PHÁT TRIỂN ỨNG DỤNG PHÁT HIỆN HƯ HỎNG ĐƯỜNG BỘ THEO THỜI GIAN THỰC

Châu Nhật Phúc[1] và Nguyễn Anh Thư[1]

[1]Sinh viên Khoa Công nghệ thông tin, Trường Đại hoc Kỹ thuật - Công nghệ Cần Thơ
Email: cnphuc272003@gmail.com

Thông tin chung
Ngày nhận bài: 02/8/2025
Ngày nhận bài sửa: 27/10/2025
Ngày duyệt đăng: 03/11/2025

Từ khóa:
Faster R-CNN,phát hiện hư hỏng đường bộ, thời gian thực, ứng dụng android, YOLOv12

TÓM TẮT

Nghiên cứu đề xuất một ứng dụng Android có khả năng phát hiện hư hỏng đường bộ theo thời gian thực. Nhóm nghiên cứu đã huấn luyện và so sánh hai mô hình phát hiện đối tượng là Faster R-CNN và YOLOv12m trên tập dữ liệu RDD2022 gồm hơn 47000 ảnh. Kết quả cho thấy mặc dù YOLOv12m có độ chính xác thấp hơn một chút so với Faster R-CNN (mAP:50 = 65% so với 66,5%), nhưng tốc độ xử lý nhanh hơn đáng kể (77 ms/ảnh so với 104 ms/ảnh) giúp nó trở thành lựa chọn phù hợp hơn cho các ứng dụng thời gian thực. Hệ thống tích hợp mô hình YOLOv12m đặt tại máy chủ, kết nối với ứng dụng Android thông qua WebSocket, cho phép hiển thị kết quả nhận dạng trực tiếp trên giao diện người dùng. Giải pháp này phù hợp để ứng dụng trong các hệ thống giám sát hạ tầng giao thông theo thời gian thực. Nghiên cứu cung cấp một giải pháp khả thi cho việc tự động hóa giám sát hạ tầng giao thông, giúp giảm phụ thuộc vào các phương pháp thủ công. Về mặt khoa học, nghiên cứu đã thử nghiệm và đánh giá hiệu năng của mô hình YOLOv12m khi so sánh với Faster R-CNN làm nổi bật hiệu năng trong lĩnh vực phát hiện thời gian thực và có thể tạo tiền đề cho những nghiên cứu tiếp theo trong lĩnh vực thị giác máy tính, phát hiện đối tượng nói chung và trong lĩnh vực phát hiện hư hỏng đường bộ thời gian thực nói riêng.

1. ĐẶT VẤN ĐỀ

Hư hỏng trên đường bộ thường là một trong những nguyên nhân gây mất an toàn giao thông. Phương pháp kiểm tra hiện nay chủ yếu vẫn dựa trên quan sát trực tiếp vừa tốn thời gian, vừa thiếu chính xác dẫn đến hạn chế về phạm vi và tính kịp thời.

Những năm gần đây, các mô hình học sâu như Faster R-CNN [1], YOLOv8 [2], SSD [3], YOLOv11 [4],[5] đã được ứng dụng hiệu quả trong phát hiện đối tượng trên dữ liệu ảnh và video với độ chính xác cao. Tuy nhiên, đa số các nghiên cứu vẫn chưa giải quyết được bài toán xử lý thời gian thực hoặc triển khai trên thiết bị di động.

YOLOv12, phiên bản mới được phát triển với nhiều cải tiến về độ chính xác và tốc độ, kiến trúc *R-ELAN (Residual Efficient Layer Aggregation Network)*, *Area Attention*, *FlashAttention* [6],[7], được kỳ vọng phù hợp hơn cho các ứng dụng thời gian thực.

Từ đó, nhóm nghiên cứu đã đề xuất mô hình YOLOv12m (biến thể kích thước của YOLOv12) và so sánh hiệu quả với mô hình Faster R-CNN nhằm có được kết quả khách quan cả về độ chính xác lẫn tốc độ để đưa ra lựa chọn giải pháp tối ưu cho triển khai thực tế.

2. CƠ SỞ LÝ THUYẾT

2.1. YOLOv12

YOLOv12 được phát hành vào 18/02/2025 bởi các nhà nghiên cứu từ Đại học Buffalo (SUNY) và Đại học Trung Quốc (UCAS).

Kiến trúc của tổng thể của YOLOv12 vẫn được chia thành ba thành phần chính gồm: *xương sống (backbone)*; *cổ (neck)*; và *đầu*

(head) [6], nhưng đã được nâng cấp với nhiều điểm mới nhằm cải thiện hiệu suất và độ chính xác.

Cũng như các phiên bản trước đó như YOLOv10 [8], YOLOv11 [4]. YOLOv12 cũng được phát triển với nhiều phiên bản biến thể kích thước khác nhau là n, s, m, l, x.

Điểm nổi bật của YOLOv12 là thay thế kiến trúc *ELAN* bằng *R-ELAN* trong cả *xương sống* và *cổ*, kiến trúc kết hợp các lớp tích chập sâu với các kết nối dư (*residual connection*) - một kỹ thuật cho phép tín hiệu đầu vào của một lớp mạng nơ-ron được truyền trực tiếp đến đầu ra, bỏ qua một hoặc nhiều lớp trung gian [6], tăng khả năng truyền thông tin và gradient giữa các tầng khi mô hình học sâu. Đầu ra được tính bằng cách cộng đầu vào với phần đầu ra của các lớp chuyển đổi, từ đó nâng cao khả năng của mô hình trong việc nắm bắt các chi tiết đối tượng phức tạp với nhiều kích thước và hình dạng khác nhau [6].

Bên cạnh đó, YOLOv12 tích hợp cơ chế *Area Attention* [6] - một cải tiến so với cơ chế chú ý của các phiên bản YOLO trước đây, cho phép mô hình không chỉ tập trung vào từng điểm ảnh riêng lẻ mà còn xét đến cả vùng lân cận. Điều này giúp mô hình nhận diện tốt hơn các vật thể nhỏ, mờ hoặc có biên không rõ ràng [6]. Đi kèm với đó là *FlashAttention* [7], một thuật toán tối ưu hóa phát hiện trên GPU giúp tăng tốc độ tính toán và giảm tiêu thụ bộ nhớ, đảm bảo hiệu quả suy luận thời gian thực [7].

Ở tầng *đầu*, YOLOv12 loại bỏ hoàn toàn hộp neo *(anchor box)* và chuyển sang kiến trúc không sử dụng hộp tham chiếu, cho phép mô hình dự đoán trực tiếp vị trí và kích thước vật thể tại mỗi ô lưới mà không cần cấu hình trước các hộp neo [7]. Cách này giúp mô hình đơn giản hơn, linh hoạt hơn và dễ thích ứng với nhiều loại hình vật thể.

YOLOv12 được phát triển dựa trên kiến trúc của các phiên bản trước đó[6],[9], với một số cải tiến nhằm đến việc tối đa hóa cả độ chính xác và hiệu quả tính toán. Cốt lõi của nó là tận dụng *R-ELAN*, *Area Attention, FlashAttention*, và *tích chập rời 7×7* để mang lại tốc độ xử lý và độ chính xác vượt trội [7]. Bằng cách kết hợp các thành phần này, YOLOv12 nâng cao hiệu suất trong các tác vụ phát hiện đối tượng và phân vùng trường hợp, đảm bảo khả năng xử lý tốt các cảnh thị giác phức tạp với mức độ chi tiết và che khuất khác nhau.

Trong nghiên cứu này, nhóm lựa chọn phiên bản YOLOv12m (medium) để cân bằng tốt giữa hiệu năng, độ chính xác và tốc độ, phù hợp triển khai thực tế.

2.2. Faster R-CNN

Faster R-CNN là mô hình phát hiện đối tượng hai giai đoạn, phiên bản cải tiến của *Fast R-CNN*, được công bố vào năm 2015 bởi tổ chức Microsoft Research, gồm các thành phần chính: *xương sống*, *RPN (Region Proposal Network)*, *gộp vùng đề xuất (Region of Interest Pooling - ROI Pooling)*, *Fast R-CNN head* [1].

Xương sống của Faster R-CNN là mạng nơ-ron tích chập (Convolutional Neural Network) trích xuất đặc trưng từ ảnh đầu vào [10], tạo ra bản đồ đặt trưng chứa thông tin hình học quan trọng. *ResNet-50* với kết nối dư giúp khắc phục hiện tượng mất gradient [11], nâng cao khả năng học sâu.

RPN sinh ra các vùng đề xuất bằng cách quét các hộp neo (có tỉ lệ khung được cấu hình từ trước) trên bản đồ đặc trưng, đánh giá khả năng chứa đối tượng [1],[12], và hiệu chỉnh hộp giới hạn (bounding box). RPN thay thế các thuật toán chọn vùng thủ công, giúp tăng tốc và cải thiện độ chính xác.

ROI Pooling chuẩn hóa kích thước các vùng đề xuất do RPN tạo ra về kích thước cố định (thường là 7×7) để đưa vào các lớp kết nối đầy đủ [12],[13], đảm bảo mô hình xử lý đồng nhất, giữ lại các đặc trưng quan trọng mà không làm mất thông tin ảnh.

Fast R-CNN head là các lớp kết nối đầy đủ nhận dữ liệu từ *ROI Pooling*, gồm hai nhánh: phân loại đối tượng và điều chỉnh hộp giới

hạn tạo ra dự đoán cuối cùng về loại và vị trí đối tượng trong ảnh.

Faster R-CNN đạt độ chính xác cao [14], đặc biệt với các đối tượng nhỏ hoặc bị che khuất. Tuy nhiên, do có hai giai đoạn nên tốc độ suy luận thường chậm hơn so với các mô hình một giai đoạn như YOLO [14], hạn chế khả năng ứng dụng trong thời gian thực.

2.3. Dữ liệu và tiền xử lý

Tập dữ liệu RDD2022 gồm 47000 ảnh từ 6 quốc gia, gồm 7 loại hư hỏng: D00 (nứt dọc); D10 (nứt ngang); D20 (nứt mai rùa); D40 (Ổ gà; D43 (mờ vạch kẻ đường); D44 (mờ vạch sơn trắng); D50 (nắp cống).

Tiền xử lý cho YOLOv12m: chuyển các file ***.xml*** *(định dạng Pascal VOC)* ban đầu về định dạng ***.txt*** để YOLO có thể đọc được, chuẩn hóa kích thước 640x640.

Quá trình huấn luyện YOLOv12m: cấu hình gồm 16 batch, 300 epoch, sử dụng mô hình YOLOv12m đã được huấn luyện sẵn của Ultralytics. Kỹ thuật: sử dụng bộ điều chỉnh tốc độ học, lưu checkpoint định kỳ, đánh giá qua mAP (mean Average Precision).

Tiền xử lý cho Faster R-CNN: chuyển các file ***.xml*** về định dạng ***.json*** *(định dạng COCO),* chuẩn hóa kích thước 600x600.

Quá trình huấn luyện Faster R-CNN: cấu hình và kỹ thuật tương tự, nhưng sử dụng mô hình Faster R-CNN huấn luyện sẵn trong torchvision (*fasterrcnn_resnet50_fpn).*

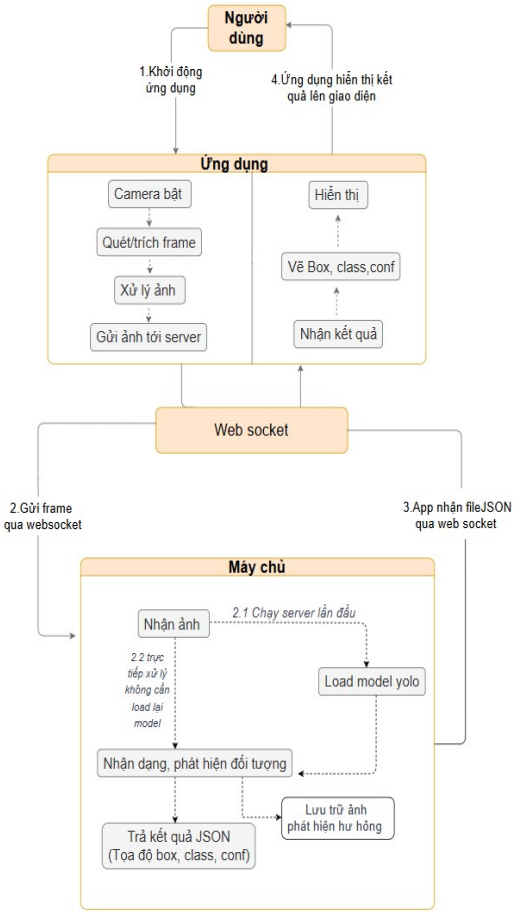
2.4. Hệ thống phát hiện thời gian thực

Kiến trúc hệ thống tổng thể được minh họa trong **Hình 1** dưới dạng luồng dữ liệu giữa các thành phần chính:

Máy khách (người dùng - ứng dụng): Dùng CameraX thu ảnh trực tiếp từ máy ảnh của thiết bị để lấy ảnh, sau khi được xử lý sơ bộ sẽ được nén WebP (gán trackId, frameId), gửi qua WebSocket [15] tới máy chủ.

Máy chủ (FastAPI - YOLOv12m): Nhận ảnh từ ứng dụng, xử lý bằng mô hình YOLOv12m, kết quả đầu ra gồm tọa độ hộp giới hạn (bounding box), nhãn lớp (class) và độ tin cậy (confidence),… được trả về dưới dạng tệp JSON.

Giao tiếp: Dựa trên WebSocket giúp duy trì kết nối hai chiều ổn định và liên tục [15], đồng bộ hóa bằng trackId, frameId đảm bảo tính nhất quán.



Hình 1. Sơ đồ tổng quát hệ thống ứng dụng

3. KẾT QUẢ NGHIÊN CỨU

3.1. So sánh YOLOv12m với Faster R-CNN

Bảng 1. So sánh kết quả giữa YOLOv12m và Faster R-CNN

Mô hình	mAP50 (%)	Fps (ảnh/giây)	Tốc độ (ms)
Faster R-CNN	66,5	9	104
YOLOv12m	65	13	77

Bảng 2. Kết quả mô hình YOLOv12m với các kịch bản tăng cường dữ liệu

Kịch bản	Tỉ lệ học	classification loss (%)	Batch Size	Tăng cường dữ liệu							mAP50(%)
				Degrees (°)	Translate (%)	Scale (%)	Shear (°)	Fliplr (%)	Mosaic (%)	Copy Paste (%)	
1	0,001	-	16	-	-	-	-	-	-	-	65
2		-		5	10	50	2	50	80	50	67
3		-		10	20	50	2	50	80	50	67,2
4		150		15	20	50	2	50	80	80	67,5

Kết quả (**Bảng 1**) được thực hiện trên Google Colab (NVIDIA Tesla T4 15GB RAM), cho thấy Faster R-CNN có chỉ số đánh giá định lượng mAP50 lớn hơn không nhiều, khi kiểm thử thực tế thông qua video do nhóm nghiên cứu tự quay thì tốc độ suy luận trung bình của YOLOv12m nhanh hơn Faster R-CNN khoảng *27m,* trong khi tốc độ nhanh là một trong những yếu tố quyết định sự mượt mà của ứng dụng thời gian thực. Vì thế, trong bối cảnh ứng dụng yêu cầu xử lý thời gian thực thì khả năng phản hồi nhanh là yếu tố ưu tiên nên YOLOv12m được lựa chọn làm mô hình phát hiện chính trong hệ thống, đảm bảo cân bằng hợp lý giữa hiệu suất và độ chính xác.

3.2. Kết quả mô hình YOLOv12m với kịch bản tăng cường dữ liệu

Sau khi YOLOv12m được lựa chọn làm mô hình chính, nhóm nghiên cứu tiến đến xây dựng kịch bản với các mức tăng cường dữ liệu khác nhau (**Bảng 2**) để tìm phương án tốt hơn từ mô hình đã chọn.

Ở **kịch bản 1**, mô hình được huấn luyện không áp dụng bất kỳ kỹ thuật tăng cường nào, kết quả thu được là mAP50 = 65%, phản ánh khả năng nhận diện hạn chế khi mô hình chỉ được huấn luyện trên tập dữ liệu gốc, không được làm phong phú qua các biến thể hình ảnh.

Kịch bản 2 áp dụng tăng cường cơ bản (degrees = 5°, translate = 10%, scale = 50%, mosaic = 80%), mAP50 tăng lên 67% cho thấy hiệu quả rõ rệt của kĩ thuật tăng cường trong việc cải thiện hiệu năng.

Kịch bản 3 nâng mức độ tăng cường lên mức trung bình (degrees = 10°, translate = 20%). Mô hình đạt mAP50 = 67,2%. Quan trọng hơn, khi được kiểm thử trên video thực tế do nhóm tự quay với đa dạng loại hư hỏng, **kịch bản 3** cho kết quả ổn định và chính xác hơn đáng kể so với các kịch bản còn lại. Điều này được minh họa trong **Hình 2**.

Hình 2. Kết quả phát hiện hư hỏng của YOLOv12m được cắt từ vid

Bảng 3. Tốc độ xử lý trung bình của hệ thống ứng dụng

Giai đoạn	Mô tả	Bắt đầu (ms)	Kết thúc (ms)	Thời gian (ms)
Trích xuất	Lấy ảnh + xử lý sơ bộ	0	12	12
Nén	Nén ảnh sang WebP	12	116	104
Gửi ảnh	Tạo WebSocket + gửi ảnh	116	118	2
Máy chủ	Mô hình suy luận	118	285	167
	Các bước phụ trợ khác	285	465	180
Vẽ	Phân tích JSON + Vẽ hộp giới hạn	465	465	~0 - 1 *(rất nhanh)*
Tổng hệ thống	Từ lúc lấy ảnh → đến khi kết quả hiển thị	0	465	465

Kịch bản 4 mặc dù đạt mAP50 = 67,5% (cao nhất về chỉ số đánh giá định lượng), nhưng lại cho kết quả kém và không nhận diện được một số lớp trong thực tế. Điều này được lý giải bởi việc áp dụng đồng thời các tăng cường mạnh (degrees = 15°, copy-paste = 80%) có thể đã làm mờ ranh giới giữa các lớp nhãn và classification loss ở mức cao 150% khiến cho mô hình bị thiên lệch vào việc tối ưu hóa khả năng phân loại, gây nhiễu quá mức trong giai đoạn huấn luyện, khiến mô hình khó học được đặc trưng phân biệt giữa các loại hư hỏng tương đồng.

Từ kết quả phân tích, **kịch bản 3** được chọn làm mô hình chính thức để triển khai vào hệ thống, nhờ đạt được sự cân bằng tốt giữa độ chính xác, tốc độ suy luận, khả năng nhận diện và độ ổn định khi thực nghiệm.

3.3. Ứng dụng Android

Hỗ trợ chế độ chụp ảnh, quay video, lưu kết quả và chuyển đổi linh hoạt giữa hai chế độ.

Ứng dụng được thực nghiệm trên điện thoại Samsung galaxy A51, hệ điều hành Android 13, CPU Exynos 9611 (10 nm), GPU Mali-G72 MP3, RAM 8GB, camera sau 48MP có độ trễ hệ thống được minh họa ở **Bảng 3.**

Mô hình YOLOv12m được đặt tại máy chủ có cấu hình 4GB RAM GPU NVIDIA RTX 3050, CPU Intel Core i5-13500H, 16GB RAM hệ thống.

4. KẾT LUẬN

Nghiên cứu đã phát triển thành công một hệ thống phát hiện hư hỏng đường bộ theo thời gian thực, kết hợp giữa mô hình YOLOv12m đặt tại máy chủ và ứng dụng Android. Hệ thống cho phép thu nhận dữ liệu từ máy ảnh, truyền qua WebSocket đến máy chủ để xử lý và trả về kết quả nhận dạng trong thời gian ngắn. Các thử nghiệm cho thấy hệ thống đáp ứng tốt yêu cầu về tốc độ xử lý và độ chính xác, đảm bảo khả năng hoạt động ổn định trong điều kiện thực tế.

Về mặt ứng dụng, hệ thống được thiết kế để dễ dàng triển khai trên xe khảo sát trang bị thiết bị di động gắn cố định, việc gắn vị trí GPS kèm theo mỗi khung cho phép định vị chính xác vị trí hư hỏng trên bản đồ. Các khung hình được gửi đồng thời kèm metadata gồm thời gian, tọa độ GPS, frameId, trackId giúp thuận tiện cho quá trình theo dõi và tổng hợp báo cáo bảo trì theo từng đoạn đường.

Trong tương lai, hệ thống có thể được cải tiến phát triển hơn thông qua mở rộng tập dữ liệu giúp tăng độ chính xác khi nhận diện. Tối ưu mô hình để xử lý trực tiếp trên thiết bị (chạy mô hình YOLO ngay trên điện thoại). Tích hợp các chức năng đánh giá mức độ hư

hỏng hoặc liên kết với hệ thống quản lý hạ tầng để nâng cao hiệu quả ứng dụng.

Tài liệu tham khảo

[1] Ren, S., He, K., et al. Faster R-CNN: Towards real-time object detection with region proposal networks. Advances in Neural Information Processing Systems (NeurIPS 2015); 2016. 91-99.

[2] Nguyễn Hữu Tuân và Nguyễn Duy Trường Giang. Ứng dụng mạng YOLOv8 trong phát hiện hư hỏng trên mặt đường bộ. Tạp chí Khoa học Công nghệ Hàng hải. 2024; 78: 74-78.

[3] Maeda, H., Sekimoto, Y., Seto, T., et al.Road damage detection using deep neural networks with images captured through a smartphone. Proceedings of the 25th ITS World Congress, Tokyo, Japan; October 2018.

[4] Luo, Y., Wu, A., Fu, Q. MAS-YOLOv11: An improved underwater object detection algorithm based on YOLOv11. Sensors. 2025; 25(11): 3433.

[5] He, Z., Wang, K., et al. Comprehensive performance evaluation of YOLOv11, YOLOv10, YOLOv9, YOLOv8 and YOLOv5 on object detection of power equipment. arXiv preprint, arXiv:2411.18871; 2024.

[6] Alif, M.A.R., Hussain, M. YOLOv12: A breakdown of the key architectural features. arXiv preprint, arXiv:2502.14740v1; 2025.

[7] Tian, Y., Ye, Q., Doermann, D. YOLOv12: Attention-centric real-time object detectors. arXiv preprint, arXiv:2502.12895; 2025.

[8] Hussain, M. Yolov5, yolov8 and yolov10: The go-to detectors for real-time vision. arXiv preprint, arXiv:2407.02988; 2024.

[9] Khanam, R., Hussain, M. YOLOv11: An overview of the key architectural enhancements. arXiv preprint, arXiv:2410.17725; 2024.

[10] Bouraya, S., Belangour, A. Object detectors convolutional neural networks backbones: A review and a comparative study. International Journal. 2021; 9(11).

[11] Islam, M., Jones, A., Faiz, M., et al. Improving performance of breast lesion classification using a ResNet50 model optimized with a novel attention mechanism. Tomography. 2022; 8: 2411-2425.

[12] Zhang, L., Lin, L., et al. Is Faster R-CNN doing well for pedestrian detection? European Conference on Computer Vision (ECCV 2016). Cham: Springer International Publishing; 2016. 443-457.

[13] Cao, C., Wang, B., Zhang, W., et al. An improved Faster R-CNN for small object detection. IEEE Access. 2019; 7: 106838-106846. doi:10.1109/ACCESS.2019.2932731.

[14] Rane, N. YOLO and Faster R-CNN object detection for smart Industry 4.0 and Industry 5.0: Applications, challenges, and opportunities. SSRN Preprint. ID: 4624206; 2023.

[15] Liu, Q., Sun, X. Research of Web real-time communication based on WebSocket. International Journal of Communications, Network and System Sciences. 2012; 5(12): 797-801.

REALTIME ROAD DAMAGE DETECTION

ABSTRACT

A real-time road damage detection system is introduced for mobile deployment using deep learning techniques. Two object detection models Faster R-CNN and YOLOv12m are trained and evaluated on the RDD2022 dataset to determine their suitability for practical application. While Faster R-CNN achieves higher accuracy, its slower inference speed limits responsiveness. YOLOv12m, enhanced with architectural improvements such as R-ELAN, Area Attention, and FlashAttention, offers a better trade-off between speed and accuracy. Based on evaluation results, YOLOv12m is selected and integrated into an Android system, the system integrates the YOLOv12m model into a backend server and connects to the Android application via WebSocket, enabling real-time display of detection results on the user interface.

Keywords: *Android App, faster R-CNN, realTime, road damage detection, YOLOv12*