hình chưa xử lý tốt các tình huống che khuất vật thể và còn gặp khó khăn khi áp dụng tại Việt Nam do sự khác biệt trong điều kiện môi trường và hành vi tham gia giao thông.

Nguyen, L., Tran, H., & Pham, D. (2024). A Hybrid YOLOv5 and Faster R-CNN Approach for Traffic Violation Detection at Intersections. Proceedings of the International Conference on Machine Learning and Artificial Intelligence (MLMI 2024): Nguyen, Tran và Pham (2024) giới thiệu một hệ thống giám sát giao thông thông minh dựa trên sự kết hợp giữa mô hình một giai đoạn (YOLOv5) và mô hình hai giai đoạn (Faster R-CNN) nhằm phát hiện vi phạm tại các giao lộ như vượt đèn đỏ và đi sai làn đường. Trong hệ thống này, YOLOv5 đảm nhiệm vai trò phát hiện phương tiện trong thời gian thực, trong khi Faster R-CNN xác định chính xác khu vực vi phạm, tất cả được tích hợp với hệ thống camera giám sát hiện có. Tập dữ liệu gồm các video giám sát đã được gán nhãn vi phạm giao thông. Kết quả thực nghiệm cho thấy hệ thống đạt độ chính xác 100% đối với hành vi vượt đèn đỏ và 90% cho hành vi đi sai làn, khẳng định tiềm năng của cách tiếp cận kết hợp này trong việc nâng cao hiệu quả giám sát. Tuy nhiên, do sử dụng đồng thời hai mô hình nặng, hệ thống yêu cầu khả năng tính toán cao và chưa được kiểm chứng hiệu quả trên dữ liệu đến từ nhiều quốc gia khác nhau.

Alam, S., Rahman, T., & Chowdhury, M. (2025). Real-Time Traffic Monitoring at Intersections Using Enhanced YOLOv5 with Fisheye Cameras: Alam, Rahman và Chowdhury (2025) đã đề xuất một phương pháp giám sát giao thông thời gian thực tại các giao lộ sử dụng camera mắt cá, nhằm giải quyết những thách thức như ánh sáng chói, bóng đổ và biến dạng hình ảnh. Nhóm tác giả cải tiến mô hình YOLOv5 bằng cách tích hợp thêm một bộ phân loại CNN nhẹ, được thiết kế phù hợp cho điều kiện chiếu sáng ngày và đêm. Đồng thời, kỹ thuật tăng cường dữ liệu đặc biệt cũng được áp dụng để nâng cao hiệu quả nhận diện trong các tình huống khó. Tập dữ liệu bao gồm video thực tế thu thập từ các camera mắt cá được lắp đặt tại giao lộ. Kết quả cho thấy mô hình cải tiến đã nâng mAP@0.5 lên 13.7% so với YOLOv5 gốc khi làm việc trong điều kiện ánh sáng phức tạp. Công trình này đóng góp vào việc cải thiện độ chính xác và tính khả dụng trong môi trường thực tế khắc nghiệt của các hệ thống giám sát giao thông. Tuy nhiên, hệ thống yêu cầu hiệu chỉnh tùy theo từng loại camera mắt cá và cần dữ liệu được gán nhãn kỹ lưỡng cho từng điều kiện ánh sáng khác nhau.

Wang, X., et al. (2021). Transformer-based Traffic Vehicle Detection. ***IEEE Transactions on Intelligent Transportation Systems:*** Wang và cộng sự (2021) nghiên cứu ứng dụng kiến trúc Transformer – vốn phổ biến trong xử lý ngôn ngữ tự nhiên – vào bài toán phát hiện phương tiện giao thông. Họ sử dụng Vision Transformer (ViT) kết hợp với các bộ lọc không gian để xử lý các khung hình trong bối cảnh giao thông phức tạp. Tập dữ liệu sử dụng là UA-DETRAC, bao gồm 140.000 khung hình với 8.250 phương tiện được gán nhãn. Mô hình đạt mAP 89.2%, cao hơn đáng kể so với YOLOv4 (82.1%) trên cùng tập dữ liệu, cho thấy tiềm năng vượt trội của Transformer trong thị giác máy tính giao thông. Thành tựu này mở ra hướng nghiên cứu mới trong ứng dụng Transformer cho các hệ thống giao thông thông minh. Tuy nhiên, một hạn chế đáng kể là nhu cầu tài nguyên tính toán lớn, khiến việc triển khai mô hình trên các thiết bị biên trở nên khó khăn.

Zhang, Y., et al. (2022). Hybrid CNN-Transformer for Efficient Traffic Object Detection. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR): Zhang và cộng sự (2022) đề xuất một mô hình lai giữa CNN và Transformer nhằm cân bằng giữa độ chính xác và hiệu suất tính toán trong nhận diện phương tiện giao thông thời gian thực. Cụ thể, họ sử dụng EfficientNet làm backbone CNN kết hợp với Transformer encoder tương tự như Vision Transformer (ViT), cùng với cơ chế attention đa tầng để tăng cường tập trung vào các vùng chứa phương tiện. Ngoài ra, kỹ thuật knowledge distillation cũng được áp dụng để giảm số lượng tham số của mô hình. Tập dữ liệu bao gồm BDD100K (100.000 khung hình video) và 50.000 ảnh được thu thập từ hệ thống camera giao thông tại Hàng Châu. Mô hình đạt 91.3% mAP trên BDD100K, tốc độ xử lý 38 FPS trên RTX 2080 Ti và giảm đến 60% số tham số so với các mô hình Transformer thuần túy. Đây là một giải pháp hiệu quả cho các hệ thống giám sát giao thông thời gian thực, đặc biệt có thể triển khai trên các thiết bị biên như Jetson Xavier NX. Tuy vậy, mô hình vẫn gặp khó khăn khi hoạt động trong điều kiện thời tiết xấu và yêu cầu giai đoạn tiền huấn luyện khá phức tạp.

Các nghiên cứu này cho thấy những tiến bộ trong việc áp dụng công nghệ thị giác máy tính vào giám sát giao thông, nhưng cũng chỉ ra các vấn đề còn tồn tại, đặc biệt là trong việc phát triển các mô hình có thể hoạt động hiệu quả trong các điều kiện giao thông phức tạp tại Việt Nam. Việc giải quyết các vấn đề như đa dạng hóa dữ liệu huấn luyện, giảm độ trễ, và khả năng triển khai trong các khu vực thiếu cơ sở hạ tầng mạng vẫn là thách thức lớn cần được nghiên cứu thêm.

3. PHƯƠNG PHÁP NGHIÊN CỨU

3.1. Thiết kế nghiên cứu

Nghiên cứu được thiết kế theo hướng ứng dụng thị giác máy tính kết hợp trí tuệ nhân tạo để tự động phát hiện hành vi vi phạm giao thông từ dữ liệu video ghi nhận tại các giao lộ ngã ba, ngã tư. Quy trình tổng thể bao gồm: nhận diện đối tượng phương tiện và người điều khiển phương tiện giao thông, xác định trạng thái đèn giao thông, theo dõi chuyển động đối tượng, và phát hiện các hành vi vi phạm cụ thể như vượt đèn đỏ và không đội nón bảo hiểm.

Quy trình nghiên cứu bao gồm các bước chính:

1. Thu thập video giao thông thực tế tại các ngã tư ở Việt Nam.
2. Sử dụng mô hình học sâu YOLOv11 để phát hiện xe hơi, xe máy, xe tải, xe buýt và trạng thái đội mũ bảo hiểm của người điều khiển xe máy.
3. Theo dõi chuyển động của phương tiện bằng thuật toán DeepSort để xác định vi phạm vượt đèn đỏ.
4. Phân tích màu đèn giao thông bằng không gian màu HSV, kết hợp cơ chế ổn định để tăng độ chính xác.
5. Hiển thị kết quả trực quan trên video, bao gồm bounding box cho các phương tiện, nhãn vi phạm ("Vượt đèn đỏ", "without helmet"), và trạng thái đèn giao thông.

3.2 Đối tượng nghiên cứu

3.2.1 Giới thiệu đối tượng nghiên cứu

Đối tượng nghiên cứu bao gồm:

- **Phương tiện giao thông:** Xe hơi, xe máy, xe đạp, xe tải và xe buýt được nhận diện trong các video giao thông. Xe máy là đối tượng chính để kiểm tra vi phạm không đội mũ bảo hiểm.
- **Người điều khiển xe máy:** Để xác định trạng thái đội mũ bảo hiểm có đội hay không đội mũ bảo hiểm.
- **Đèn giao thông:** Để xác định trạng thái đèn (đỏ, xanh, vàng), từ đó phát hiện vi phạm vượt đèn đỏ.

3.2.2 Nguyên tắc hoạt động của hệ thống

Nhận diện phương tiện: Sử dụng mô hình YOLOv11 để phát hiện các loại phương tiện giao thông như xe hơi, xe máy, xe tải, xe buýt trong từng frame video. Ngưỡng tin cậy (conf) được đặt là 0.5 và ngưỡng IoU là 0.65.

Theo dõi phương tiện: Thuật toán DeepSort (deep_sort_realtime) được sử dụng để gán ID cho từng phương tiện và theo dõi chuyển động qua các frame. DeepSort giúp xác định hướng di chuyển và lịch sử vị trí của phương tiện.

Phát hiện mũ bảo hiểm: Với các phương tiện được xác định là xe máy, vùng bounding box của xe máy được cắt và đưa vào mô hình YOLOv11 để kiểm tra trạng thái đội mũ bảo hiểm. Ngưỡng tin cậy là 0.5 và IoU là 0.45. Nếu phát hiện "without helmet", nhãn sẽ được hiển thị màu đỏ.

Phát hiện màu đèn giao thông: Một vùng ROI cố định được sử dụng để phân tích màu đèn. Hình ảnh được chuyển sang không gian màu HSV, và tỷ lệ màu đỏ, xanh, vàng được tính và cài đặt dựa theo từng cột đèn giao thông ở các vị trí khác nhau trên đường phố.

Phát hiện vi phạm:

- **Vượt đèn đỏ:** Một vạch dừng được xác định bằng tọa độ cố định, hiển thị bằng đường màu xanh dương. Nếu đèn đỏ và phương tiện vượt qua vạch dừng, đồng thời di chuyển theo hướng thẳng (xác định qua lịch sử vị trí từ DeepSort), hệ thống ghi nhận vi phạm "Vượt đèn đỏ". Nếu xe máy hoặc xe oto vượt qua vạch dừng thì phương tiện đó được gắn nhãn là "Vượt đèn đỏ".
- **Không đội mũ bảo hiểm:** Nếu mô hình YOLOv11 phát hiện người điều khiển xe máy không đội nón bảo hiểm thì nhãn "without helmet" sẽ được hiển thị.

3.3 Địa điểm và thời gian nghiên cứu

Quá trình nghiên cứu được thực hiện chủ yếu trên máy tính cá nhân của các thành viên trong nhóm, cho phép linh hoạt lựa chọn địa điểm làm việc mà không bị giới hạn bởi một không gian cụ thể. Việc sử dụng mô hình làm việc từ xa giúp mỗi thành viên có thể phát huy tối đa thế mạnh cá nhân, đồng thời chủ động trong việc lên kế hoạch và sắp xếp thời gian làm việc phù hợp.

Để đảm bảo tính phối hợp và tiến độ công việc, nhóm nghiên cứu duy trì liên lạc thường xuyên thông qua các nền tảng làm việc trực tuyến như Google Meet, Zalo, Google Drive, giúp chia sẻ dữ liệu, phân chia công việc và thảo luận các vấn đề kỹ thuật một cách hiệu quả. Ngoài ra, các buổi họp nhóm định kỳ hàng tuần được tổ chức nhằm rà soát tiến độ, giải quyết các vướng mắc phát sinh và đảm bảo sự thống nhất trong từng bước triển khai.

Thời gian dự kiến để hoàn thành toàn bộ nghiên cứu là 6 tuần, bắt đầu tính từ thời điểm khởi động dự án. Khoảng thời gian này được hoạch định nhằm đảm bảo mọi giai đoạn trong quy trình nghiên cứu – từ lập kế hoạch, xử lý và thu thập dữ liệu, đến huấn luyện mô hình, đánh giá và tổng hợp kết quả – đều được thực hiện một cách kỹ lưỡng, bài bản và có sự phối hợp chặt chẽ giữa các thành viên trong nhóm.

3.4 Công cụ nghiên cứu và kỹ thuật thu thập thông tin

Dự án sử dụng các công cụ và thư viện sau để xây dựng hệ thống phát hiện vi phạm giao thông, kết hợp với kỹ thuật thu thập dữ liệu phù hợp:

Ngôn ngữ lập trình: Python được sử dụng làm ngôn ngữ chính, chạy trên môi trường ảo (venv) để quản lý các thư viện và đảm bảo tính tương thích.

Thư viện chính:

- **OpenCV:** Thư viện xử lý hình ảnh và video, được sử dụng để đọc video, xử lý từng frame, và hiển thị kết quả. OpenCV hỗ trợ vẽ các bounding box, đường vạch dừng, và nhãn vi phạm trên video đầu ra.
- **Ultralytics YOLO:** Triển khai mô hình YOLOv11 để phát hiện các đối tượng, bao gồm xe hơi, xe máy, xe buýt, xe tải và trạng thái đội mũ bảo hiểm của người điều khiển xe máy.
- **DeepSort (deep_sort_realtime):** Thuật toán theo dõi phương tiện qua các frame video, giúp gán ID duy nhất cho từng phương tiện và lưu trữ lịch sử vị trí để phát hiện vi phạm vượt đèn đỏ.

Công cụ hỗ trợ gắn nhãn: MakeSense AI, một công cụ trực tuyến hỗ trợ gắn nhãn dữ liệu hình ảnh, được nhóm sử dụng để xác định và gắn nhãn các phương tiện (xe hơi, xe máy) và trạng thái đội mũ bảo hiểm của người điều khiển xe máy trong các hình ảnh huấn luyện và xác thực. MakeSense AI cho phép tạo bounding box và gắn nhãn một cách nhanh chóng theo định dạng của YOLO, hỗ trợ chuẩn bị dữ liệu cho việc huấn luyện mô hình YOLOv11.

Môi trường triển khai: Để tối ưu quá trình huấn luyện mô hình YOLOv11, nhóm đã sử dụng nền tảng Kaggle Notebooks, cho phép khai thác GPU miễn phí có hỗ trợ CUDA. Việc sử dụng Kaggle giúp đẩy nhanh quá trình huấn luyện mô hình trên tập dữ liệu có kích thước lớn, đồng thời đảm bảo tính linh hoạt trong việc thử nghiệm và tinh chỉnh tham số mô hình. Sau khi huấn luyện, mô hình được tải về và triển khai cục bộ trong hệ thống chính.

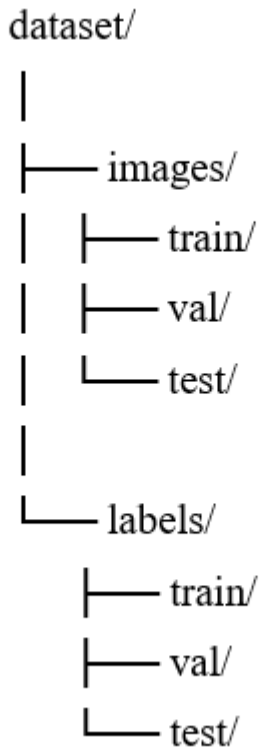
Hệ thống sau đó được tích hợp và phát triển trên môi trường lập trình Visual Studio Code chạy trên hệ điều hành Windows 11. Visual Studio Code được cấu hình với môi trường ảo (venv) để quản lý thư viện Python, đồng thời hỗ trợ các tiện ích như gỡ lỗi (debugging) và quản lý mã nguồn. GPU hỗ trợ CUDA cũng được tận dụng trong quá trình suy luận để tăng tốc độ xử lý video và hình ảnh đầu vào.

Kỹ thuật thu thập dữ liệu: Các video giao thông được thu thập từ hai nguồn chính: camera giám sát giao thông được đặt cố định tại các giao lộ ngã tư, ghi lại cảnh giao thông thực tế tại các ngã tư ở Việt Nam, và các nguồn công khai (giả định) như video có sẵn trên mạng. Video được lưu dưới định dạng MP4 và sử dụng làm dữ liệu đầu vào để

kiểm tra hệ thống. Đối với dữ liệu hình ảnh huấn luyện, nhóm đã sử dụng MakeSense AI để gắn nhãn các phương tiện và mũ bảo hiểm, tạo dữ liệu đầu vào cho mô hình YOLOv11 vì vậy hệ thống sẽ hoạt động hiệu quả trên dữ liệu thực tế, với khả năng nhận diện phương tiện và phát hiện vi phạm dễ dàng hơn tại các ngã ba, ngã tư.

3.5 Tập dữ liệu

Trong dự án, nhóm sử dụng **hai tập dữ liệu riêng biệt** để huấn luyện hai mô hình YOLOv11 khác nhau, phục vụ cho hai nhiệm vụ chính của hệ thống: (1) phát hiện phương tiện giao thông, và (2) phát hiện nón bảo hiểm. Việc tách riêng hai mô hình nhằm tăng độ chính xác và tối ưu hóa khả năng nhận diện của từng tác vụ cụ thể. Cả hai tập dữ liệu đều được tiền xử lý, gắn nhãn theo định dạng chuẩn của YOLO, và tổ chức theo cấu trúc thư mục phù hợp cho quá trình huấn luyện.



Hình 1. Cấu trúc thư mục YOLO

3.5.1 Tập dữ liệu huấn luyện mô hình YOLOv11 phát hiện phương tiện

Tập dữ liệu này bao gồm **2.519 hình ảnh** về các loại phương tiện như xe máy, ô tô, xe buýt và xe tải, được trích xuất từ video giao thông thực tế và một số nguồn công khai. Dữ liệu được gắn nhãn bằng công cụ MakeSense AI theo định dạng .txt đi kèm với từng ảnh .jpg, với mỗi dòng thể hiện một bounding box và nhãn lớp đối tượng theo định dạng chuẩn của YOLO.

Bảng 1. Phân bố ảnh trong tập dữ liệu mô hình phát hiện phương tiện

Tập con	Tỷ lệ	Số lượng ảnh	Ghi chú
Train	84%	2.107	Đa dạng loại phương tiện và điều kiện ánh sáng
Validation	8%	206	Dùng để điều chỉnh tham số mô hình
Test	8%	206	Đánh giá độ chính xác cuối cùng của mô hình

Việc chia tách này nhằm đảm bảo mô hình học được đặc trưng chung của dữ liệu và giảm thiểu hiện tượng overfitting.

3.5.2 Tập dữ liệu huấn luyện mô hình YOLOv11 phát hiện nón bảo hiểm

Tập dữ liệu này bao gồm **6978 hình ảnh** tập trung vào việc phát hiện người điều khiển xe máy có hoặc không đội nón bảo hiểm. Dữ liệu được thu thập từ nhiều nguồn, đảm bảo đa dạng về góc nhìn, thời tiết và điều kiện ánh sáng. Công cụ MakeSense AI cũng được sử dụng để gắn nhãn hình ảnh với hai lớp chính: **"Helmet"** và **"WithoutHelmet"**.

Bảng 2. Phân bố ảnh trong tập dữ liệu mô hình phát hiện nón bảo hiểm

Tập con	Tỷ lệ	Số lượng ảnh	Ghi chú
Train	90%	6262	Gồm người đội và không đội mũ bảo hiểm
Validation	5%	373	Kiểm tra khả năng phân biệt hai lớp
Test	5%	343	Dùng cho đánh giá độc lập sau huấn luyện

Mỗi tập con chứa đầy đủ thông tin nhãn kèm theo ảnh, đảm bảo sẵn sàng cho quá trình huấn luyện mô hình trong môi trường Ultralytics YOLOv11.

3.5.3 Đánh giá chất lượng dữ liệu và minh họa

Việc đảm bảo chất lượng dữ liệu đầu vào là yếu tố then chốt ảnh hưởng trực tiếp đến hiệu quả của mô hình học sâu. Các tiêu chí nhóm sử dụng để đánh giá chất lượng tập dữ liệu bao gồm:

- **Đa dạng bối cảnh**: Hình ảnh được thu thập từ nhiều thời điểm trong ngày, nhiều góc quay khác nhau và trong các điều kiện thời tiết đa dạng (nắng, râm, mưa nhẹ), giúp mô hình học được các đặc trưng tổng quát.
- **Cân bằng lớp**: Trong cả hai tập dữ liệu, tỷ lệ xuất hiện của các lớp (ví dụ: xe máy, ô tô, có nón, không nón) được kiểm tra và điều chỉnh để tránh hiện tượng mất cân bằng nghiêm trọng.
- **Chất lượng hình ảnh**: Các hình ảnh mờ, nhiễu, hoặc có vật cản che khuất đối tượng được lọc bỏ trong giai đoạn tiền xử lý để đảm bảo độ chính xác cao khi gắn nhãn.
- **Tính nhất quán của nhãn**: Tất cả nhãn được kiểm tra thủ công sau khi gắn nhãn tự động bằng MakeSense AI để đảm bảo bounding box được đặt chính xác và đúng lớp.

3.6 Phương pháp sử dụng

Trong nghiên cứu này, nhóm xây dựng một hệ thống phát hiện vi phạm giao thông sử dụng các mô hình học sâu kết hợp với theo dõi đối tượng và xử lý video thời gian thực. Hệ thống tập trung vào hai hành vi chính: vượt đèn đỏ và không đội mũ bảo hiểm, được phát hiện bằng cách tích hợp hai mô hình YOLOv11 và thuật toán DeepSORT.

3.6.1 YOLOv11 (You Only Look Once version 11)

YOLO là một thuật toán phát hiện đối tượng trong thị giác máy tính. Điểm đặc biệt của YOLO nằm ở cách nó đưa ra dự đoán về các hộp giới hạn (bounding box) và xác suất của đối tượng trong một lần truyền hình ảnh duy nhất. Nó được áp dụng vào dự án này để phát hiện phương tiện giao thông và mũ bảo hiểm.

Ưu điểm của YOLOv11 được đánh giá là nhanh chóng và thời gian phản hồi thấp, giúp xử lý các tác vụ nhận diện đối tượng và phân-segment ảnh trong thời gian thực. YOLOv11 được xây dựng trên các tiến bộ về học sâu và thị giác máy tính, đảm bảo độ chính xác cao trong việc nhận diện đối tượng. YOLOv11 còn hỗ trợ việc nhận diện đối tượng và phân-segment trên cả GPU và CPU, tận dụng các công nghệ như TensorRT của Nvidia và OpenVino của Intel.

Kiến trúc YOLOv11 có thể được chia thành ba thành phần chính:

- **Backbone (Mạng nền tảng)**: Đây là mạng nơ-ron tích chập (CNN) có nhiệm vụ trích xuất đặc trưng từ ảnh đầu vào. YOLOv11 sử dụng một backbone tùy biến là **CSPDarknet53**, tích hợp các kết nối phân đoạn theo tầng (Cross-Stage Partial connections) nhằm cải thiện luồng thông tin giữa các lớp và tăng độ chính xác của mô hình.
- **Neck (Khối trung gian)**: Neck, còn được gọi là bộ trích xuất đặc trưng, có nhiệm vụ kết hợp các bản đồ đặc trưng (feature maps) từ nhiều tầng khác nhau trong backbone để nắm bắt thông tin ở nhiều cấp độ. Kiến trúc YOLOv11 sử dụng **mô-đun C2f** mới thay vì mạng kim tự tháp đặc trưng truyền thống (FPN).

Mô-đun này kết hợp giữa các đặc trưng ngữ nghĩa ở tầng cao với thông tin không gian ở tầng thấp, giúp tăng độ chính xác, đặc biệt trong việc phát hiện các đối tượng nhỏ.

- **Head (Khối đầu ra)**: Đây là thành phần chịu trách nhiệm tạo ra các dự đoán. YOLOv11 sử dụng nhiều module phát hiện để dự đoán các bounding box, điểm tin cậy (objectness score) và xác suất phân loại đối tượng tại từng ô lưới trên bản đồ đặc trưng. Các dự đoán này sau đó được tổng hợp để tạo thành đầu ra cuối cùng của hệ thống phát hiện.

3.6.2 DeepSort (Deep Simple Online and Realtime Tracking):

DeepSORT (Deep learning-based SORT) là một thuật toán theo dõi đối tượng trên deep learning, kết hợp deep learning với thuật toán theo dõi đối tượng truyền thống SORT (simple Online and Realtime Tracking).

DeepSORT phát hiện mục tiêu trong mỗi khung hình dựa trên các máy dò đối tượng (chẳng hạn như YOLO, Faster R-CNN, v.v.), sử dụng công nghệ hợp nhất tính năng Multi để biểu diễn và mô tả mục tiêu, sau đó sử dụng thuật toán SORT để theo dõi mục tiêu. Trên cơ sở thuật toán SORT, DeepSORT giới thiệu mô hình Re-Identification (Re-ID) để giải quyết vấn đề xác định ID mục tiêu và mô hình Re-ID xác định ID duy nhất của mục tiêu bằng cách tính toán độ tương đồng của mục tiêu trong nhiều hình ảnh khung hình.

Ưu điểm của thuật toán DeepSORT là: độ chính xác cao, độ bền mạnh và khả năng thích ứng tốt với sự tắc nghẽn và biến dạng mục tiêu. Nó đã được sử dụng rộng rãi trong việc theo dõi người đi bộ, xe cộ và các mục tiêu khác và giám sát video thông minh.

Trong dự án, DeepSort hỗ trợ theo dõi chuyển động của xe hơi và xe máy, cung cấp thông tin cần thiết để xác định vi phạm vượt đèn đỏ (hướng di chuyển và vị trí so với vạch dừng).

Kiến trúc của DeepSort bao gồm:

- **Kalman Filter:** Dự đoán vị trí của các phương tiện trong các frame tiếp theo dựa trên lịch sử vị trí và vận tốc. Kalman Filter giúp duy trì tính liên tục của các track (ID của phương tiện) ngay cả khi đối tượng bị che khuất tạm thời.
- **Mạng học sâu (Appearance Model):** Trích xuất đặc trưng ngoại hình (appearance features) của các phương tiện từ vùng bounding box được phát hiện bởi YOLOv11. Đặc trưng này được sử dụng để so sánh giữa các frame, đảm bảo tính nhất quán của ID ngay cả khi có nhiều phương tiện tương tự xuất hiện.
- **Thuật toán Hungarian (Assignment):** Gán các ID cho các phương tiện dựa trên sự kết hợp giữa dự đoán vị trí (từ Kalman Filter) và đặc trưng ngoại hình (từ mạng học sâu). Thuật toán này giúp giảm thiểu lỗi gán sai ID, đặc biệt trong các tình huống giao thông đông đúc.

3.6.3 Kiến trúc tổng thể của hệ thống

Hệ thống phát hiện vi phạm giao thông được thiết kế theo mô hình xử lý **đa luồng (multi-threaded)**, nhằm đảm bảo hiệu năng và khả năng xử lý video gần thời gian thực. Kiến trúc hệ thống được chia thành ba thành phần chính hoạt động song song, mỗi thành phần đảm nhiệm một vai trò riêng biệt trong toàn bộ pipeline xử lý:

a) Luồng đọc video (Frame Reader Thread)

Luồng này có nhiệm vụ mở và đọc video đầu vào (thường là các video từ camera giám sát tại các giao lộ). Mỗi khung hình (frame) được trích xuất tuần tự và đưa vào hàng đợi frame_queue. Hàng đợi này đóng vai trò trung gian, giúp tách biệt quá trình đọc video với quá trình xử lý, đồng thời giảm tải cho CPU trong trường hợp xử lý bị chậm.

b) Luồng xử lý (Processing Thread)

Đây là luồng trung tâm thực hiện tất cả các thao tác xử lý ảnh, bao gồm:

Phân tích đèn giao thông: Một vùng quan tâm (Region of Interest – ROI) được xác định tại vị trí đèn giao thông trong mỗi frame. ROI này được chuyển sang không gian màu **HSV** để tách biệt rõ ràng các thành phần màu. Các ngưỡng màu được định nghĩa trước để nhận diện đèn đỏ, xanh và vàng dựa trên tỷ lệ diện tích màu tương ứng.

Công thức tính tỷ lệ màu:

$$Ratio_{\{color\}} = \frac{\{cv2.countNonZero(mask_{\{color\}})\}}{\{area\}}$$

Trong đó, $mask_{color}$ là mask nhị phân của màu (đỏ, xanh, vàng) và area là diện tích vùng ROI.

Phát hiện phương tiện giao thông: Sử dụng mô hình YOLOv11 thứ nhất để phát hiện các phương tiện như xe máy, ô tô, xe buýt, xe tải. Kết quả phát hiện bao gồm tọa độ bounding box, độ tin cậy và nhãn lớp đối tượng.

Theo dõi phương tiện bằng DeepSORT: Các đối tượng được phát hiện được chuyển qua thuật toán DeepSORT, giúp gán ID cố định cho mỗi phương tiện và theo dõi liên tục qua các frame. DeepSORT dựa trên thông tin không gian và đặc trưng đối tượng để duy trì tính nhất quán của ID.

Phát hiện hành vi vượt đèn đỏ: Với thông tin màu đèn và hướng di chuyển (dựa trên vector vị trí từ DeepSORT), hệ thống xác định xem phương tiện có di chuyển vượt qua vạch dừng khi đèn đỏ hay không. Điều kiện kiểm tra bao gồm:

- Tọa độ bounding box vượt qua vị trí vạch dừng
- Hướng di chuyển từ dưới lên (trong khung hình)
- Đèn giao thông đang ở trạng thái đỏ

Phát hiện nón bảo hiểm: Đối với các phương tiện là xe máy, hệ thống trích xuất vùng đầu người điều khiển và sử dụng mô hình YOLOv11 thứ hai để phát hiện xem người đó có đội mũ bảo hiểm hay không. Dữ liệu phát hiện gồm bounding box và nhãn ("có nón" hoặc "không nón").

Kết quả xử lý cuối cùng được đưa vào hàng đợi processed_frame_queue để chuyển sang luồng hiển thị.

c) Luồng hiển thị (Display Thread)

Luồng này đảm nhiệm việc **hiển thị các khung hình đã xử lý**, kèm theo các thông tin trực quan:

- Bounding box và nhãn lớp cho từng phương tiện
- ID của phương tiện (do DeepSORT gán)
- Nhãn vi phạm như "**Vượt đèn đỏ**" hoặc "**Không đội mũ bảo hiểm**"
- Vị trí vạch dừng và trạng thái đèn giao thông

Việc hiển thị giúp kiểm tra trực quan hiệu quả của hệ thống, đồng thời phục vụ cho việc ghi hình, trình chiếu hoặc ghi log.

3.7 Tiêu chí đánh giá

Hệ thống được đánh giá dựa trên các tiêu chí sau: **Accuracy**, **F1-score và mAP@0.5**.

3.7.1 Accuracy (Độ chính xác)

Accuracy đo lường tỷ lệ dự đoán đúng trên tổng số dự đoán, bao gồm cả các trường hợp phát hiện đúng và không phát hiện nhầm.

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

- TP (True Positive): Số đối tượng không tồn tại và được hệ thống dự đoán là không có).
- TN (True Negative): Số đối tượng không tồn tại và được hệ thống dự đoán là không có.
- FP (False Positive): Số trường hợp mô hình phát hiện nhầm (phát hiện sai đối tượng).
- FN (False Negative): Số đối tượng có trong ảnh nhưng bị mô hình bỏ sót.

Mặc dù Accuracy cho cái nhìn tổng thể, nhưng trong bài toán có dữ liệu mất cân bằng (ví dụ: ít người không đội nón hơn người có đội), các tiêu chí khác như F1-score hoặc mAP sẽ phản ánh hiệu suất rõ ràng hơn.

3.7.2 F1-score

F1-score là trung bình điều hòa giữa Precision và Recall, giúp cân bằng giữa khả năng phát hiện đúng và tránh dự đoán sai. Phù hợp với bài toán có dữ liệu không cân bằng như phân loại "có nón" và "không nón".

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Trong đó:

- $Precision = \frac{TP}{TP+FP}$: Tỷ lệ dự đoán đúng trên tổng số dự đoán dương.
- $Recall = \frac{TP}{TP+FN}$: Tỷ lệ đối tượng được phát hiện đúng trên tổng số đối tượng thực sự.

3.7.3 mAP@0.5 (mean Average Precision)

mAP@0.5 là tiêu chí quan trọng nhất trong đánh giá các hệ thống phát hiện đối tượng. Nó đo lường độ chính xác trung bình của mô hình trên toàn bộ các lớp, khi ngưỡng **IoU (Intersection over Union)** là 0.5. **IoU (Intersection over Union)** là tỷ lệ phần giao giữa bounding box dự đoán và bounding box thực tế.

IoU là tỷ lệ phần giao giữa **bounding box dự đoán** và **bounding box thực tế**:

$$IoU = \frac{Diện\ tích\ giao\ nhau}{Diện\ tích\ hợp\ nhau}$$

Với mỗi lớp đối tượng, mô hình vẽ đường cong Precision-Recall, tính diện tích dưới đường cong đó (gọi là **Average Precision – AP**).

mean Average Precision (mAP) là trung bình cộng của tất cả các giá trị AP trên từng lớp:

$$mAP = \frac{1}{N} . \sum_{i=1}^{N} AP_i$$

Trong đó:

- N: Tổng số lớp đối tượng trong bài toán.
- AP_i: Giá trị Average Precision của lớp thứ i.

4. KẾT QUẢ ĐẠT ĐƯỢC

Hệ thống đã thành công trong việc nhận diện xe hơi và xe máy từ video giao thông thực tế, cung cấp nền tảng quan trọng để phát hiện vi phạm. Mô hình YOLOv11 đã chứng minh khả năng phân biệt chính xác các phương tiện, giúp hệ thống xác định các đối tượng cần kiểm tra vi phạm. Đối với vi phạm không đội mũ bảo hiểm, hệ thống đã cho thấy tiềm năng trong việc phát hiện trạng thái đội mũ của người điều khiển xe máy, mặc dù hiệu quả cần được kiểm chứng thêm trên tập dữ liệu đa dạng hơn.

Đặc biệt, hệ thống đã đạt được kết quả đáng chú ý trong việc phát hiện vi phạm vượt đèn đỏ. Nhờ việc sử dụng thuật toán DeepSort để theo dõi chuyển động phương tiện và phân tích màu đèn giao thông bằng không gian màu HSV với cơ chế ổn định, hệ thống đã xác định chính xác các trường hợp vượt đèn đỏ trong điều kiện thực tế. Hình ảnh minh họa (Hình 1) thể hiện một ví dụ rõ ràng, với nhãn "Vượt đèn đỏ" được hiển thị khi phương tiện vượt qua vạch dừng trong trạng thái đèn đỏ, chứng minh khả năng ứng dụng thực tiễn của hệ thống.

Ngoài ra, cơ chế ổn định màu đã cải thiện độ tin cậy khi nhận diện trạng thái đèn giao thông, giảm nhiễu do ánh sáng hoặc góc quay thay đổi, đặc biệt hữu ích khi vùng ROI được xác định thủ công cho từng video. Tuy nhiên, hiệu suất tổng thể vẫn cần được đánh giá định lượng trên tập dữ liệu kiểm tra có nhãn để khẳng định độ chính xác và độ bao phủ của hệ thống.

Tóm lại, nghiên cứu đã đạt được các kết quả ban đầu ấn tượng, đặc biệt trong việc phát hiện vi phạm vượt đèn đỏ, và đặt nền tảng cho việc mở rộng ứng dụng hệ thống trong quản lý giao thông thực tế.

4.1. Công cụ và thư viện đã sử dụng

Dự án đã sử dụng các công cụ và thư viện sau để xây dựng hệ thống phát hiện vi phạm giao thông:

Python: Ngôn ngữ lập trình chính, chạy trên môi trường ảo (venv) để đảm bảo tính tương thích và quản lý thư viện.

OpenCV: Thư viện xử lý hình ảnh và video, được sử dụng để đọc video, xử lý frame, và hiển thị kết quả (vẽ bounding box, nhãn vi phạm, đường vạch dừng).

Ultralytics YOLO: Triển khai mô hình YOLOv11 để phát hiện phương tiện (xe hơi, xe máy) và trạng thái đội mũ bảo hiểm của người điều khiển xe máy.

DeepSort (deep_sort_realtime): Thuật toán theo dõi phương tiện qua các frame video, giúp xác định chuyển động và lịch sử vị trí để phát hiện vi phạm vượt đèn đỏ.

NumPy: Thư viện hỗ trợ tính toán ma trận, được sử dụng trong việc xử lý dữ liệu hình ảnh (ví dụ: chuyển đổi không gian màu HSV).

Collections (Deque): Cấu trúc dữ liệu để lưu lịch sử vị trí của phương tiện (track_history) và ổn định kết quả màu đèn giao thông (color_buffer).

Các công cụ và thư viện này đã hỗ trợ hiệu quả trong việc xây dựng và triển khai hệ thống, từ xử lý video, phát hiện đối tượng, đến hiển thị kết quả trực quan.

4.2. Hiệu suất của mô hình

Hệ thống đã đạt được những kết quả quan trọng trong việc phát hiện vi phạm giao thông, bao gồm: nhận diện phương tiện giao thông, phát hiện trạng thái đội mũ bảo hiểm và ghi nhận hành vi vượt đèn đỏ. Các mô hình được huấn luyện và đánh giá thông qua các chỉ số phổ biến như Precision, Recall, F1-score và mAP@0.5. Dưới đây là các kết quả cụ thể:

4.2.1 Kết quả train mô hình phát hiện phương tiện giao thông

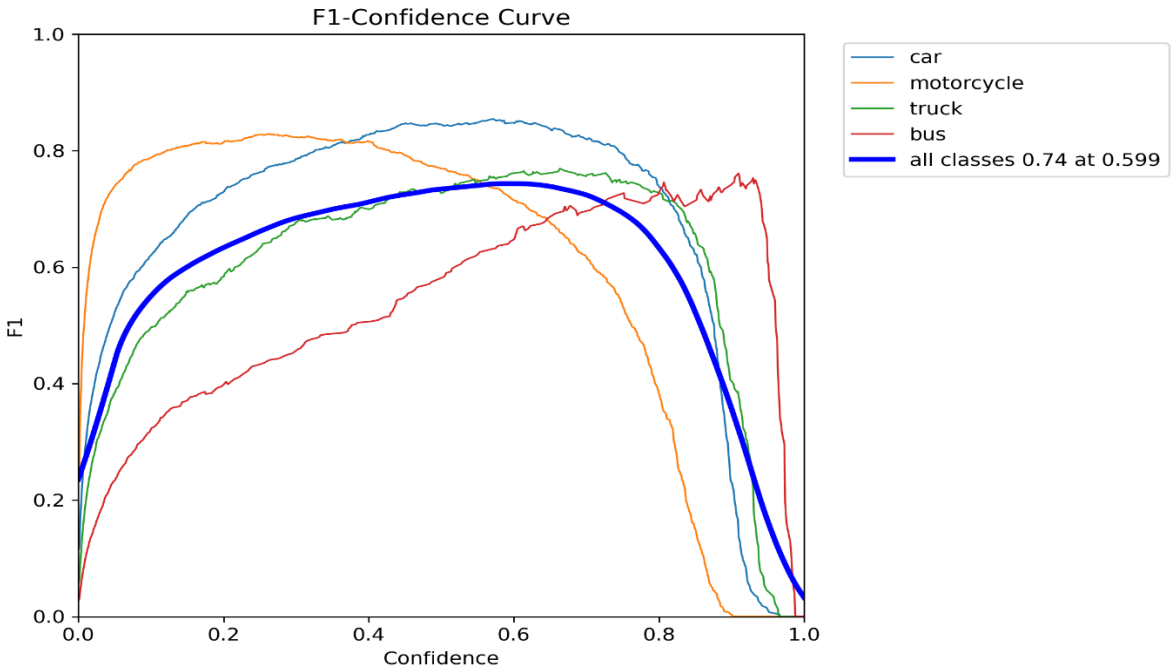
Mô hình YOLOv11 được huấn luyện để nhận diện các phương tiện phổ biến như ô tô, xe máy, xe buýt và xe tải. Kết quả đánh giá được thể hiện trong Bảng 1.

Bảng 3. Kết quả huấn luyện mô hình phát hiện phương tiện (YOLOv11)

Lớp	Precision	Recall	F1-score	mAP@0.5
Car	0.869	0.828	0.848	0.915
Motorcycle	0.94	0.578	0.715	0.886
Bus	0.486	0.966	0.646	0.847
Truck	0.801	0.725	0.761	0.816
All	0.774	0.774	0.774	0.866

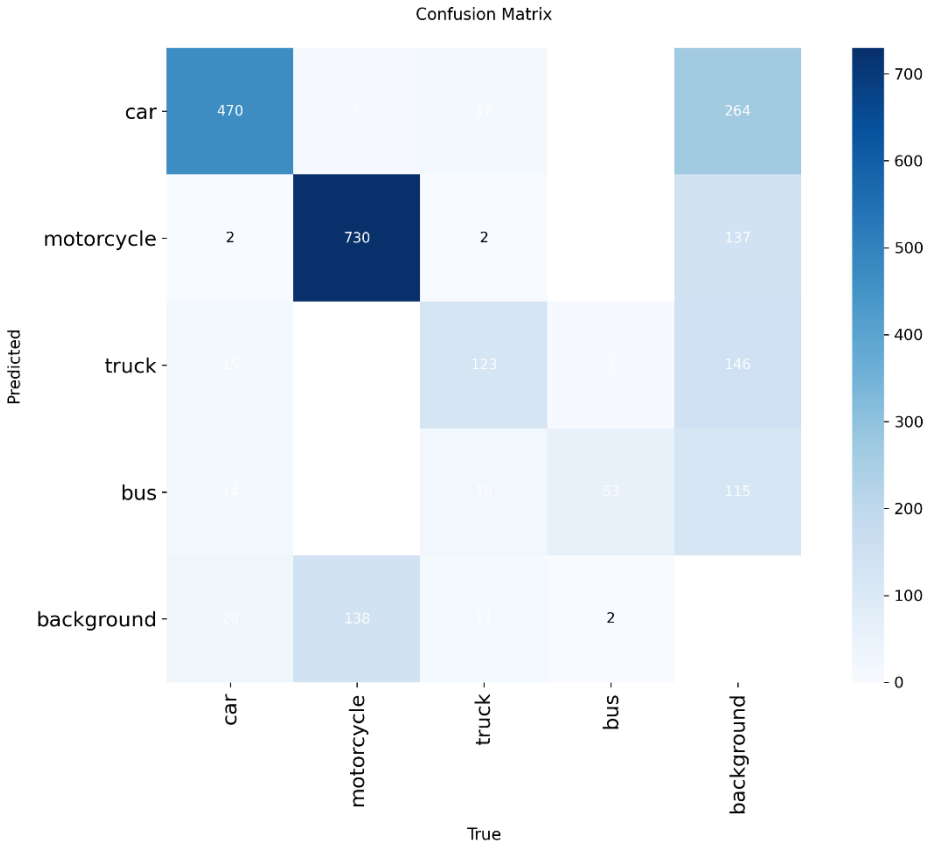
Để đánh giá chi tiết hiệu suất của mô hình YOLOv11 trong nhiệm vụ phát hiện phương tiện giao thông, hai công cụ trực quan quan trọng được sử dụng là đường cong F1-score theo confidence và ma trận nhầm lẫn (confusion matrix).

Đường cong F1-score phản ánh sự thay đổi của độ chính xác tổng hợp (F1-score) theo ngưỡng confidence – tức mức tin cậy cần có để một dự đoán được chấp nhận. Đường cong này được vẽ riêng cho từng lớp (car, motorcycle, truck, bus), cho phép xác định ngưỡng tối ưu để cân bằng giữa Precision và Recall. Mô hình đạt F1-score tốt nhất là 0.74 tại ngưỡng confidence = 0.599 với lớp motorcycle và car duy trì F1 tốt hơn truck và bus ở hầu hết các ngưỡng. Ngưỡng 0.599 được chọn để triển khai suy luận thực tế nhằm tối ưu Precision và Recal.



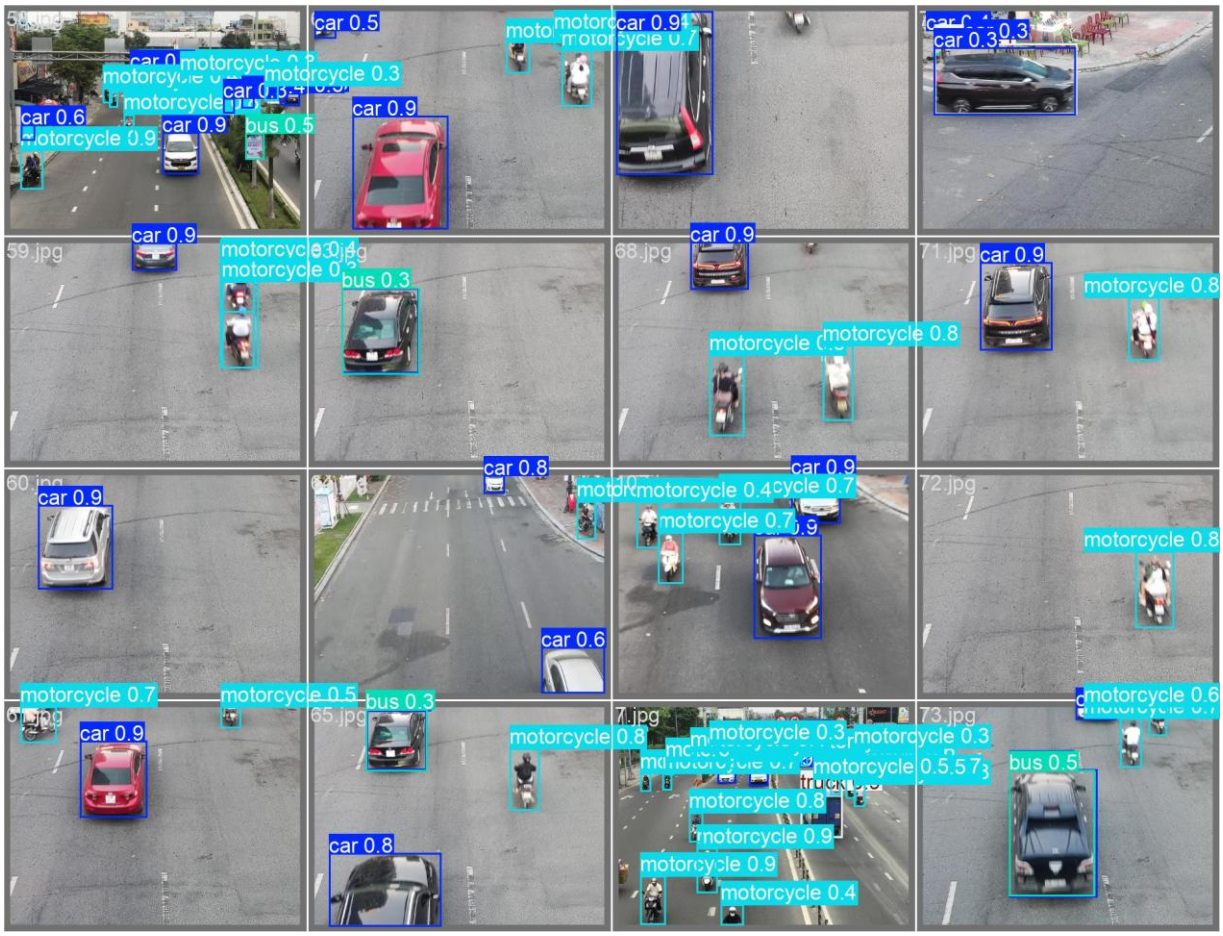
Hình 2. F1–Confidence Curve của mô hình nhận diện phương tiện

Trong khi đó, ma trận nhầm lẫn cho thấy chi tiết mức độ nhầm lẫn giữa các lớp dự đoán và nhãn thực tế. Các giá trị nằm trên đường chéo chính đại diện cho các dự đoán chính xác; các ô ngoài đường chéo phản ánh lỗi phân loại. Mô hình nhận diện rất tốt lớp "motorcycle" và "car", nhưng còn nhầm lẫn giữa "truck" và "bus" và có 138 background nhầm thành motocycle – đây là điểm cần cải thiện trong các vòng huấn luyện tiếp theo.



Hình 3. Confusion Matrix của mô hình nhận diện phương tiện

Kết quả cho thấy mô hình nhận diện tốt các phương tiện phổ biến như car và motorcycle, truck và bus với nhiều bounding box có confidence cao trên 0.8. Các nhãn lớp được hiển thị rõ ràng, không bị trùng hoặc chồng lẫn lên nhau, phản ánh khả năng định vị và phân loại chính xác của mô hình trong môi trường thực tế. Tuy nhiên vẫn có sự nhầm lẫn giữa car và bus cần cải thiện.



Hình 4. Kết quả nhận diện phương tiện giao thông

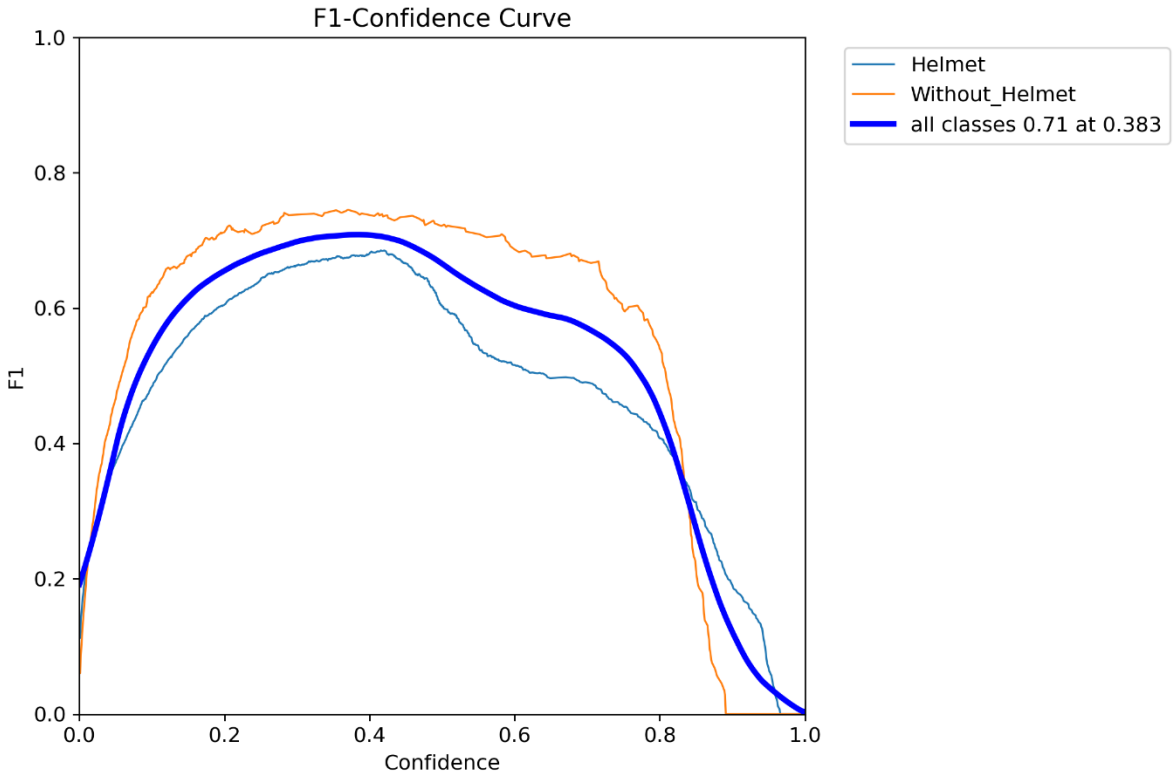
4.2.2 Kết quả train mô hình phát hiện nón bảo hiểm

Mô hình thứ hai của YOLOv11 được sử dụng để phân loại người điều khiển xe máy thành hai nhóm: "Helmet" và "Without Helmet". Kết quả được trình bày trong Bảng 4:

Bảng 4. Kết quả huấn luyện mô hình phát hiện mũ bảo hiểm (YOLOv11)

Lớp	Precision	Recall	F1-score	mAP@0.5
Helmet	0.66	0.696	0.677	0.724
WithoutHelmet	0.782	0.704	0.740	0.757
All	0.721	0.7	0.710	0.741

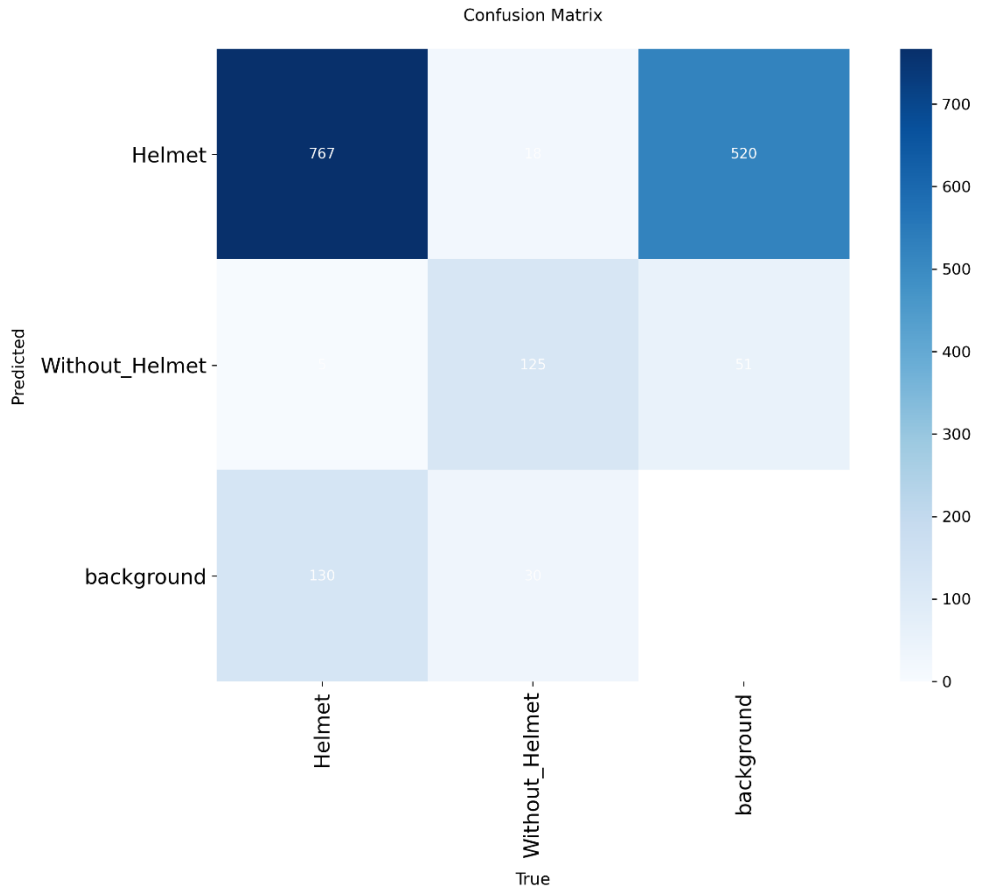
Đường cong F1–confidence trong Hình 5 minh họa sự thay đổi F1-score theo ngưỡng confidence cho từng lớp. **Tổng hợp cả ba lớp** (đường dày xanh) đạt F1 cao nhất **0.71** tại ngưỡng **0.383**. Lớp **Without_Helmet** (đường cam) đạt F1 đỉnh khoảng **0.76** quanh ngưỡng **0.38**, cao hơn so với lớp **Helmet** (đường xanh nhạt), vốn chỉ đạt đỉnh khoảng **0.7** tại khoảng **0.38**.



Hình 5. F1-Confidence Curve của mô hình nhận diện mũ bảo hiểm

Ngoài đường cong F1-score, ma trận nhầm lẫn trong Hình 6 cung cấp cái nhìn chi tiết hơn về khả năng phân loại của mô hình giữa ba lớp: Helmet, Without Helmet và Background.

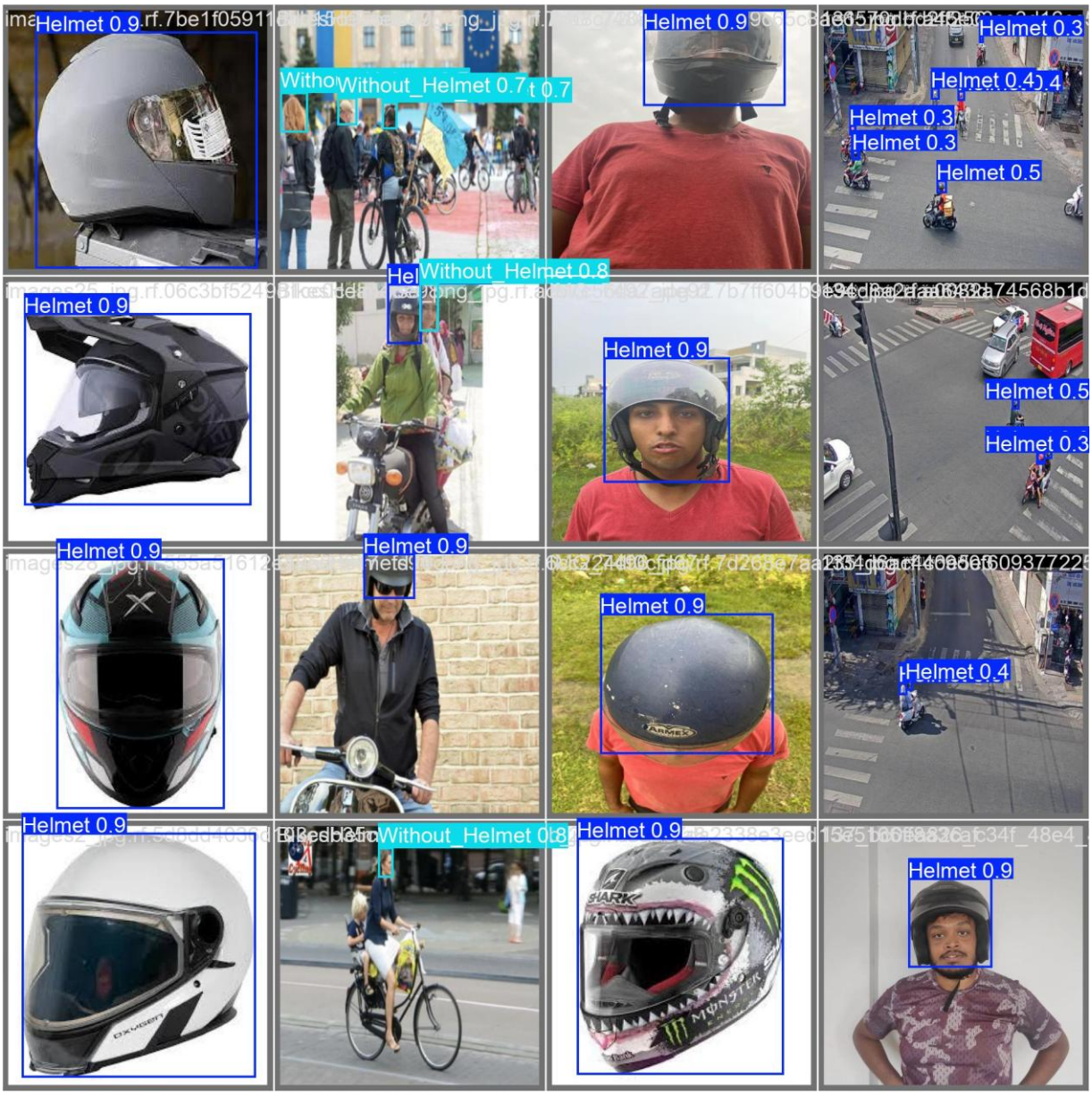
Ma trận nhầm lẫn cho thấy mô hình dự đoán đúng 767 hình có mũ, 125 hình không mũ và bỏ sót nhiều ảnh nền (130 hình nhầm thành mũ). Điều này cho thấy vẫn còn nhiều false positive trên lớp Background cần cải thiện.



Hình 6. Confusion Matrix của mô hình nhận diện mũ bảo hiểm

Hình kết quả bên dưới cho thấy mô hình YOLOv11 đã tự động phát hiện mũ bảo hiểm ("Helmet") và không đội mũ ("Without_Helmet") với các khung bao quanh (bounding box) kèm theo độ tin cậy (confidence score) trên nhiều điều kiện khác nhau: góc nhìn trực diện và nghiêng, kiểu dáng mũ đa dạng, ánh sáng thay đổi, một phần mũ bị che khuất hoặc cảnh đường phố từ xa. Phần lớn các phát hiện nón bảo hiểm đạt độ tin cậy cao (≥ 0.8), chứng tỏ mô hình vận hành ổn định, chính xác trong môi trường thực tế với cả hình ảnh có nền phức tạp và cự ly xa.

Tuy nhiên, vẫn còn nhiều dự đoán có confidence thấp (0.3–0.5), chủ yếu xảy ra trong ảnh ở xa, điều kiện ánh sáng không đồng đều, hoặc **đối tượng bị che khuất**.

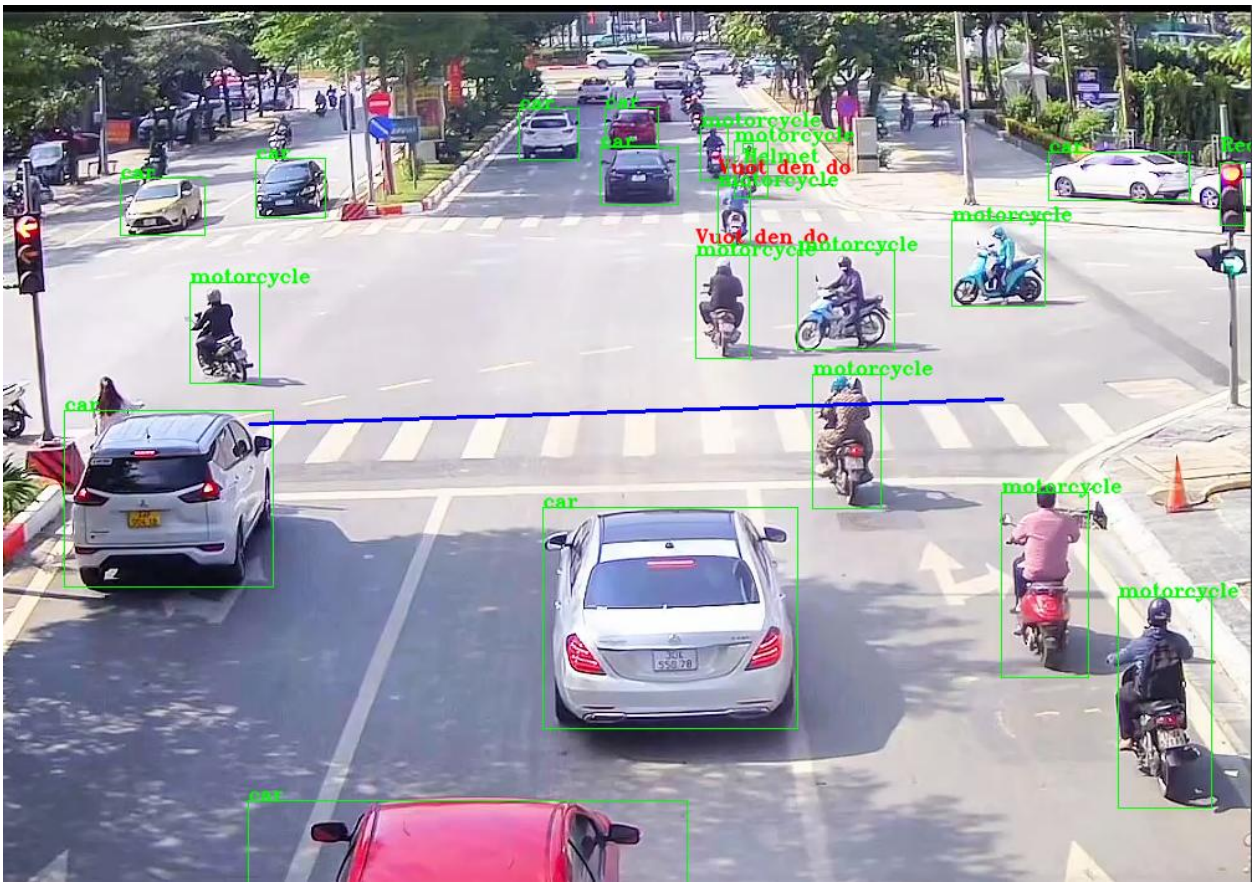


Hình 7. Kết quả nhận diện mũ bảo hiểm

4.2.3 Phát hiện vi phạm vượt đèn đỏ

Hệ thống sử dụng kỹ thuật xử lý ảnh để xác định màu đèn giao thông bằng cách phân tích vùng quan tâm (ROI) theo không gian màu HSV. Màu đèn được xác nhận là hợp lệ khi xuất hiện liên tiếp trong 9/10 frame để giảm nhiễu.

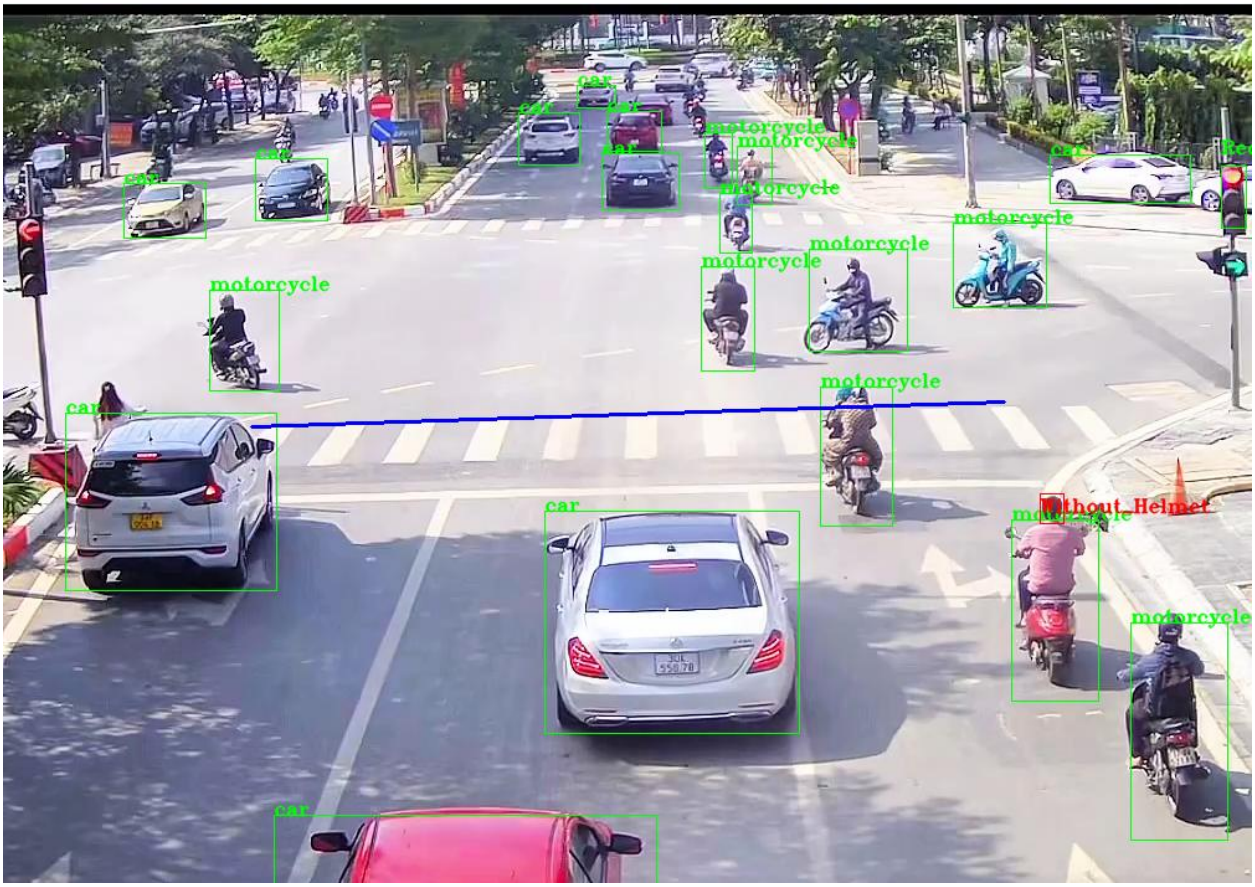
Khi đèn ở trạng thái đỏ, hệ thống so sánh vị trí của các phương tiện với vạch dừng đã được cố định trước theo tọa độ pixel. Nếu một phương tiện di chuyển vượt qua vạch này và hướng di chuyển là từ dưới lên (dựa vào vector dịch chuyển từ DeepSORT), hệ thống sẽ đánh dấu hành vi "Vượt đèn đỏ" và hiển thị nhãn cảnh báo.



Hình 8. Phát hiện lỗi vượt đèn đỏ

4.2.4 Phát hiện vi phạm không đội mũ bảo hiểm

Mô hình YOLO được sử dụng để kiểm tra trạng thái đội mũ bảo hiểm của người điều khiển xe máy. Với các phương tiện được nhận diện là xe máy, vùng bounding box được cắt và đưa vào mô hình để phân loại ("with helmet" hoặc "without helmet"). Trong trường hợp phát hiện "without helmet", nhãn sẽ được hiển thị màu đỏ.



Hình 9. Phát hiện lỗi không đội mũ bảo hiểm

4.2.5 Đánh giá tổng quan

Hệ thống đã cơ bản đạt được mục tiêu phát hiện vi phạm giao thông, bao gồm hai hành vi phổ biến là vượt đèn đỏ và không đội mũ bảo hiểm, thể hiện tính khả thi trong ứng dụng giám sát giao thông thông minh. Mô hình YOLOv11 phát hiện phương tiện giao thông cho kết quả tốt với các lớp phổ biến như ô tô và xe máy, đạt F1-score trên 0.774 và mAP@0.5 trung bình 0.866. Kết quả thực nghiệm cho thấy khả năng định vị chính xác và ổn định trong điều kiện thực tế. Trong khi đó, mô hình phân loại trạng thái mũ bảo hiểm đạt F1-score trung bình 0.710 và mAP@0.5 là 0.741, cho thấy hiệu suất ở mức thấp hơn nhưng vẫn còn hạn chế trong việc phát hiện lớp “Helmet” và “Without Helmet”. Hệ thống cũng tích hợp hiệu quả DeepSORT để theo dõi chuyển động phương tiện và xác định hành vi vượt đèn đỏ, cùng với phân tích màu đèn giao thông bằng không

gian HSV kết hợp ngưỡng ổn định (9/10 frame) nhằm tăng độ tin cậy. Tuy nhiên, vẫn còn một số hạn chế cần cải thiện như việc xác định ROI đèn giao thông còn thủ công, và tốc độ xử lý chưa được tối ưu hóa khi xử lý video độ phân giải cao hoặc giao thông dày đặc. Trong tương lai, hệ thống cần mở rộng tập dữ liệu đa dạng hơn về ánh sáng, góc quay và điều kiện giao thông, đồng thời kết hợp kỹ thuật tối ưu hóa mô hình để tăng tính ứng dụng thực tế và hiệu suất thời gian thực.

5. KẾT LUẬN VÀ HÀM Ý QUẢN TRỊ

5.1 Kết luận chung

Nghiên cứu đã phát triển thành công một hệ thống phát hiện vi phạm giao thông dựa trên các kỹ thuật thị giác máy tính, bao gồm mô hình phát hiện đối tượng YOLOv11 và thuật toán theo dõi DeepSORT. Hệ thống được thiết kế để nhận diện hai hành vi vi phạm phổ biến là vượt đèn đỏ và không đội mũ bảo hiểm. Kết quả thực nghiệm trên tập dữ liệu thực tế cho thấy hệ thống có khả năng hoạt động ổn định, nhận diện chính xác các loại phương tiện chính (car, motorcycle) với mAP@0.5 trung bình đạt 0.866, đồng thời phân loại trạng thái đội mũ bảo hiểm với độ chính xác thấp hơn (mAP@0.5 trung bình 0.741). Việc tích hợp phân tích màu đèn giao thông dựa trên không gian HSV và theo dõi hành vi phương tiện qua nhiều frame đã nâng cao độ chính xác trong phát hiện vi phạm.

5.2 Ý nghĩa của nghiên cứu

Cải thiện an toàn giao thông: Việc triển khai hệ thống phát hiện vi phạm sử dụng YOLOv11 và DeepSort giúp giảm đáng kể các hành vi nguy hiểm như vượt đèn đỏ hoặc không đội mũ bảo hiểm. Thay vì phụ thuộc vào sự hiện diện của con người, hệ thống giám sát tự động hoạt động liên tục, cung cấp dữ liệu chính xác và kịp thời, góp phần ngăn ngừa tai nạn và nâng cao ý thức chấp hành pháp luật.

Tăng cường hiệu quả quản lý: Dựa vào các đặc trưng trực quan (trạng thái đèn giao thông, sự hiện diện của mũ bảo hiểm) và hành vi di chuyển, hệ thống giúp phát hiện vi phạm một cách chính xác và khách quan. Cơ quan chức năng có thể dựa vào dữ liệu thu thập được để truy xuất, xác minh và xử phạt nhanh chóng mà không cần can thiệp trực tiếp.

Lợi ích xã hội và kinh tế: Khi vi phạm giao thông được kiểm soát tốt, các hậu quả tiêu cực như tai nạn, chi phí y tế và thiệt hại tài sản sẽ được giảm thiểu. Điều này không chỉ mang lại lợi ích cho cá nhân người dân mà còn giảm gánh nặng cho hệ thống y tế, góp phần xây dựng một môi trường giao thông an toàn và bền vững.

5.3 Hướng phát triển và đề xuất tiếp theo

5.3.1 Mở rộng phạm vi phát hiện

Hệ thống hiện tại mới tập trung vào hai hành vi vi phạm chính là vượt đèn đỏ và không đội mũ bảo hiểm. Trong giai đoạn tiếp theo, cần tích hợp thêm các hành vi vi phạm khác như lấn làn, quay đầu không đúng nơi quy định, vượt đèn khi còn tín hiệu vàng, đi sai chiều, v.v. Việc mở rộng phạm vi sẽ giúp tăng tính toàn diện của hệ thống và nâng cao hiệu quả giám sát giao thông tự động.

5.3.2 Tối ưu hóa hiệu suất mô hình và khả năng triển khai

Để đảm bảo hệ thống có thể triển khai ở quy mô lớn với chi phí hợp lý, cần áp dụng các kỹ thuật nén mô hình (model pruning), giảm chiều dữ liệu (quantization) và tối ưu hóa suy luận thời gian thực. Ngoài ra, việc triển khai mô hình trên các thiết bị phần cứng nhúng như NVIDIA Jetson Nano hoặc Raspberry Pi sẽ giúp hệ thống hoạt động liên tục tại hiện trường mà không cần phụ thuộc vào máy chủ trung tâm. Đồng thời, cần nghiên cứu các giải pháp xử lý song song đa luồng và sử dụng GPU để tăng tốc độ xử lý video độ phân giải cao.

5.3.3 Nâng cao độ chính xác mô hình và mở rộng tập dữ liệu

Hiệu suất và độ chính xác của mô hình có thể được cải thiện thông qua việc mở rộng tập dữ liệu huấn luyện, đảm bảo đa dạng về điều kiện ánh sáng, thời tiết, góc quay, vị trí địa lý và mật độ giao thông. Cần đặc biệt chú trọng đến tăng số lượng dữ liệu vi phạm (ground truth) để làm rõ ranh giới giữa hành vi hợp pháp và vi phạm. Bên cạnh đó, có thể áp dụng kỹ thuật tăng cường dữ liệu (data augmentation) và huấn luyện mô hình chuyên biệt cho từng khu vực (custom fine-tuning) nhằm nâng cao khả năng khái quát và giảm lỗi nhận diện sai.

5.3.4 Phát triển giao diện người dùng

Xây dựng một giao diện quản trị trực quan, thân thiện, cho phép lực lượng chức năng dễ dàng theo dõi vi phạm theo thời gian thực, truy xuất hình ảnh/video, thống kê vi phạm theo khu vực, khung giờ hoặc loại phương tiện. Giao diện này có thể tích hợp bản đồ, bộ lọc thông minh và chức năng xuất báo cáo để hỗ trợ tác nghiệp nhanh chóng, hiệu quả.

5.3.5 Khai thác phản hồi thực tế

Việc thu thập phản hồi từ các đơn vị trực tiếp sử dụng hệ thống như cảnh sát giao thông, thanh tra giao thông hoặc người dân tại khu vực triển khai sẽ giúp mô hình được hiệu chỉnh tốt hơn theo điều kiện cụ thể của từng địa phương. Phản hồi thực tiễn sẽ giúp xác định các lỗi nhận diện chưa chính xác, điều chỉnh vùng phát hiện, đồng thời bổ sung các tình huống ngoại lệ mà mô hình chưa nhận diện được.

6. TÀI LIỆU THAM KHẢO

Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement*. arXiv preprint arXiv:1804.02767. https://arxiv.org/abs/1804.02767

Ren, S., He, K., Girshick, R., & Sun, J. (2015). *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Advances in Neural Information Processing Systems (NeurIPS). https://arxiv.org/abs/1506.01497

Geiger, A., Lenz, P., & Urtasun, R. (2012). *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). https://www.cvlibs.net/datasets/kitti/

Jocher, G., et al. (2023). *YOLO by Ultralytics* [Computer software]. GitHub. https://github.com/ultralytics/ultralytics

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and real-time tracking with a deep association metric. *2017 IEEE International Conference on Image Processing (ICIP)*, 3645–3649. https://doi.org/10.1109/ICIP.2017.8296962

Wang, X., Zhang, J., Li, Z., Wang, Y., & Zhang, H. (2021). Real-time vehicle detection and tracking using improved YOLO and DeepSORT. *Sensors*, *21*(14), 4780. https://doi.org/10.3390/s21144780

Hsieh, J. W., Yu, S. H., Chen, Y. S., & Hu, W. F. (2006). Video-based traffic surveillance system for vehicle tracking and classification. *IEEE Transactions on Intelligent Transportation Systems*, *7*(2), 175–187. https://doi.org/10.1109/TITS.2006.875706

Nguyễn, V. A., & Trần, V. B. (2024). Ứng dụng YOLOv11 trong giám sát giao thông đô thị tại TP. Hồ Chí Minh. *Tạp chí Khoa học & Công nghệ Giao thông Vận tải*, *12*(3), 50–58.

Lê, T. C., & Phạm, V. D. (2023). Mô hình tự động phát hiện và xử lý vi phạm giao thông tại các ngã tư đô thị. *Hội nghị Quốc gia về Trí Tuệ Nhân Tạo*, 115–123.

Phan, M. E., & Võ, T. F. (2022). Thị giác máy tính trong xử lý vi phạm giao thông – hiện trạng và triển vọng. *Tạp chí Công nghệ Thông tin & Truyền thông*, *9*(2), 40–46.

Trần, V. G., & Đỗ, M. H. (2023). So sánh các phiên bản YOLO trong phát hiện phương tiện giao thông. *Tạp chí Khoa học Công nghệ Giao thông*, *15*(1), 33–39.

Hồ, V. I., & Lê, M. J. (2022). Nhận dạng hành vi vi phạm giao thông bằng thị giác máy tính. *Kỷ yếu Hội nghị Công nghệ Thông tin và Truyền thông*, 89–95.

Vũ, T. K., & Nguyễn, T. L. (2024). Những thách thức trong triển khai hệ thống phát hiện vi phạm giao thông thông minh. *Tạp chí Công nghệ Thông tin Việt Nam*, *10*(1), 12–20.

YOLOv11 Architecture; Deep Dive into its Architecture - YOLOv11. (n.d.).

7. PHỤ LỤC

7.1 Mã huấn luyện mô hình cho nhận diện phương tiện

Cài đặt thư viện ultralytics để sử dụng YOLOv11

```
!pip install ultralytics
```

Tạo file data.yaml chứa đường dẫn đến tập dữ liệu train, val, test và danh sách lớp.

```
# Chuẩn bị file dữ liệu
import yaml

# Cấu trúc dữ liệu cho file data.yaml
data_yaml = {
    'train': '/kaggle/input/dataset-vehicle-version1/Data_vehicle/images/train',
    'val': '/kaggle/input/dataset-vehicle-version1/Data_vehicle/images/val',
    'test': '/kaggle/input/dataset-vehicle-version1/Data_vehicle/images/test',
    'nc': 4,
    'names': ['car','motorcycle', 'truck','bus']
}

# Ghi file data.yaml vào /kaggle/working/
with open('/kaggle/working/data.yaml', 'w') as f:
    yaml.dump(data_yaml, f)

# Kiểm tra nội dung file
!cat /kaggle/working/data.yaml
```

Sử dụng mô hình YOLOv11s pretrained, huấn luyện trong 100 epoch với kích thước ảnh 640x640 trên tập dữ liệu phương tiện giao thông.

```
from ultralytics import YOLO
model = YOLO("yolo11s.pt")
# Train the model on the dataset for 100 epochs
results = model.train( data="/kaggle/working/data.yaml", epochs=100, imgsz=640, batch=16, lr0=1e-4, optimizer='AdamW', augment=True, mosaic=0.8, mixup=0.3, hsv_h=0.015, hsv_s=0.5, hsv_v=0.4, flipud=0.0, fliplr=0.5, translate=0.2, scale=0.5, shear=0.0, degrees=10.0, erasing=0.4
)
```

7.2 Mã huấn luyện mô hình cho nhận diện nón bảo hiểm

Cài đặt thư viện ultralytics để sử dụng YOLOv11

```
!pip install ultralytics
```

Tạo file data.yaml chứa đường dẫn đến tập dữ liệu train, val, test và danh sách lớp.

```
# Chuẩn bị file dữ liệu
import yaml
# Cấu trúc dữ liệu cho file data.yaml
data_yaml = {
    'train': '/kaggle/input/dataset-helmet/Data_helmet/images/train',
    'val': '/kaggle/input/dataset-helmet/Data_helmet/images/val',
    'test': '/kaggle/input/dataset-helmet/Data_helmet/images/test',
    'nc': 2,
    'names': ['Helmet', 'Without_Helmet']
}
# Ghi file data.yaml vào /kaggle/working/
with open('/kaggle/working/data.yaml', 'w') as f:
    yaml.dump(data_yaml, f)
# Kiểm tra nội dung file
!cat /kaggle/working/data.yaml
```

Sử dụng mô hình YOLOv11s pretrained, huấn luyện trong 100 epoch với kích thước ảnh 640x640 trên tập dữ liệu nón bảo hiểm.

```
from ultralytics import YOLO
model = YOLO("yolo11s.pt")
# Train the model on the dataset for 100 epochs
results = model.train(
    data="/kaggle/working/data.yaml",
    epochs=100,
    imgsz=640,
    batch=16,
    lr0=1e-4,
    optimizer='AdamW',
    augment=True,
    mosaic=0.8,
    mixup=0.3,
    hsv_h=0.015,
    hsv_s=0.5,
    hsv_v=0.4,
    flipud=0.0,
    fliplr=0.5,
    translate=0.2,
    scale=0.5,
    shear=0.0,
    degrees=10.0,
    erasing=0.4
)
```

7.3 Mã xử lý video để phát hiện vi phạm giao thông

Tải mô hình YOLOv11 cho phương tiện và nón bảo hiểm, khởi tạo DeepSort, và sử dụng hàng đợi để quản lý frame.

```
import cv2
import numpy as np
from ultralytics import YOLO
from deep_sort_realtime.deepsort_tracker import DeepSort
from collections import deque
#tối ưu hóa bằng sử dụng bất đồng bộ
import threading
import queue

# Hàng đợi để truyền frame giữa các luồng
frame_queue = queue.Queue(maxsize=1000)  # Hàng đợi cho frame gốc
processed_frame_queue = queue.Queue(maxsize=1000)  # Hàng đợi cho frame đã xử lý
# Biến cờ để báo hiệu dừng chương trình
stop_event = threading.Event()
# Dictionary để lưu lịch sử vị trí của các track
track_history = {}
# Biến để lưu màu đèn giao thông
color_light_traffic = ""
# Tải hai mô hình YOLOv11
model_helmet = YOLO("helmet_model.pt")  # Mô hình phát hiện nón bảo hiểm
model_vehicle = YOLO("vehicle_model.pt")  # Mô hình phát hiện phương tiện
print(model_helmet.names)
print(model_vehicle.names)
# Khởi tạo DeepSort để theo dõi phương tiện
track_vehicle = DeepSort(max_age=100, n_init=1, nn_budget=50)
```

Phát hiện đèn giao thông bằng cách phân tích vùng ROI để xác định màu đèn dựa trên không gian màu HSV.

```
def detect_traffic_light_color(light_roi):
    # Kiểm tra light_roi
    if light_roi is None or light_roi.size == 0:
        print("light_roi is empty!")
        return "Unknown"
    # Tính diện tích vùng
    h, w = light_roi.shape[:2]
    area = w * h
    if area < 100:  # Diện tích quá nhỏ
        print("Region too small to analyze!")
        return "Unknown"
    # Làm mịn hình ảnh để giảm nhiễu
    light_roi = cv2.GaussianBlur(light_roi, (5, 5), 0)
    # Chuyển đổi sang không gian màu HSV
    hsv = cv2.cvtColor(light_roi, cv2.COLOR_BGR2HSV)
    # Định nghĩa các ngưỡng màu
    low_red1 = np.array([0, 15, 15])
    high_red1 = np.array([10, 255, 255])
    low_red2 = np.array([160, 30, 30])
    high_red2 = np.array([179, 255, 255])
    low_green = np.array([40, 30, 30])
    high_green = np.array([90, 255, 255])
    low_yellow = np.array([20, 30, 30])
    high_yellow = np.array([40, 255, 255])
# Tạo mask cho từng màu
    mask_red1 = cv2.inRange(hsv, low_red1, high_red1)
    mask_red2 = cv2.inRange(hsv, low_red2, high_red2)
    mask_red = cv2.bitwise_or(mask_red1, mask_red2)
```

```
mask_green = cv2.inRange(hsv, low_green, high_green)
mask_yellow = cv2.inRange(hsv, low_yellow, high_yellow)
# Tính tỷ lệ màu
red_ratio = cv2.countNonZero(mask_red) / area
green_ratio = cv2.countNonZero(mask_green) / area
yellow_ratio = cv2.countNonZero(mask_yellow) / area
# Chọn màu có tỷ lệ cao nhất
ratios = {
    "Red": red_ratio,
    "Green": green_ratio,
    "Yellow": yellow_ratio
}
max_color = max(ratios, key=ratios.get)
max_ratio = ratios[max_color]
# Đặt ngưỡng tối thiểu để tránh nhiễu
if max_ratio > 0.2:  # Ngưỡng tối thiểu 20%
    return max_color
else:
    return color_light_traffic  # Trả về màu trước đó nếu không đủ rõ ràng
```

Luồng đọc frame: đọc video và đẩy frame vào hàng đợi

```
# Luồng đọc frame
def read_frames():
    video_path = "video\\Hải Phòng.mp4"
    cap = cv2.VideoCapture(video_path)
    if not cap.isOpened():
        print("Không thể mở video. Kiểm tra đường dẫn!")
        return

    while cap.isOpened():
        ret, frame = cap.read()
        if not ret:
            print("Hết video hoặc lỗi khi đọc frame.")
            break

        # Đẩy frame vào hàng đợi
        try:
            frame_queue.put_nowait(frame)
        except queue.Full:
            pass  # Bỏ qua nếu hàng đợi đầy

    cap.release()
    frame_queue.put(None)  # Thông báo dừng
```

Luồng xử lý frame: phát hiện phương tiện, theo dõi bằng DeepSort, kiểm tra vi phạm vượt đèn đỏ và không đội nón bảo hiểm, vẽ bounding box và thông báo vi phạm.

```
# Luồng xử lý frame
def process_frames():
    global color_light_traffic
    frame_count = 0
    last_tracks = []  # Lưu trữ tracks từ lần cập nhật DeepSort trước đó
    while True:
        if stop_event.is_set():
            break
        try:
            frame = frame_queue.get(timeout=1.0)  # Lấy frame từ hàng đợi
        except queue.Empty:
            break
        if frame is None:  # Kết thúc nếu nhận được None
            break
        # Định nghĩa vùng quan tâm (ROI) - Toàn bộ frame
        roi = frame.copy()  # Sử dụng toàn bộ frame làm ROI
        #vị trí của vạch dừng
       x1_crosswalk, x2_crosswalk, y1_crosswalk, y2_crosswalk = 200, roi.shape[1]-200, 340, 320
        # Phát hiện đèn giao thông
         x1_trafficligth, y1_trafficligth, x2_trafficligth, y2_trafficligth = roi.shape[1]-25, 130, roi.shape[1]-5, 180
                                roi_light_traffic = frame[y1_trafficligth:y2_trafficligth, x1_trafficligth:x2_trafficligth]
        if frame_count % 5 == 0:  # Chỉ kiểm tra mỗi 5 frame
            color_light_traffic = detect_traffic_light_color(roi_light_traffic)
       cv2.rectangle(roi, (x1_trafficligth, y1_trafficligth), (x2_trafficligth,y2_trafficligth), (0, 255, 0), 1)
```

```
        cv2.putText(roi, color_light_traffic, (x1_trafficligth, y1_trafficligth - 10), cv2.FONT_HERSHEY_TRIPLEX, 0.5, (0, 255, 0), 1, cv2.LINE_AA)
    # Phát hiện phương tiện bằng model_vehicle
    detect_vehicle = []
    results_vehicle = model_vehicle.predict(source=frame, conf=0.5, iou=0.65)[0]
    for box in results_vehicle.boxes:
        x1, y1, x2, y2 = map(int, box.xyxy[0].tolist())
        conf = box.conf.item()
        cls = int(box.cls.item())
        # Định dạng cho DeepSort: [bbox, confidence, class]
        detect_vehicle.append([[x1, y1, x2 - x1, y2 - y1], conf, cls])
    # Theo dõi phương tiện bằng DeepSort (chỉ cập nhật mỗi 2 frame)
    if frame_count % 2 == 0:
                    current_track_vehicle=track_vehicle.update_tracks(detect_vehicle, frame=frame)
        last_tracks = current_track_vehicle  # Lưu tracks để sử dụng cho frame tiếp theo
    else:
        current_track_vehicle = last_tracks  # Sử dụng tracks cũ
    # Vẽ bounding box cho phương tiện và phát hiện nón bảo hiểm
    for track in current_track_vehicle:
        if not (track.is_confirmed() and track.det_conf):
            continue
        # Lấy tọa độ bounding box của phương tiện
        ltrb = track.to_ltrb()
        x1, y1, x2, y2 = list(map(int, ltrb))
        track_id = track.track_id
        label = track.det_class
        confidence = track.det_conf
        # Vẽ bounding box cho phương tiện
        cv2.rectangle(roi, (x1, y1), (x2, y2), (0, 255, 0), 1)
```

```
                cv2.putText(roi, model_vehicle.names[label], (x1, y1), cv2.FONT_HERSHEY_TRIPLEX, 0.5, (0, 255, 0), 1, cv2.LINE_AA)
        #xác định vi phạm vượt đèn đỏ
        # Tính tọa độ trung tâm của bounding box
        center_x = (x1 + x2) / 2
        center_y = (y1 + y2) / 2
        # Lưu lịch sử vị trí
        if track_id not in track_history:
            track_history[track_id] = deque(maxlen=5)  # Lưu tối đa 5 vị trí gần nhất
        track_history[track_id].append((center_x, center_y))
        # Tính vector di chuyển (nếu có ít nhất 2 vị trí)
        is_straight = False
        if len(track_history[track_id]) >= 2:
            prev_center_x, prev_center_y = track_history[track_id][-2]  # Vị trí trước đó
            delta_x = center_x - prev_center_x
            delta_y = center_y - prev_center_y
            # Xác định hướng di chuyển
           # Nếu delta_y < 0 (y giảm) và |deltqqa_x| nhỏ, xe di chuyển từ dưới lên (đường thẳng)
            if delta_y < 0 and abs(delta_x) < abs(delta_y) * 2 and x1 > x1_crosswalk and x2 < x2_crosswalk:  # |delta_x| nhỏ hơn 2 lần |delta_y|
                is_straight = True


        if color_light_traffic == "Red" and  (y2 < min(y1_crosswalk, y2_crosswalk) and y2 > min(y1_crosswalk, y2_crosswalk)-125 ) and is_straight:
                                cv2.putText(roi, "Vuot den do", (x1, y1 - 10), cv2.FONT_HERSHEY_TRIPLEX, 0.5, (0, 0, 255), 1, cv2.LINE_AA)


        if label == 1: #nếu là xe máy
            # Cắt vùng phương tiện để phát hiện nón bảo hiểm
            crop_img = frame[y1-25:y2, x1:x2]
            if crop_img.size != 0:  # Kiểm tra xem vùng cắt có hợp lệ không
```

```
                # Phát hiện nón bảo hiểm bằng model_helmet
                    results_helmet = model_helmet.predict(source=crop_img, imgsz=320, iou=0.45)[0]

                for helmet_box in results_helmet.boxes:
                    hx1, hy1, hx2, hy2 = map(int, helmet_box.xyxy[0].tolist())
                    hlabel = helmet_box.cls[0]
                    hconfidence = helmet_box.conf[0]
                    htext = f"{model_helmet.names[int(hlabel)]}"

                    # Vẽ bounding box cho nón bảo hiểm
                    if(hconfidence > 0.5): # Chỉ vẽ nếu nón bảo hiểm được phát hiện
                                color = (0, 0, 255) if model_helmet.names[int(hlabel)] == "Without_Helmet" else (0, 255, 0)
                        cv2.rectangle(roi, (x1 + hx1, y1-25 + hy1), (x1 + hx2, y1-25 + hy2), color, 1)
                                cv2.putText(roi, htext, (x1 + hx1, y1 + hy1 - 10), cv2.FONT_HERSHEY_TRIPLEX, 0.5, color, 1, cv2.LINE_AA)

        #xác định vi trí của vạch dừng
        cv2.line(roi, (x1_crosswalk,y1_crosswalk), (x2_crosswalk,y2_crosswalk), (255, 0, 0), 2) # Vẽ đường kẻ ngang

        # Đẩy frame đã xử lý vào hàng đợi
        try:
            processed_frame_queue.put_nowait(roi)
        except queue.Full:
            pass

        frame_count+=1
    # Đánh dấu kết thúc bằng cách đẩy None vào hàng đợi
    processed_frame_queue.put(None)
```

Luồng hiển thị: hiển thị frame đã xử lý, cho phép dừng bằng phím 'q'.

```
# Luồng hiển thị frame
def display_frames():
    while True:
        frame = processed_frame_queue.get()  # Lấy frame từ hàng đợi
        if frame is None:  # Kết thúc nếu nhận được None
            break

        cv2.imshow("Phat hien vi pham giao", frame)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            stop_event.set()
            break

    cv2.destroyAllWindows()
```

Thực thi các luồng đã định nghĩa

```
# Chạy các luồng
read_thread = threading.Thread(target=read_frames)
process_thread = threading.Thread(target=process_frames)
display_thread = threading.Thread(target=display_frames)

read_thread.start()
process_thread.start()
display_thread.start()

read_thread.join()
process_thread.join()
display_thread.join()
```