TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN: XỬ LÍ ẢNH

ĐỒ ÁN: ĐẾM PHƯƠNG TIỆN GIAO THÔNG LƯU THÔNG BẰNG PHƯƠNG PHÁP YOLOv8

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

TRƯỜNG ĐẠI HỌC CÔNG THƯƠNG TP. HCM

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO MÔN: XỬ LÍ ẢNH

ĐỒ ÁN: ĐẾM PHƯƠNG TIỆN GIAO THÔNG LƯU THÔNG BẰNG PHƯƠNG PHÁP YOLOv8

Lớp danh nghĩa: 12DHTH14

TKB chính thức: Thứ 2 – Tiết 4_6

GVHD: Trần Đình Toàn

Sinh viên thực hiện:

Phạm Hồ Đăng Huy

THÀNH PHỐ HỒ CHÍ MINH, NĂM 2024

LỜI CẢM ƠN

Em, Phạm Hồ Đăng Huy, xin gửi lời cảm ơn chân thành đến thầy Trần Đình Toàn, người đã dành thời gian và kiến thức để hướng dẫn và hỗ trợ em trong quá trình hoàn thiện đề tài báo cáo này.

Đồ án kết thúc môn này được hoàn thành dưới sự hướng dẫn một cách chi tiết và khoa học của thầy Trần Đình Toàn. Em muốn bày tỏ lòng biết ơn sâu sắc đến thầy - người đã tận tình giúp đỡ và cung cấp những ý kiến quý báu giúp em hoàn thiện công việc của mình.

Em xin chúc thầy luôn luôn mạnh khỏe và hạnh phúc!

Người thực hiện

(Ký tên)

Phạm Hồ Đăng Huy

MỤC LỤC HÌNH ẢNH

MỤC LỤC

CHƯƠNG I. GIỚI THIỆU

1. Lý do chọn đề tài

Việc quản lý và điều phối mật độ giao thông đóng vai trò quan trọng trong việc duy trì an toàn và hiệu suất của hệ thống giao thông đô thị. Sự cải thiện và tự động hóa quy trình đếm lưu lượng giao thông có thể cung cấp thông tin quan trọng để hiểu và dự đoán mẫu di chuyển của phương tiện trên đường. Điều này sẽ hỗ trợ trong việc thiết kế và triển khai các biện pháp điều phối giao thông một cách hiệu quả hơn.

2. Mục tiêu của báo cáo

Mục tiêu chính của báo cáo này là áp dụng phương pháp YOLOv8 để đếm lưu lượng giao thông trên đường và từ đó cung cấp thông tin cần thiết cho quy trình điều phối mật độ giao thông. Bằng việc tự động hóa quá trình đếm và phân tích lưu lượng giao thông, chúng tôi hy vọng có thể cung cấp cho các cơ quan quản lý giao thông một công cụ hữu ích để hiểu rõ hơn về tình hình giao thông và đưa ra các biện pháp điều phối phù hợp để cải thiện hiệu suất và an toàn giao thông.

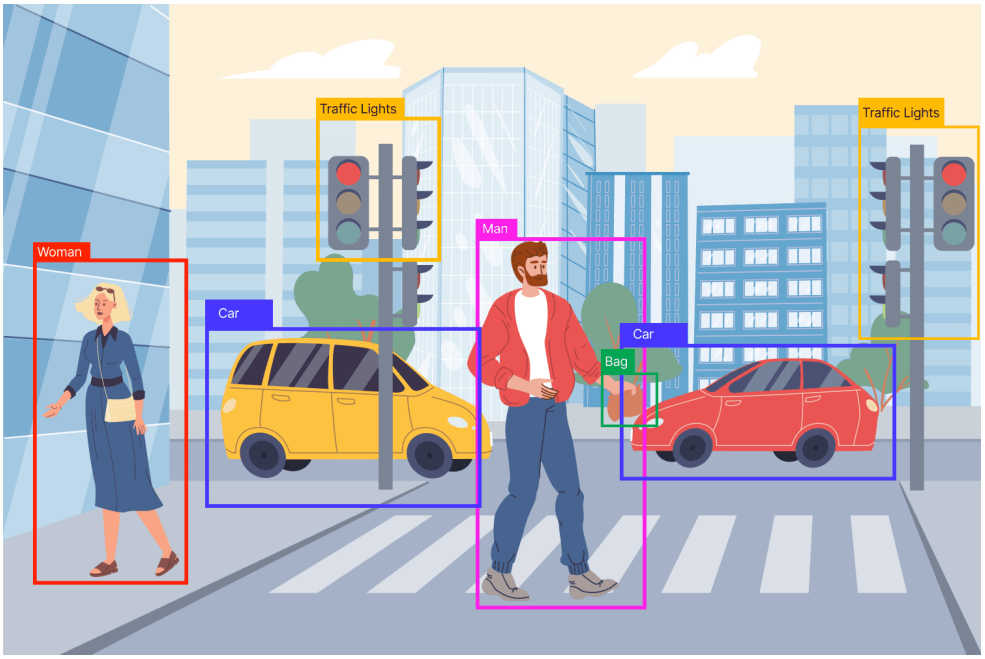
CHƯƠNG II. PHÁT HIỆN ĐỐI TƯỢNG BẰNG PHƯƠNG PHÁP YOLO

1. Tổng quan về phát hiện đối tượng trong thị giác máy tính

1.1. Phát hiện đối tượng là gì?

Phát hiện đối tượng là quá trình nhận diện và xác định vị trí của các đối tượng trong hình ảnh hoặc video. Trong lĩnh vực thị giác máy tính, việc này đóng vai trò quan trọng trong nhiều ứng dụng như phát hiện kẻ xấu trong hệ thống giám sát an ninh, phát hiện biển báo giao thông trong hệ thống hỗ trợ lái xe, và đặc biệt là đếm lưu lượng phương tiện giao thông trên đường để quản lý giao thông và dự đoán tình trạng đường.

Các thuật toán phát hiện đối tượng có thể được chia thành hai loại chính: phát hiện một giai đoạn (Single-shot object detection) và phát hiện hai giai đoạn (Two-shot object detection).



Hình 1. Object Detection

1.2. Các loại thuật toán phát hiện đối tượng

Có hai loại chính của thuật toán phát hiện đối tượng:

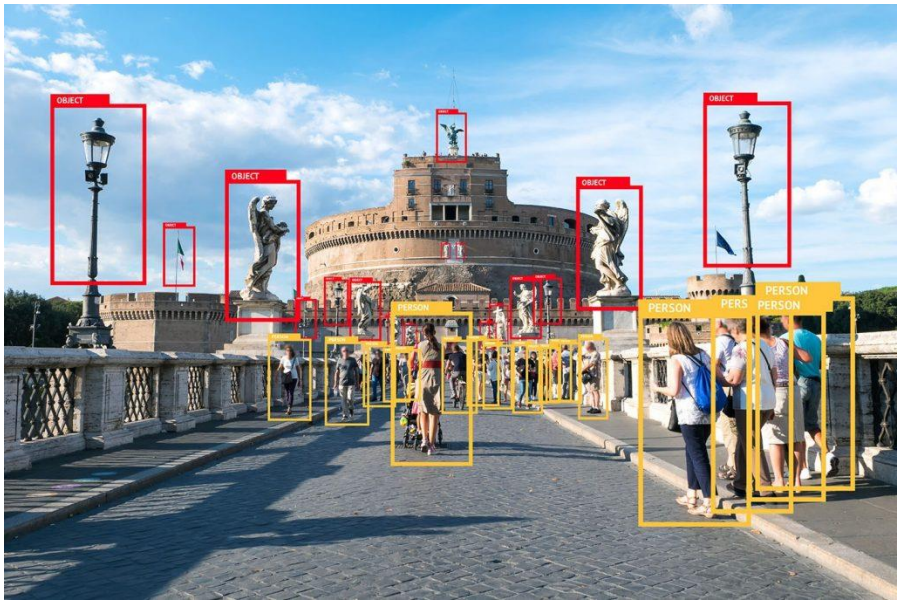
– **Phát hiện một giai đoạn (Single-shot object detection)**: Phương pháp này sử dụng một lần truyền hình ảnh đầu vào để đưa ra dự đoán về sự hiện diện và vị trí của các đối tượng trong ảnh. Điều này đồng nghĩa với việc

xử lý toàn bộ hình ảnh trong một lần chạy, giúp tối ưu hóa hiệu suất tính toán. Tuy nhiên, single-shot object detection thường kém chính xác hơn so với phương pháp hai giai đoạn và khó khăn hơn trong việc nhận diện các đối tượng nhỏ hoặc gần nhau. YOLO là thuật toán single-shot object detection sử dụng mạng thần kinh tích chập (CNN) để xử lý hình ảnh.

– **Phát hiện hai giai đoạn (Two-shot object detection)**: Phương pháp này sử dụng hai lần truyền hình ảnh đầu vào để đưa ra dự đoán về sự hiện diện và vị trí của đối tượng. Lần đầu tiên được sử dụng để tạo ra một tập hợp các đề xuất hoặc vị trí tiềm năng của đối tượng, sau đó lần thứ hai được sử dụng để tinh chỉnh các đề xuất này và đưa ra dự đoán cuối cùng. Cách tiếp cận này chính xác hơn single-shot object detection nhưng cũng tốn kém hơn về mặt tính toán.

⇒ Nhìn chung, Single-shot object detection phù hợp hơn cho các ứng dụng thời gian thực, trong khi two-shot object detection tốt hơn cho các ứng dụng đề cao độ chính xác.

- Ngoài ra còn có các thuật toán nhận dạng đối tượng phổ biến khác như R-CNN (Region-based Convolutional Neural Networks) và các biến thể của nó như Fast R-CNN, Faster R-CNN; SSD (Single Shot Multibox Detector).



Hình 2. Tool Object Detection

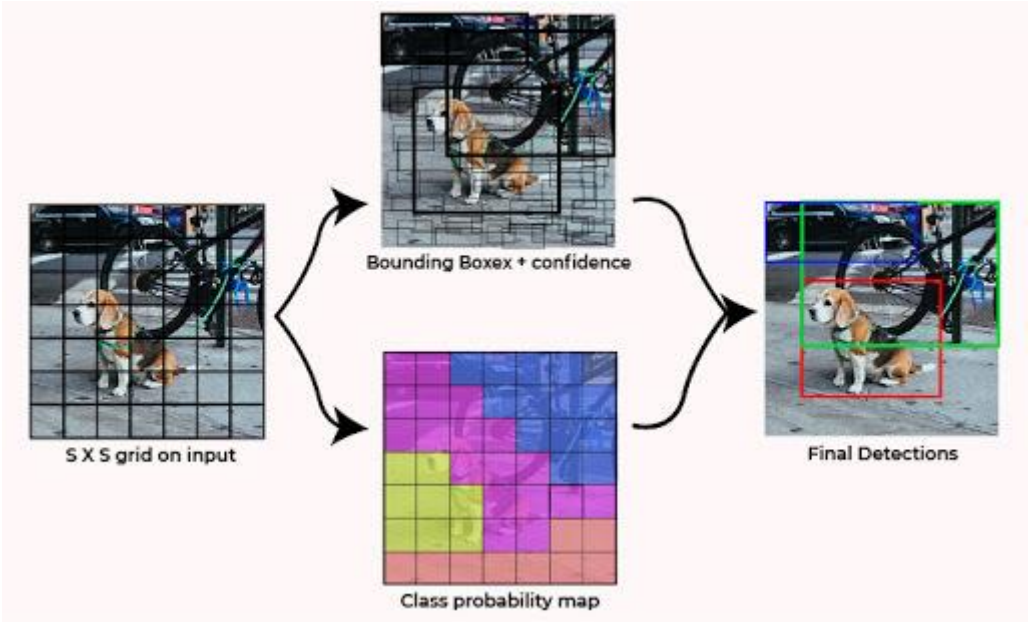
2. Giới thiệu về phương pháp YOLO

2.1. YOLO là gì?

YOLO (You Only Look Once) là một phương pháp tiên tiến trong lĩnh vực phát hiện đối tượng trong thị giác máy tính. Điểm nổi bật của YOLO là sự đề xuất sử dụng mạng thần kinh đầu cuối để đồng thời đưa ra dự đoán về các hộp giới hạn (bounding box) và xác suất của các đối tượng trong một lúc. Khác với các thuật toán trước đây, YOLO không sử dụng lại các bước phân loại riêng biệt mà thực hiện cả hai nhiệm vụ này trong một lần duy nhất.

Trong khi các thuật toán trước đây như Faster R-CNN tập trung vào việc phát hiện các khu vực quan trọng trong hình ảnh thông qua Region Proposal Network và sau đó thực hiện phân loại trên từng khu vực đó, YOLO tiếp cận bài toán theo một cách hoàn toàn khác. YOLO thực hiện tất cả các dự đoán một cách toàn diện chỉ thông qua một lớp mạng được kết nối đầy đủ.

Điểm mạnh của YOLO là khả năng thực hiện dự đoán trong một lần chạy duy nhất, giúp tối ưu hóa hiệu suất tính toán và tăng tốc độ xử lý. Điều này khác biệt lớn so với các phương pháp sử dụng Region Proposal Network, cần thực hiện nhiều lần lặp trên cùng một hình ảnh.



Hình 3. YOLO

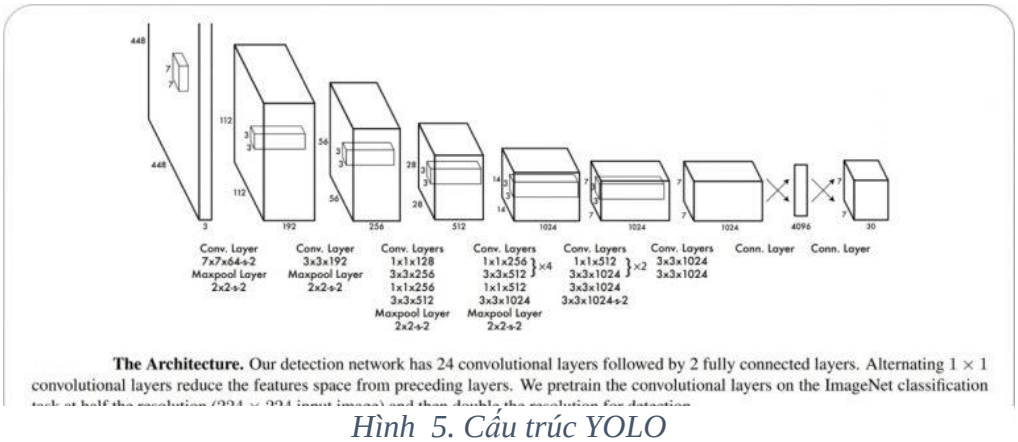
2.2. Lịch sử phát triển của YOLO

Kể từ khi được giới thiệu lần đầu vào năm 2015 bởi Joseph Redmon và các đồng nghiệp, YOLO (You Only Look Once) đã trải qua nhiều cải tiến và phiên bản mới đã được giới thiệu để cải thiện hiệu suất và độ chính xác của mô hình. Sự tiến bộ này thể hiện sự phát triển không ngừng của YOLO trong những năm gần đây. Được biết đến với khả năng phát hiện đối tượng trong thời gian thực với độ chính xác cao, YOLO đã thu hút sự chú ý lớn trong cộng đồng thị giác máy tính. Dưới đây là mốc thời gian thể hiện sự phát triển của YOLO trong những năm gần đây.

Hình 4. YOLO timeline

2.3. Kiến trúc và nguyên lý hoạt động của YOLO

Thuật toán YOLO lấy hình ảnh làm đầu vào, sau đó sử dụng mạng nơ-ron tích chập sâu đơn giản để phát hiện các đối tượng trong ảnh. Kiến trúc của mô hình CNN tạo thành xương sống của YOLO được hiển thị bên dưới.



Hình 5. Cấu trúc YOLO

Mô hình YOLO (You Only Look Once) bao gồm 20 lớp tích chập ban đầu, được đào tạo trước với tập dữ liệu ImageNet. Trong quá trình này, một lớp tổng hợp trung bình tạm thời và một lớp kết nối đầy đủ được thêm vào để chuyển đổi mô hình để có thể thực hiện phát hiện đối tượng. Lớp kết nối đầy đủ cuối cùng của YOLO đưa ra dự đoán về xác suất của các lớp và tọa độ của các hộp giới hạn.

Trong quá trình phát hiện, YOLO chia hình ảnh đầu vào thành một lưới ô có kích thước S × S. Mỗi ô trong lưới này đảm nhận nhiệm vụ phát hiện các đối tượng bằng cách dự đoán các hộp giới hạn và điểm tin cậy tương ứng. Các điểm tin cậy này phản ánh mức độ tin cậy của mô hình về sự hiện diện của đối tượng trong hộp và mức độ chính xác của dự đoán.

Một đặc điểm quan trọng của YOLO là khả năng dự đoán nhiều hộp giới hạn trên mỗi ô trong lưới. Tuy nhiên, trong quá trình đào tạo, chỉ có một bộ dự đoán hộp giới hạn được chọn dựa trên chỉ số IOU (Intersection over Union) cao nhất với thực tế. Điều này giúp tối ưu hóa độ chính xác và hiệu suất của mô hình bằng cách tạo ra các dự đoán chuyên biệt cho từng đối tượng.

Một kỹ thuật quan trọng được áp dụng trong YOLO là NMS (non-maximum suppression). NMS được sử dụng để loại bỏ các hộp giới hạn dư thừa hoặc trùng lặp và chỉ giữ lại các hộp giới hạn duy nhất và chính xác nhất cho mỗi đối tượng trong hình ảnh. Điều này giúp cải thiện độ chính xác và hiệu suất của quá trình phát hiện đối tượng.

2.4. Ưu điểm và nhược điểm của YOLO

Ưu điểm: YOLO là một phương pháp đơn giản và hiệu quả, cho phép phát hiện đối tượng trong thời gian thực và độ chính xác cao.

Nhược điểm: YOLO có thể gặp khó khăn trong việc nhận diện các đối tượng nhỏ hoặc gần nhau, cũng như trong việc xử lý các bối cảnh phức tạp.

2.5. Điểm cải tiến của các phiên bản YOLO

2.5.1. YOLO v2

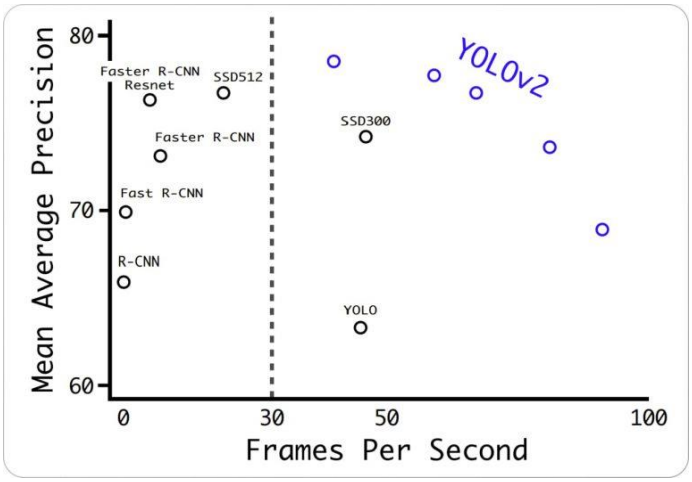
YOLO v2, hay còn gọi là YOLO9000, ra mắt vào năm 2016 như một bước tiến so với phiên bản ban đầu của thuật toán YOLO. Mục tiêu của YOLO v2 là tăng cường tốc độ và độ chính xác so với phiên bản trước đó, đồng thời có khả năng phát hiện được nhiều loại đối tượng hơn. Để đạt được điều này, phiên bản cập nhật này áp dụng một kiến trúc CNN mới được gọi là Darknet-19, một biến thể của kiến trúc VGGNet với các lớp convolution tiến triển và các lớp pooling đơn giản.

Một trong những cải tiến quan trọng nhất trong YOLO v2 là sử dụng các anchor boxes. Anchor boxes là một tập hợp các hộp giới hạn được định nghĩa trước với các tỷ lệ và tỷ lệ khung hình khác nhau. Khi dự đoán các hộp giới hạn, YOLO v2 kết hợp các anchor boxes và sự điều chỉnh được dự đoán để xác định hộp giới hạn cuối cùng. Điều này giúp thuật toán xử lý một loạt các kích thước và tỷ lệ khung hình của các đối tượng một cách linh hoạt.

Cải tiến khác trong YOLO v2 là sử dụng chuẩn hóa hàng loạt, giúp cải thiện độ chính xác và ổn định của mô hình. YOLO v2 cũng áp dụng chiến lược đào tạo đa quy mô bằng cách đào tạo mô hình trên hình ảnh với nhiều tỷ lệ khác nhau và sau đó kết hợp các dự đoán từ các tỷ lệ này. Điều này giúp nâng cao hiệu suất phát hiện các đối tượng nhỏ.

YOLO v2 cũng giới thiệu một hàm mất mát – loss function mới phù hợp hơn với tác vụ phát hiện đối tượng. Hàm mất mát này dựa trên tổng các lỗi bình phương giữa các hộp giới hạn dự đoán và thực tế cũng như xác suất của các lớp.

Kết quả thu được từ YOLO v2 so với phiên bản gốc và các mô hình hiện đại khác được hiển thị bên dưới.



Hình 6. Kết quả của YOLO v2

2.5.2. YOLO v3

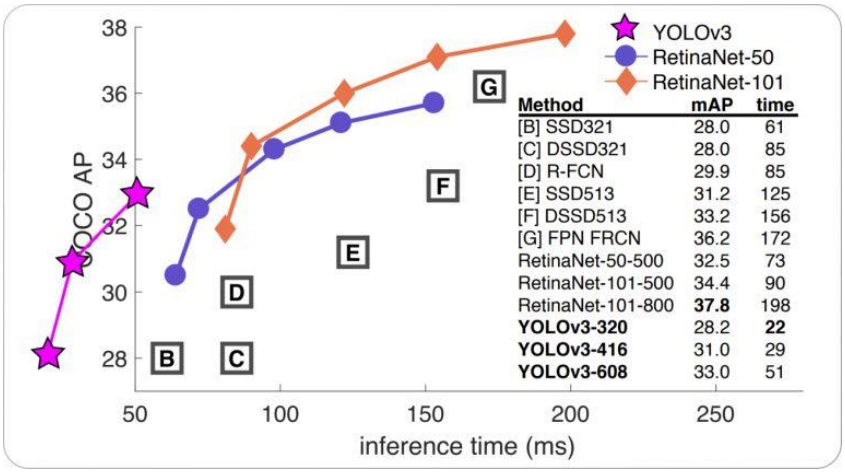
Phiên bản thứ ba của thuật toán phát hiện đối tượng YOLO, hay YOLO v3, ra đời vào năm 2018 với mục tiêu cải thiện đáng kể về độ chính xác và tốc độ so với phiên bản YOLO v2.

Một trong những điểm nổi bật nhất trong YOLO v3 là việc áp dụng kiến trúc CNN mới được gọi là Darknet-53. Đây là một biến thể của kiến trúc ResNet, được tối ưu hóa đặc biệt cho các nhiệm vụ phát hiện đối tượng, với tổng cộng 53 lớp tích chập. Darknet-53 mang lại sự tiến bộ đáng kể trên nhiều tiêu chuẩn phát hiện đối tượng khác nhau.

Cải tiến khác trong YOLO v3 là sử dụng các anchor box với các tỷ lệ và tỷ lệ khung hình đa dạng. Trong phiên bản YOLO v2, các anchor box có kích thước cố định, điều này đã hạn chế khả năng phát hiện các đối tượng có kích thước và hình dạng đa dạng. Trong YOLO v3, việc biến đổi tỷ lệ và tỷ lệ khung hình của các anchor box giúp mô hình phát hiện được đa dạng hơn.

YOLO v3 cũng giới thiệu khái niệm "feature pyramid networks" (FPN), một kiến trúc CNN được sử dụng để phát hiện các đối tượng ở nhiều tỷ lệ khác nhau. FPN xây dựng một kim tự tháp đặc trưng, mỗi cấp độ của kim tự tháp được sử dụng để phát hiện đối tượng ở một tỷ lệ khác nhau. Điều này giúp cải thiện hiệu suất phát hiện trên các đối tượng nhỏ, vì mô hình có thể nhìn thấy các đối tượng ở nhiều tỷ lệ khác nhau.

Ngoài những cải tiến nêu trên, YOLO v3 còn có khả năng xử lý đa dạng hơn về kích thước đối tượng và tỷ lệ khung hình. Đồng thời, nó cũng chính xác và ổn định hơn so với các phiên bản trước của YOLO.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Hình 7. Kết quả YOLO v3

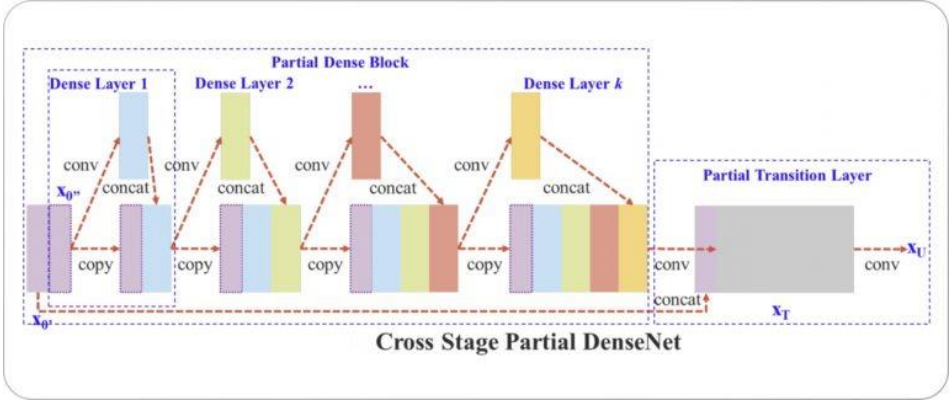
2.5.3. YOLO v4

Chú ý: Joseph Redmon, người sáng tạo ban đầu của YOLO, đã rời khỏi cộng đồng Trí tuệ Nhân tạo vài năm trước, vì vậy YOLOv4 và các phiên bản trước đó không được xem là tác phẩm chính thức của ông. Một số trong số chúng được duy trì bởi các đồng tác giả, nhưng không có bản phát hành nào trước đó của YOLOv3 được coi là YOLO "chính thức".

YOLO v4, ra đời vào năm 2020 bởi Bochkovskiy và đồng nghiệp, đánh dấu một bước tiến mới so với YOLO v3.

Một trong những cải tiến chính của YOLO v4 so với YOLO v3 là sự áp dụng của kiến trúc CNN mới, được gọi là CSPNet (Cross Stage Partial Network). CSPNet là một biến thể của kiến trúc ResNet, được tối ưu hóa cho nhiệm vụ phát hiện đối tượng, với một cấu trúc tương đối đơn giản với chỉ 54 lớp chập. Tuy nhiên, nó mang lại hiệu suất tiên tiến trên nhiều tiêu chuẩn phát hiện đối tượng khác nhau. Cả YOLO v3 và YOLO v4 đều sử dụng các anchor box có tỷ lệ và tỷ lệ khung hình đa dạng để phù hợp với kích thước và hình dạng của các đối tượng được phát hiện. YOLO v4 giới thiệu một phương pháp mới để tạo ra các anchor box, gọi là "k-means clustering", dựa trên việc nhóm các hộp giới hạn thực tế

thành các cụm và sử dụng trọng tâm của các cụm làm anchor box. Điều này giúp cải thiện sự căn chỉnh chặt chẽ của anchor box với kích thước và hình dạng của các đối tượng.



Hình 8. CSPNet

Mặc dù cả YOLO v3 và YOLO v4 đều sử dụng hàm mất mát tương tự để đào tạo mô hình, nhưng YOLO v4 giới thiệu một thuật ngữ mới gọi là "GHM loss", một biến thể của hàm mất mát focal, được thiết kế để cải thiện hiệu suất của mô hình trên các bộ dữ liệu mất cân bằng. Ngoài ra, YOLO v4 cũng cải thiện kiến trúc FPN được sử dụng trong YOLO v3.

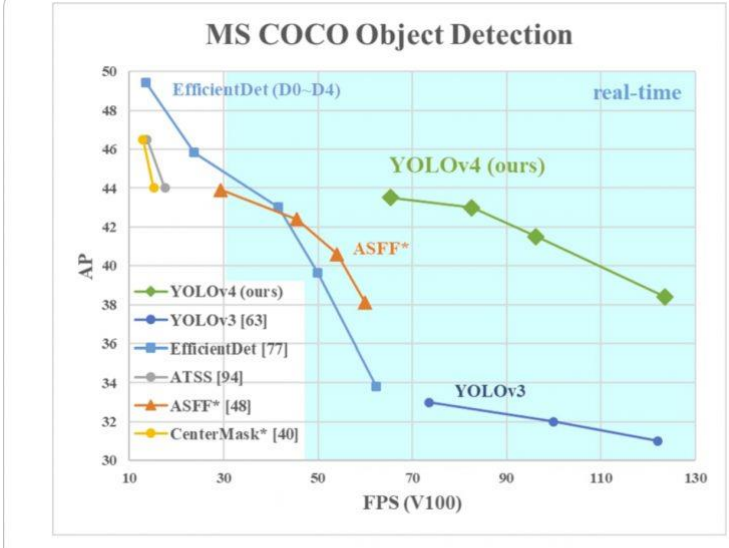


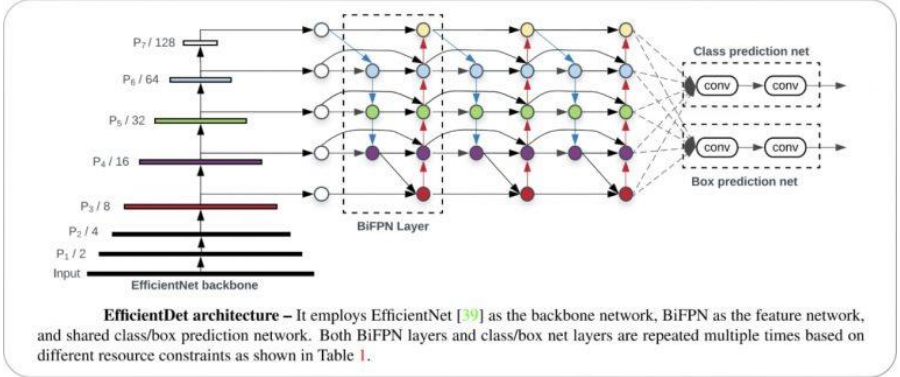
Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Hình 9. YOLO v4

2.5.4. YOLO v5

YOLO v5, ra mắt vào năm 2020 bởi nhóm phát triển gốc của YOLO và được duy trì bởi Ultralytics, đánh dấu một bước tiến mới trong thế giới phát hiện đối tượng. Dự án này, là một dự án mã nguồn mở, tiếp tục truyền lửa từ sự thành công của các phiên bản trước đó và đem lại nhiều tính năng và cải tiến mới.

Trong khi YOLO v5 là một sự tiếp nối, nhưng nó sử dụng một kiến trúc phức tạp hơn được gọi là EfficientDet, dựa trên mạng EfficientNet. Điều này giúp YOLO v5 đạt được độ chính xác cao hơn và khả năng khái quát hóa tốt hơn cho nhiều loại đối tượng.



EfficientDet architecture – It employs EfficientNet [39] as the backbone network, BiFPN as the feature network, and shared class/box prediction network. Both BiFPN layers and class/box net layers are repeated multiple times based on different resource constraints as shown in Table 1.

Hình 10. EfficientNet

Sự khác biệt nổi bật giữa YOLO và YOLO v5 là dữ liệu đào tạo được sử dụng. Trong khi YOLO được đào tạo trên bộ dữ liệu PASCAL VOC với 20 danh mục đối tượng, YOLO v5 sử dụng tập dữ liệu lớn hơn và đa dạng hơn, được biết đến là D5, bao gồm tổng cộng 600 danh mục đối tượng.

YOLO v5 giới thiệu một phương pháp mới để tạo các anchor box, được gọi là "dynamic anchor boxes". Điều này giúp cải thiện sự căn chỉnh của anchor box với kích thước và hình dạng của các đối tượng.

Ngoài ra, YOLO v5 còn đưa vào khái niệm "spatial pyramid pooling" (SPP), một loại lớp tổng hợp được sử dụng để cải thiện hiệu suất phát hiện trên các đối tượng nhỏ. SPP cho phép mô hình nhìn thấy các đối tượng ở nhiều tỷ lệ khác nhau, giúp cải thiện hiệu suất phát hiện.

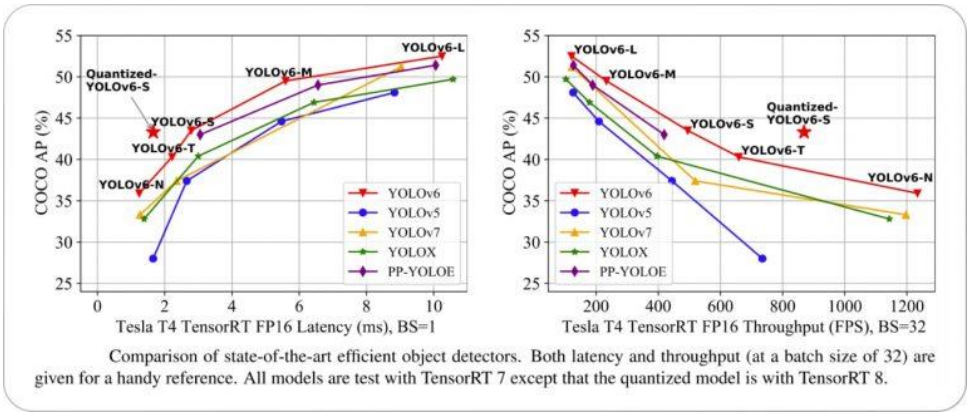
Cả YOLO v4 và YOLO v5 sử dụng hàm mất mát tương tự để huấn luyện mô hình, nhưng YOLO v5 giới thiệu một thuật ngữ mới gọi là "CIoU loss", một biến thể của IoU loss function, được thiết kế để cải thiện hiệu suất của mô hình trên các bộ dữ liệu mất cân bằng.

2.5.5. YOLO v6

YOLO v6, đề xuất vào năm 2022 bởi Li và cộng sự, đánh dấu một bước tiến mới trong phát triển của thuật toán phát hiện đối tượng. Một trong những điểm đặc biệt giữa YOLO v5 và YOLO v6 là sự thay đổi trong kiến trúc mạng CNN. Trong YOLO v6, một biến thể của kiến trúc EfficientNet được sử dụng, được gọi là EfficientNet-L2. Kiến trúc này được đánh giá là hiệu quả hơn so với EfficientDet, được sử dụng trong YOLO v5, với ít tham số hơn và tính toán hiệu quả hơn. Với khả năng đạt được kết quả tiên tiến trên nhiều tiêu chuẩn phát hiện đối tượng khác nhau, EfficientNet-L2 mang lại sự cải tiến đáng kể cho YOLO v6.

Mô hình YOLO v6 được tạo ra dựa trên một framework mới, được thể hiện trong hình dưới đây. Điểm đáng chú ý là YOLO v6 giới thiệu một phương pháp mới để tạo ra các anchor box, được gọi là "dense anchor boxes".

Kết quả thu được từ YOLO v6 so với các phương pháp hiện đại khác đã được thể hiện và sẽ được trình bày dưới đây.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

Hình 11. EfficientNet-L2

2.5.6. YOLO v7

YOLO v7, là phiên bản mới nhất của YOLO, đem lại nhiều cải tiến so với các phiên bản trước, đặc biệt là trong việc sử dụng các anchor box.

Các anchor box là một tập hợp các hộp giới hạn được xác định trước, với các tỷ lệ khung hình khác nhau, giúp trong việc phát hiện các đối tượng có hình dạng đa dạng. YOLO v7 sử dụng chín anchor box, mở rộng phạm vi của việc xác định hình dạng và kích thước của các đối tượng. Điều này giúp giảm số lượng xác định sai.

Một điểm cải tiến quan trọng trong YOLO v7 là việc áp dụng một loss function mới được gọi là "focal loss". Trong quá khứ, YOLO đã sử dụng cross-entropy loss function, nhưng focal loss giải quyết vấn đề phát hiện các đối tượng nhỏ bằng cách tập trung vào việc giảm trọng số mất mát cho các ví dụ được phân loại chính xác và tập trung vào việc xử lý các ví dụ khó - những đối tượng khó phát hiện.

Độ phân giải cao hơn cũng là một điểm đáng chú ý trong YOLO v7. Với việc xử lý hình ảnh ở độ phân giải 608 x 608 pixel, cao hơn so với 416 x 416 được sử dụng trong YOLO v3, nó có khả năng phát hiện các đối tượng nhỏ và đạt được độ chính xác tổng thể cao hơn.

Một trong những ưu điểm lớn nhất của YOLO v7 là tốc độ xử lý. Với khả năng xử lý hình ảnh lên đến 155 khung hình mỗi giây, nhanh hơn nhiều so với các thuật toán phát hiện đối tượng hiện đại khác, nó trở thành một lựa chọn phù hợp cho các ứng dụng thời gian thực như giám sát và ô tô tự lái, nơi tốc độ xử lý là rất quan trọng.

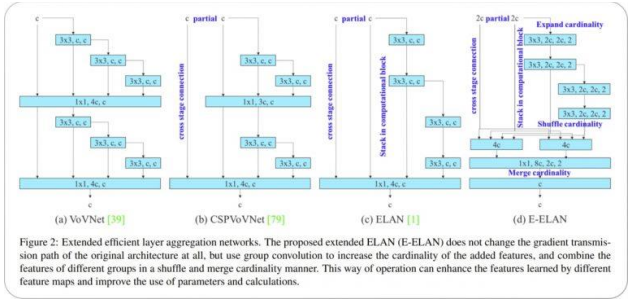


Figure 2: Extended efficient layer aggregation networks. The proposed extended ELAN (E-ELAN) does not change the gradient transmission path of the original architecture at all, but use group convolution to increase the cardinality of the added features, and combine the features of different groups in a shuffle and merge cardinality manner. This way of operation can enhance the features learned by different feature maps and improve the use of parameters and calculations.

Hình 12. YOLO v7

YOLO v7, dù mạnh mẽ và hiệu quả, vẫn tồn tại một số hạn chế như sau:

– Phát hiện đối tượng nhỏ còn hạn chế: YOLOv7 gặp khó khăn trong việc phát hiện các đối tượng nhỏ hoặc ở khoảng cách xa. Điều này đặc biệt quan trọng trong các tình huống đông đúc hoặc khi đối tượng cần phát hiện không nổi bật trong bối cảnh.

– Khó khăn với tỷ lệ và kích thước khung hình khác nhau: YOLOv7 có thể gặp khó khăn trong việc phát hiện đối tượng ở các tỷ lệ khung hình khác nhau. Điều này đôi khi gây khó khăn khi có sự biến đổi về kích thước giữa các đối tượng trong cùng một hình ảnh.

– Nhạy cảm với biến đổi ánh sáng và môi trường: YOLOv7 có thể không ổn định khi ánh sáng hoặc điều kiện môi trường thay đổi. Nó có thể dẫn đến việc không thể phát hiện đối tượng hoặc tạo ra các dự đoán không chính xác trong điều kiện môi trường biến đổi.

– Yêu cầu tính toán nhiều: YOLOv7 yêu cầu nhiều tính toán để xử lý ảnh, điều này có thể gây khó khăn khi cố gắng chạy thuật toán trong thời gian thực trên các thiết bị có tài nguyên hạn chế như điện thoại thông minh. Việc cải thiện tốc độ suy luận có thể đòi hỏi sự đầu tư vào phần cứng mạnh mẽ.

3. Tổng quan về YOLO v8

3.1. Sơ lược về YOLO v8

YOLO v8, phiên bản mới nhất của dòng mô hình YOLO, đại diện cho một tiến bộ đáng kể trong lĩnh vực phát hiện đối tượng.

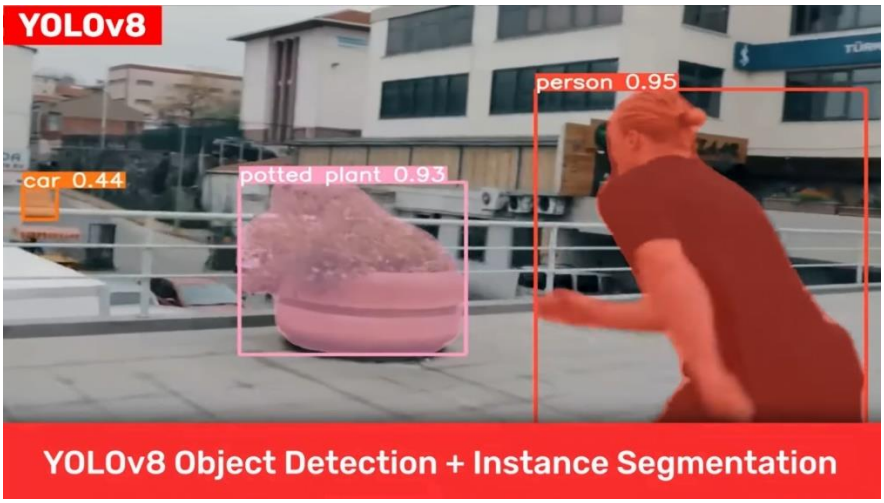
Mô hình YOLO được huấn luyện trên các bộ dữ liệu lớn như COCO và ImageNet, giúp cung cấp dự đoán chính xác và đồng thời học các lớp mới một cách hiệu quả. Tính linh hoạt và khả năng huấn luyện nhanh chóng của các mô hình YOLO trên GPU đơn cũng là một điểm nổi bật, tạo điều kiện thuận lợi cho nhà phát triển.

YOLO v8, ra mắt vào đầu năm 2023, đem lại nhiều cải tiến so với các phiên bản trước, bao gồm phát hiện không sử dụng anchor, giới thiệu lớp tích

chập C3 và tăng cường mosaic. Đặc biệt, YOLOv8 được phát triển và duy trì bởi nhóm Ultralytics, là một mô hình SOTA mã nguồn mở, phân phối dưới Giấy phép Công cộng GNU, đồng thời được sự đóng góp tích cực từ cộng đồng người dùng.

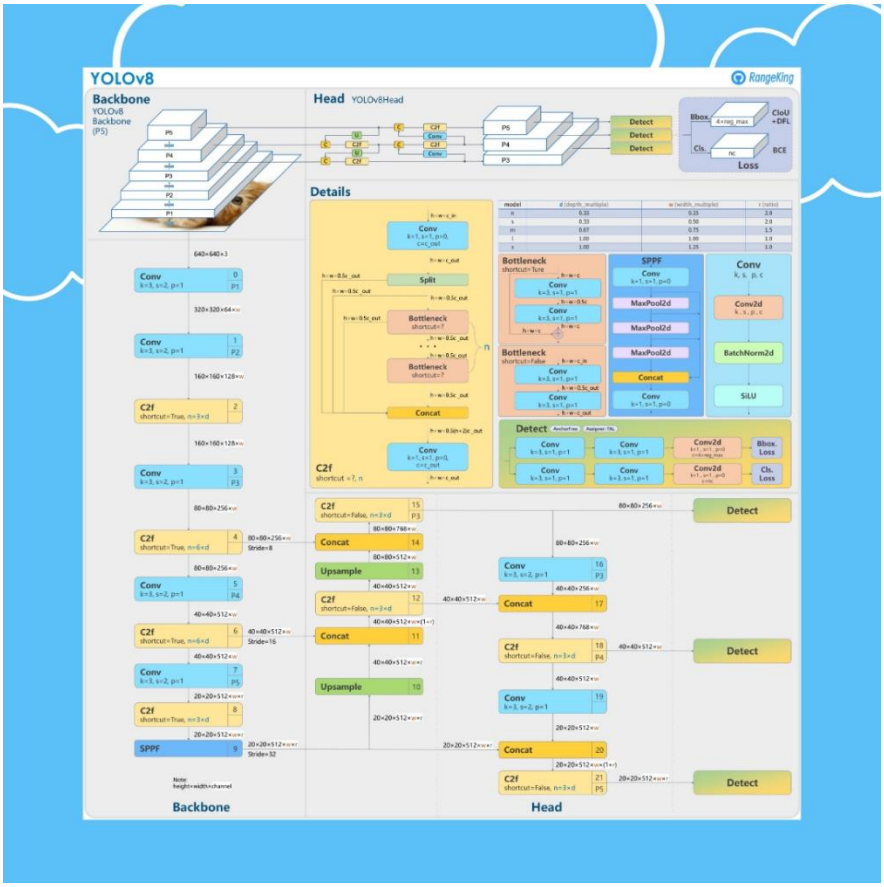
Tác giả của YOLO v8 là nhóm Ultralytics, người đã xây dựng và duy trì mô hình này. Trong quá trình phát triển, từ các phiên bản YOLO ban đầu của Joseph Redmon, qua YOLOv3 và YOLOv5 của Glenn Jocher, kiến trúc của YOLOv8 đã trải qua sự điều chỉnh và tối ưu để đạt được hiệu suất tốt nhất.

Với sự ra đời của YOLO v8 vào ngày 10 tháng 1 năm 2023, mô hình này vẫn đang tiếp tục được nghiên cứu và phát triển tích cực, hứa hẹn mang lại những đóng góp ý nghĩa cho lĩnh vực phát hiện đối tượng trong tương lai.



Hình 13. YOLO v8

3.2. Một số thông số kỹ thuật quan trọng và kết quả



Hình 14. Kiến trúc YOLO v8

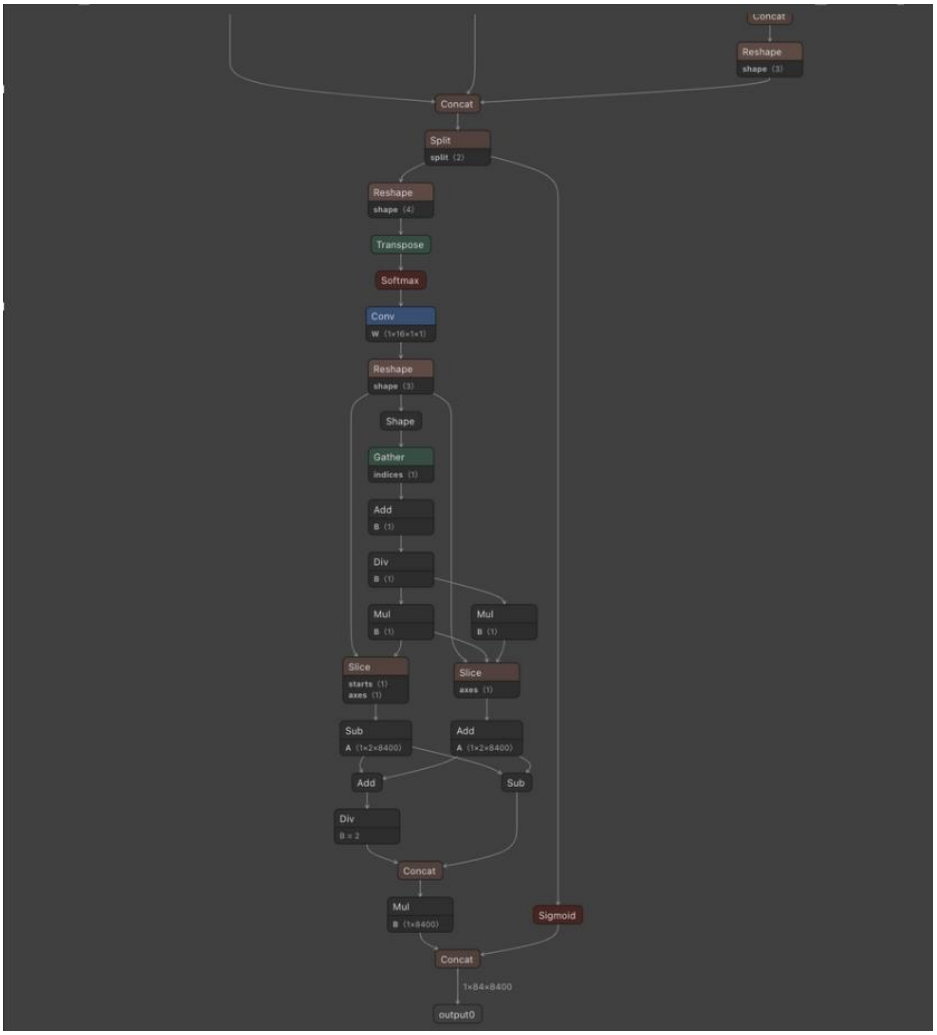
Chúng ta cũng có thể tưởng tượng kiến trúc của YOLOv8 thông qua việc chuyển đổi nó sang định dạng ONNX. ONNX, viết tắt của Open Neural Network Exchange, là một định dạng mở được dùng để biểu diễn các mô hình học máy. Nó có thể được coi như một phương tiện giao tiếp chung cho mọi loại mô hình ML, bởi vì tính phổ biến của nó cho bất kỳ mô hình nào, dù code mô hình được viết trong PyTorch, TensorFlow hoặc các framework nào khác.

Việc chuyển đổi bất kỳ mô hình YOLO nào sang định dạng ONNX cũng khá dễ dàng. Bằng cách sao chép và code dưới đây :

```
!pip install ultralytics

from ultralytics import YOLO
model = YOLO('yolov8m.pt')
model.export(format = "onnx")
```

Hình 15. Chuyển định dạng của mô hình YOLO sang ONNX

Sau khi chạy dòng code thì ta có thể lấy tệp .onnx này và tải lên ứng dụng Netron. Netron là một ứng dụng giúp hiển thị hình ảnh của các mạng thần kinh.



Hình 16. Kiến trúc YOLO v8 trong mạng Neuron

Từ đây, ta có thể so sánh những thay đổi trong YOLOv8 với phiên bản trước của nó, YOLOv5. Có hai thay đổi chính:

- Anchor-Free Detection
- Mosaic Augmentation

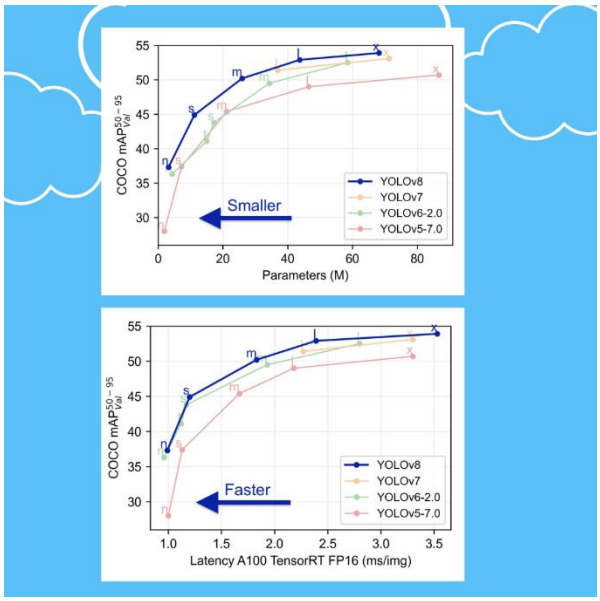
Chúng ta sẽ tìm hiểu sự thay đổi này ở phần sau.

3.3. YOLO v8 có gì cải tiến so với các phiên bản trước đó?

Như chúng ta có thể thấy từ biểu đồ, YOLOv8 có nhiều tham số hơn so với các phiên bản tiền nhiệm như YOLOv5, nhưng ít tham số hơn so với

YOLOv6. Nó cung cấp khoảng 33% mAP nhiều hơn cho các mô hình kích thước n và mAP lớn hơn nói chung.

Từ biểu đồ thứ hai, chúng ta có thể thấy YOLOv8 có thời gian suy luận nhanh hơn so với tất cả các phiên bản YOLO khác.



Hình 17. Tính toán của YOLO v8

Trong YOLOv8, ta có các kích thước mô hình khác nhau như yolov8- n – nano, s – small, m – medium, l – large và x – extra large.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	0.99	3.2	8.7
YOLOv8s	640	44.9	128.4	1.20	11.2	28.6
YOLOv8m	640	50.2	234.7	1.83	25.9	78.9
YOLOv8l	640	52.9	375.2	2.39	43.7	165.2
YOLOv8x	640	53.9	479.1	3.53	68.2	257.8

Hình 18. Các mô hình kích thước của YOLO v8

Kích thước mô hình tương ứng tuyến tính với mAP và nghịch đảo tương ứng với thời gian suy luận. Các mô hình lớn mất nhiều thời gian suy luận để phát hiện đối tượng một cách chính xác với mAP cao hơn. Các mô hình nhỏ có thời

gian suy luận nhanh hơn nhưng có mAP tương đối thấp hơn. Các mô hình lớn hơn tốt hơn nếu chúng ta có ít dữ liệu. Các mô hình nhỏ hơn hiệu quả hơn nếu ta có ít không gian (các tình huống biên).

- Ngoài ra còn các sự thay đổi khác ở YOLO v8 so với YOLO v5 như đã đề cập ở phần trước đó, là Anchor-Free Detection và Mosaic Augmentation.

- Anchor-Free Detection

Để hiểu về Anchor-Free Detection, hãy khám phá sâu hơn về anchor boxes, một khái niệm quan trọng trong phát hiện đối tượng.

Trước khi có anchor boxes, việc xác định các đối tượng trong ảnh đòi hỏi phải gán mỗi đối tượng vào một ô lưới dựa trên điểm trung tâm của nó. Tuy nhiên, khi hai đối tượng chia sẻ cùng một điểm trung tâm, việc xác định và phân loại chúng thành các lớp khác nhau trở nên phức tạp.

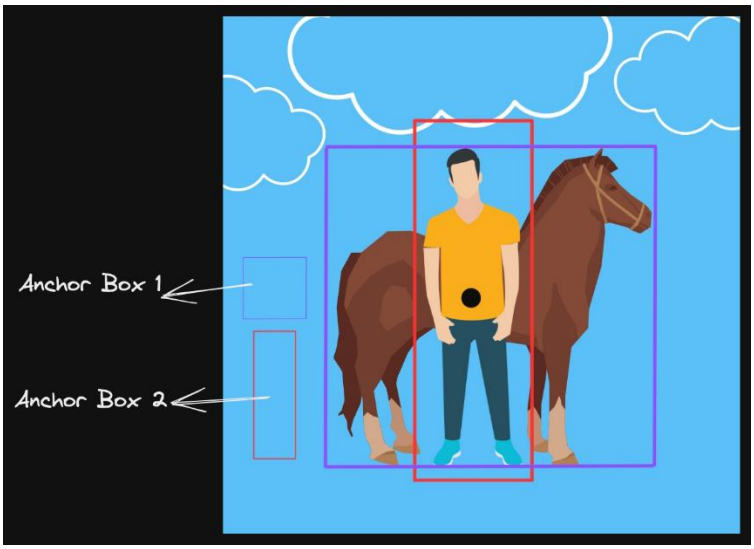
Hãy tưởng tượng một tình huống trong đó một con người và một con ngựa có cùng tọa độ trung tâm. Trong trường hợp này, việc xác định và xây dựng các hộp giới hạn cho từng đối tượng một cách chính xác trở nên khó khăn. Anchor boxes đã giúp giải quyết vấn đề này bằng cách cung cấp các điểm tham chiếu để xác định và phân loại đối tượng dễ dàng hơn.

Hình 19. Ví dụ Anchor Boxes

Anchor boxes có thể được ví như các mẫu cookie-cutter trong quá trình phát hiện đối tượng. Giả sử chúng ta có hai anchor boxes: Anchor Box 1 và Anchor Box 2.

Bước tiếp theo là kiểm tra từ danh sách các anchor boxes xem anchor boxes nào có chỉ số IoU (Intersection over Union) cao nhất với hộp giới hạn thực tế và sau đó gán các anchor boxes này cho các lớp tương ứng.

Dưới đây là một minh họa: Anchor Box 1 thích hợp cho các hình dạng ngang như ngựa, trong khi Anchor Box 2 thích hợp cho các hình dạng dọc thẳng như con người.



Hình 20. Ví dụ Anchor Box 1 và 2

Anchor box đã làm tăng mAP (mean Average Precision) và cải thiện quá trình huấn luyện trong các phiên bản YOLO trước đó. Tuy nhiên, trong YOLOv8, kiến trúc đã quyết định bỏ qua việc sử dụng Anchor box với một số lý do:

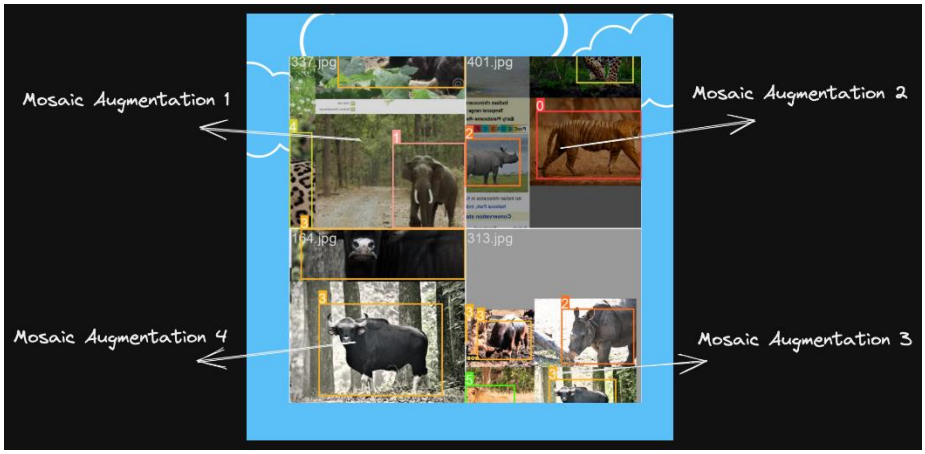
- Thiếu tính tổng quát: Huấn luyện với các Anchor box có sẵn khiến mô hình trở nên cứng nhắc và khó tự nhiên hóa với dữ liệu mới.

- Không có Anchor box phù hợp trong tình huống không đều: Các tình huống không đều không thể được ánh xạ một cách rõ ràng bằng các Anchor box hình đa giác.

- Mosaic Augmentation

Trong quá trình huấn luyện, YOLOv8 sử dụng nhiều kỹ thuật tăng cường cho dữ liệu huấn luyện, và một trong những kỹ thuật nổi bật là tăng cường dữ liệu mosaic.

Tăng cường dữ liệu mosaic là một phương pháp đơn giản, trong đó bốn hình ảnh khác nhau được kết hợp lại với nhau và được đưa vào mô hình như là dữ liệu đầu vào. Phương pháp này giúp mô hình học được các đối tượng từ các góc độ và vị trí đa dạng, cũng như trong các tình huống bị che khuất.



Hình 21. Mosaic Augementaion

Việc thực hiện tăng cường dữ liệu Mosaic đã cho thấy làm giảm hiệu suất, vì vậy nó đã được chuyển sang sử dụng cho 10 epoch cuối cùng.

3.4. Ứng dụng

3.4.1. Phân loại (Classification)

Phân loại là một nhiệm vụ đơn giản trong đó kết quả bao gồm chỉ số lớp và điểm tự tin. Nó hữu ích khi chỉ cần xác định sự hiện diện của một lớp cụ thể trong hình ảnh mà không cần xác định vị trí chính xác của đối tượng.

Việc tạo một bộ phân loại với YOLOv8 rất đơn giản - chỉ cần thêm tiền tố -cls vào mô hình YOLOv8 mong muốn của bạn. YOLOv8 có các biến thể với các kích thước khác nhau: yolov8n (Nano), yolov8s (Small), yolov8m (Medium), yolov8l (Large) và yolov8x (Extra Large). Sức mạnh của khả năng phân loại này xuất phát từ việc mô hình được huấn luyện trước trên tập dữ liệu ImageNet rộng lớn, bao gồm hàng triệu hình ảnh.

```
from ultralytics import YOLO

# Load the model
model = YOLO('yolov8n-cls.pt')

# Classification result
result = model('SOURCE_PATH')
```

Hình 22. Cài đặt YOLO (Classicfication)

3.4.2. Nhận dạng đối tượng (Object Detection)

Nhận dạng đối tượng là một bước tiến từ phân loại. Trong nhiệm vụ này, chúng ta cần xác định các lớp khác nhau xuất hiện trong hình ảnh và đồng thời phát hiện vị trí chính xác của chúng. Vị trí của các đối tượng này được biểu diễn trực quan qua các hộp giới hạn (Bounding Boxes).

Các mô hình YOLOv8 đã được huấn luyện trước trên tập dữ liệu COCO, một tập dữ liệu hình ảnh lớn. Vì vậy, chúng có thể thực hiện nhận dạng đối tượng ngay sau khi được triển khai mà không cần phải thêm bất kỳ tiền tố nào.

```
from ultralytics import YOLO

# Load the model
model = YOLO('yolov8n.pt')

# Object Detection result
result = model('SOURCE_PATH')
```

Hình 23. Cài đặt YOLO (Object Detection)

3.4.3. Phân đoạn (Segmentation)

Phân đoạn là bước tiếp theo sau nhận dạng đối tượng. Trong nhận dạng đối tượng, chúng ta đã xác định vị trí của các đối tượng và ước tính vị trí của chúng thông qua các hộp giới hạn. Phân đoạn là nhiệm vụ trong đó chúng ta xác định các pixel riêng lẻ thuộc về một đối tượng, mang lại độ chính xác cao hơn nhiều so với nhận dạng đối tượng. Phân đoạn có tiềm năng ứng dụng rộng rãi, từ hình ảnh y tế đến hình ảnh vệ tinh và nhiều lĩnh vực khác.

Thực hiện phân đoạn với YOLOv8 dễ dàng hơn bao giờ hết, chỉ cần thêm tiền tố -seg.

```
from ultralytics import YOLO

# Load the model
model = YOLO('yolov8m-seg.pt')

# Segmentation result
result = model('../Assets/Apples.jpg', save = True, project = "../Results/")
```

Hình 24. Cài đặt YOLO (Segmentation)

CHƯƠNG III. XÂY DỰNG ỨNG DỤNG

1. Ý tưởng

Chúng ta lựa chọn YOLOv8 từ nhóm phát triển Ultralytics làm nền tảng để xây dựng một ứng dụng đơn giản nhằm mục đích đếm lượng phương tiện di chuyển trên một đoạn đường cố định vào một thời điểm cụ thể. Ban đầu, chúng ta sẽ sử dụng các video sẵn có để phát triển ứng dụng. Tuy nhiên, từ những nền tảng này, chúng ta sẽ tiến xa hơn bằng việc kết nối với các camera thực tế trên các đoạn đường, từ đó tạo ra các giải pháp thông minh để giảm thiểu tình trạng ùn tắc giao thông và tối ưu hóa việc điều phối phương tiện giao thông.

Các bước cơ bản để thực hiện ý tưởng:

- Bước 1: Nhập video hoặc dữ liệu từ camera vào phần mềm và áp dụng các mặt nạ (mask) để lọc ra các vị trí quan trọng cần nhận dạng.
- Bước 2: Xây dựng hàm Detect_object, một hàm quan trọng để theo dõi các đối tượng trong một mảng dữ liệu, gọi là currentArray, chứa thông tin về các đối tượng cần nhận dạng. Hàm này sẽ tính toán độ tin cậy (confidence) dựa trên thông số của mỗi đối tượng trong mảng, từ đó xác định xem đối tượng đó là một phương tiện giao thông nào.
- Bước 3: Sử dụng hàm Draw_object để vẽ các khung xung quanh các đối tượng đã được nhận dạng. Tạo biến count và limits để đếm số lượng phương tiện đi qua một ngưỡng nhất định. Khi số lượng phương tiện vượt quá ngưỡng này, chúng sẽ được tính là một đối tượng và biến count sẽ tăng lên. Kết quả sẽ được hiển thị trực tiếp trên màn hình.

2. Môi trường phát triển và cài đặt

Ngoài các thư viện cơ bản đã được tích hợp trong Python, chúng ta cần cài đặt một số môi trường và thư viện bổ sung để phát triển ứng dụng. Đây là danh sách các thư viện cần thiết:

Tên thư viện	Phiên bản Tối thiểu	Lệnh Cài đặt
cvzone	1.5.6	pip install cvzone==1.5.6

ultralytics	8.0.26	pip install ultralytics==8.0.26
opencv-python	4.5.4.60	pip install opencv-python==4.5.4.60

Trong đó:

- ultralytics (Version 8.0.26): Được sử dụng cho mô hình YOLO để nhận diện và đếm các phương tiện giao thông.
- opencv-python (Version 4.5.4.60): Thư viện này cung cấp các công cụ và thuật toán xử lý ảnh và video.
- cvzone (Version 1.5.6): Thư viện này cung cấp các công cụ hỗ trợ cho việc xử lý ảnh và video.

3. Hướng dẫn sử dụng mã nguồn

Chúng ta thêm các thư viện vào trong dự án của chúng ta với các thư viện cần thiết sau khi cài đặt.

```
import numpy as np
from ultralytics import YOLO
import cv2
import cvzone
import math
from sort import *
import tkinter as tk
from tkinter import filedialog
```

Hình 25. Thư viện

Viết hàm đọc video và mask để đưa vào xử lí

```
1 usage
def select_video_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        entry_video_path.delete( first: 0, tk.END)
        entry_video_path.insert( index: 0, file_path)

1 usage
def select_mask_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        entry_mask_path.delete( first: 0, tk.END)
        entry_mask_path.insert( index: 0, file_path)
```

Hình 26. Hàm đọc file và mask

Viết hàm detect_object để xử lí các đối tượng được nhận dạng và tính toán độ tin cậy (conf) để nhận diện đối tượng, khi đối tượng được nhận dạng sẽ được đưa vào mảng currentArray.

```python
def detect_objects(img, model):
    results = model(img, stream=True)
    detections = np.empty((0, 5))

    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            w, h = x2 - x1, y2 - y1
            conf = math.ceil((box.conf[0] * 100)) / 100
            currentClass = classNames[int(box.cls[0])]

            if currentClass in ["car", "truck", "bus", "motorbike", "motorcycles"] and conf > 0.3:
                currentArray = np.array([x1, y1, x2, y2, conf])
                detections = np.vstack((detections, currentArray))

    return detections
```

Hình 27. Hàm detect_object

Tiếp theo là hàm draw_objects để vẽ khung các đối tượng được nhận dạng, khi các đối tượng được nhận dạng ở hàm trên kết hợp với hàm để thông qua tracker do thư viện Sort.py ta có thể lấy mã nguồn này từ tác giả Alex Bewley, chúng ta sẽ dùng hàm tracker để theo dõi đối tượng này đi qua giới hạn limits, nếu đối tượng này đi qua thì biến totalCount sẽ ghi nhận là 1 đối tượng.

```python
def draw_objects(img, detections, tracker, totalCount):
    resultsTracker = tracker.update(detections)
    cv2.line(img, pt1: (limits[0],limits[1]), pt2: (limits[2], limits[3]), color: (0, 0, 255), thickness: 5)

    for result in resultsTracker:
        x1, y1, x2, y2, id = map(int, result)
        w, h = x2 - x1, y2 - y1
        cvzone.cornerRect(img, bbox: (x1, y1, w, h), l=5, rt=2, colorR=(255,0,255))
        cvzone.putTextRect(img, text: f'{int(id)}', pos: (max(0, x1), max(35, y1)), scale=2, thickness=1, offset=3)

        cx, cy = x1 + w // 2, y1 + h // 2
        cv2.circle(img, center: (cx, cy), radius: 5, color: (255, 0, 255), cv2.FILLED)

        if limits[0] < cx < limits[2] and limits[1] - 20 < cy < limits[1] + 20:
            if totalCount.count(id) == 0:
                totalCount.append(id)
                cv2.line(img, pt1: (limits[0], limits[1]), pt2: (limits[2], limits[3]), color: (0, 255, 0), thickness: 5)

    cvzone.putTextRect(img, text: f"Count: {len(totalCount)}", pos: (50,50),colorR=(0, 0, 0))
    return img
```

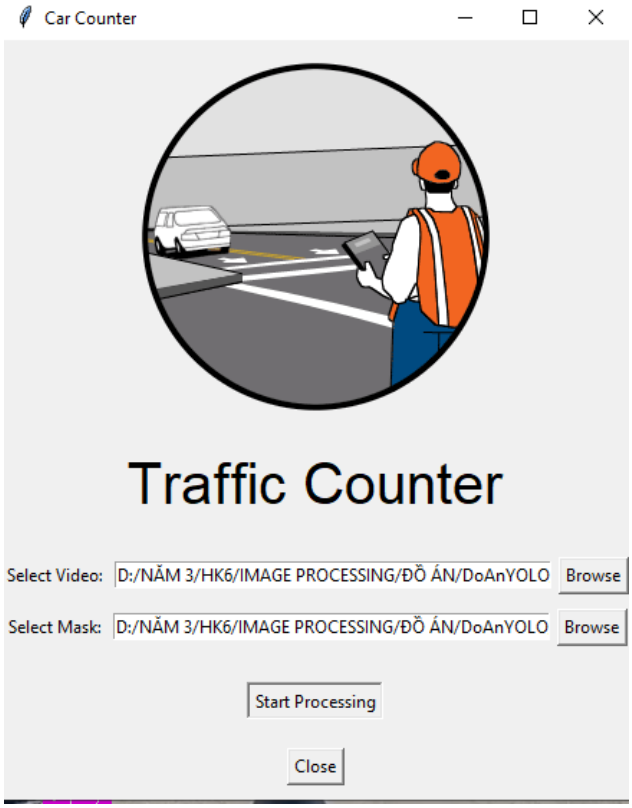
Hình 28. Hàm draw_object

Đây là các hàm chính của bài toán đếm số lượng phương tiên lưu thông qua cung đường cố định.

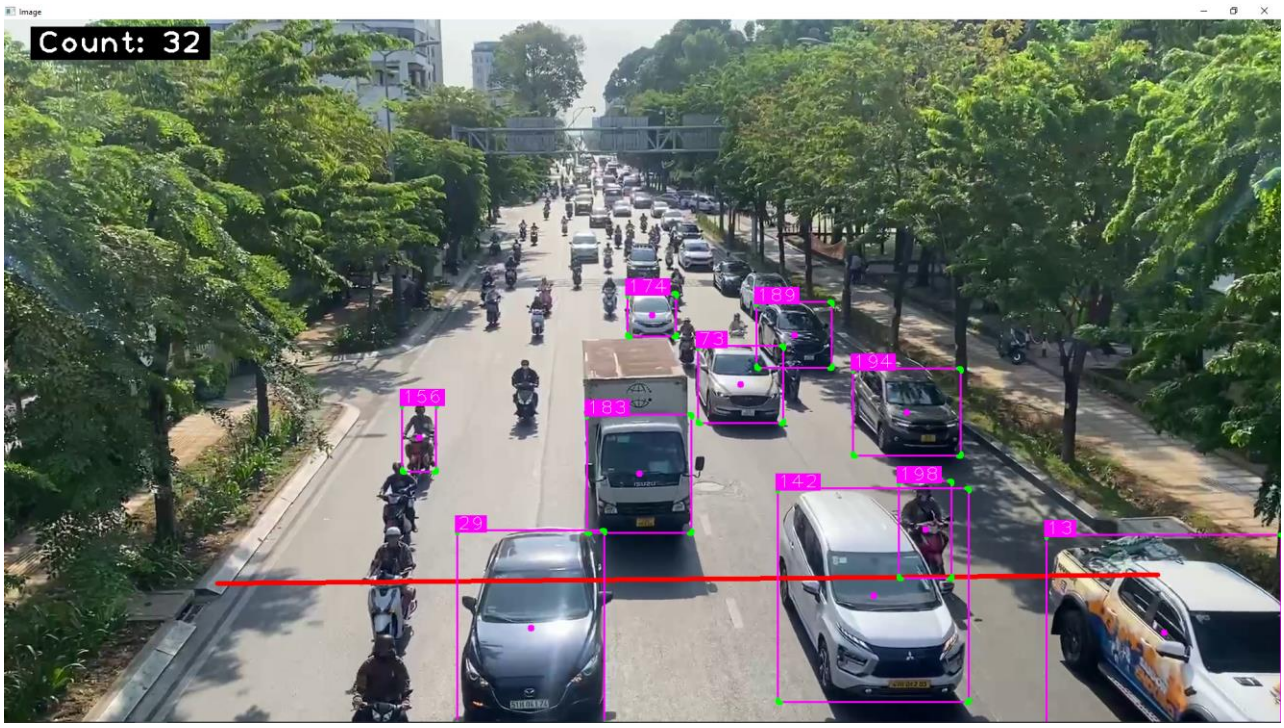
4. Mã nguồn mẫu

```
def select_video_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        entry_video_path.delete(0, tk.END)
        entry_video_path.insert(0, file_path)

def select_mask_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        entry_mask_path.delete(0, tk.END)
        entry_mask_path.insert(0, file_path)

def detect_objects(img, model):
    results = model(img, stream=True)
    detections = np.empty((0, 5))

    for r in results:
        boxes = r.boxes
        for box in boxes:
            x1, y1, x2, y2 = map(int, box.xyxy[0])
            w, h = x2 - x1, y2 - y1
            conf = math.ceil((box.conf[0] * 100)) / 100
            currentClass = classNames[int(box.cls[0])]

            if currentClass in ["car", "truck", "bus", "motorbike",
"motorcycles"] and conf > 0.3:
                currentArray = np.array([x1, y1, x2, y2, conf])
                detections = np.vstack((detections, currentArray))

    return detections

def draw_objects(img, detections, tracker, totalCount):
    resultsTracker = tracker.update(detections)
    cv2.line(img, (limits[0],limits[1]), (limits[2], limits[3]), (0, 0, 255), 5)

    for result in resultsTracker:
        x1, y1, x2, y2, id = map(int, result)
        w, h = x2 - x1, y2 - y1
        cvzone.cornerRect(img, (x1, y1, w, h), l=5, rt=2, colorR=(255,0,255))
        cvzone.putTextRect(img, f'{int(id)}', (max(0, x1), max(35, y1)),
scale=2, thickness=1, offset=3)

        cx, cy = x1 + w // 2, y1 + h // 2
        cv2.circle(img,(cx, cy), 5, (255, 0, 255), cv2.FILLED)

        if limits[0] < cx < limits[2] and limits[1] - 20 < cy < limits[1] + 20:
            if totalCount.count(id) == 0:
                totalCount.append(id)
                cv2.line(img, (limits[0], limits[1]), (limits[2], limits[3]),
(0, 255, 0), 5)

    cvzone.putTextRect(img,f"Count: {len(totalCount)}",(50,50),colorR=(0, 0, 0))
    return img
```

5. Giao diện và kết quả sau khi thực hiện thuật toán



Hình 29. Giao diện



Hình 30. Kết quả

CHƯƠNG IV. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Tóm tắt kết quả

Trong quá trình nghiên cứu và phát triển, em đã thành công trong việc xây dựng một ứng dụng sử dụng mô hình YOLOv8 để đếm lưu lượng phương tiện giao thông trên các cung đường. Ứng dụng của chúng tôi có khả năng nhận diện và đếm phương tiện với độ chính xác cao, đồng thời cung cấp một giao diện người dùng thân thiện.

2. Những hạn chế và hướng phát triển trong tương lai

Tuy kết quả đạt được là tích cực, nhưng còn một số hạn chế cần được giải quyết và các hướng phát triển tiềm năng cần được khai thác trong tương lai. Một số hạn chế bao gồm hạn chế về tốc độ xử lý, độ chính xác của việc đếm trong điều kiện ánh sáng yếu và khả năng ứng dụng trong các khu vực có mật độ phương tiện giao thông cao.

Để cải thiện hiệu suất và mở rộng khả năng ứng dụng, chúng tôi đề xuất một số hướng phát triển trong tương lai như:

- Tối ưu hóa thuật toán để cải thiện tốc độ xử lý và độ chính xác của việc đếm.
- Nghiên cứu và tích hợp các công nghệ cảm biến mới để cải thiện khả năng nhận diện trong các điều kiện ánh sáng yếu.
- Phát triển giao diện người dùng và tính năng bổ sung để tăng tính tương tác và sự thuận tiện cho người dùng cuối.

Chúng tôi tin rằng việc thực hiện những hướng phát triển này sẽ nâng cao sự hiệu quả và tính ứng dụng của ứng dụng đếm lưu lượng phương tiện giao thông của chúng tôi trong thực tế.

TÀI LIỆU THAM KHẢO

Geeksforgeeks. (n.d.). *YOLO : You Only Look Once – Real Time Object Detection*. Retrieved from Geeks for Geeks: https://www.geeksforgeeks.org/yolo-you-only-look-once-real-time-object-detection/

VINBIGDATA. (n.d.). *YOLO V7: Thuật toán phát hiện đối tượng có gì mới?* Retrieved from https://vinbigdata.com/kham-pha/yolo-v7-thuat-toan-phat-hien-doi-tuong-co-gi-moi.html#YOLO_la_gi

VINBIGDATA. (n.d.). *YOLOv8 có gì nâng cấp so với các phiên bản trước?* Retrieved from https://vinbigdata.com/cong-nghe-hinh-anh/yolov8-co-gi-nang-cap-so-voi-cac-phien-ban-truoc.html

Wikipedia. (n.d.). *YOLO*. Retrieved from https://vi.wikipedia.org/wiki/YOLO

Xia Zhao, Y. N. (n.d.). *Modified Object Detection Method Based on YOLO.* Sringer.

YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. (n.d.). In M. Hussain.