TRƯỜNG ĐẠI HỌC CÔNG NGHIỆP TP. HCM

**KHOA CÔNG NGHỆ THÔNG TIN**

--------------------------------



**BÁO CÁO BÀI TẬP LỚN**

**NHẬN DIỆN BIỂN BÁO BẰNG YOLO**

Các thành viên nhóm:

|  |  |
| --- | --- |
| Họ Tên | Mã số sinh viên |
| Chau Tiểu Long | 21094341 |
| Nguyễn Nhật Tùng | 21096911 |

Lớp: DHKHMT17B

GV hướng dẫn: Võ Quang Hoàng Khang

*Thành phố Hồ Chí Minh – Năm 2025*

Mục Lục

[1. Tổng quan về đồ án 1](#_Toc20594)

[1.1 Ngự cảnh ứng dụng 1](#_Toc30956)

[1.2. Input và ouput bài toán 2](#_Toc20640)

[2.Xây Dựng Bộ Dữ Liệu 3](#_Toc21450)

[2.1. Thu thập dữ liệu 3](#_Toc29292)

[2.1.1. Thông tin thu thập dữ liệu 3](#_Toc2389)

[2.1.2. Tiền xữ lí dữ liệu 4](#_Toc27299)

[2.1.3. Công cụ label dữ liệu 6](#_Toc6477)

[2.1.4. Kết quả thu thập dữ liệu 12](#_Toc1922)

[3. Huấn luyện và đánh giá model 16](#_Toc14452)

[3.1 Hướng tiếp cận và chọn model 16](#_Toc13401)

[3.2. Tổng quan YOLOv5, YOLOv8, YOLO11 17](#_Toc20522)

[3.2.1 YOLOv5 17](#_Toc23339)

[3.2.2 YOLOv8 18](#_Toc14330)

[3.2.3 YOLO11 19](#_Toc7303)

[3.3 Các bước tiến hành huấn luyên 20](#_Toc24741)

[3.3.1 Nền tảng 20](#_Toc22250)

[3.3.2 YOLOv5 23](#_Toc9324)

[3.3.3 YOLOv8 25](#_Toc615)

[3.3.4 YOLO11 26](#_Toc20126)

[3.3.5 Cấu Trúc Dự Án Huấn Luyện YOLO 27](#_Toc19058)

[3.4. Phương thức đánh giá mô hình 29](#_Toc15550)

[3.4.1 Độ đo mAP-50 30](#_Toc24305)

[3.5 Đánh giá các mô hình 33](#_Toc19905)

[4. Ứng dụng 42](#_Toc32525)

[4.1 Ý tưởng 42](#_Toc14372)

[4.2 Cách thức thực hiện. 42](#_Toc4953)

[4.2.1 Chuyển đổi model 42](#_Toc26108)

[4.2.2 Xây dựng app android bằng phần mềm android studio 43](#_Toc6718)

[4.2.3 Kết quả thực nghiệm 44](#_Toc11813)

[4.3. Đánh giá và tối ưu 46](#_Toc217)

[Tài liệu Tham khảo 47](#_Toc15414)

# 1. Tổng quan về đồ án

## 1.1 Ngự cảnh ứng dụng

Tại Việt Nam, giao thông đường bộ giữ vai trò chủ đạo và phát triển mạnh mẽ nhất trong các loại hình giao thông. Theo thống kê của Cục Cảnh sát giao thông, tính đến cuối năm 2019, cả nước có khoảng 62 triệu xe máy được đăng ký (Nguồn: [Cục Cảnh sát giao thông – Bộ Công an, 2019](http://www.csgt.vn)). Sự gia tăng nhanh chóng của phương tiện kéo theo nhu cầu mở rộng và hiện đại hóa hệ thống hạ tầng giao thông. Tuy nhiên, song song với sự phát triển đó là những vấn đề tồn tại, đặc biệt là tình trạng vi phạm luật giao thông.

Trong thực tế, người điều khiển phương tiện đôi khi quá tập trung vào việc lái xe mà không để ý đến các biển báo giao thông trên đường. Việc không chú ý này có thể khiến họ vô tình vi phạm quy định hoặc tệ hơn là gây ra những tai nạn đáng tiếc.

Để giải quyết tình trạng này, nhóm chúng em đề xuất xây dựng một mô hình máy học có khả năng nhận diện biển báo giao thông theo thời gian thực, hỗ trợ người tham gia giao thông nâng cao ý thức và khả năng quan sát khi lưu thông trên đường.

Thành phố Hồ Chí Minh được chọn làm khu vực thu thập dữ liệu và tiến hành đánh giá mô hình trong thực nghiệm.

Đối tượng sử dụng là người điều khiển xe máy, có trang bị:

Điện thoại di động, ảnh chụp từ điện thoại có kích thước 1920x1080.

Giá đỡ điện thoại gắn trên xe máy (như hình 1).



*hình 1. thiết bị gắn trên xe*

## 1.2. Input và ouput bài toán

Input: Video được quay từ camera điện thoại gắn trên giá đỡ.

Output: Video với các thông tin:

Bbox bao quanh các biển báo của biển báo.

Tên biển báo

# 2.Xây Dựng Bộ Dữ Liệu

## 2.1. Thu thập dữ liệu

### 2.1.1. Thông tin thu thập dữ liệu

Dữ liệu được thu thập bằng cách sử dụng điện thoại di động(FPS=30) để quay video trong quá trình di chuyển bằng xe máy hoặc ô tô. Tốc độ di chuyển duy trì ổn định ở mức 15–20 km/h, đảm bảo an toàn và tuân thủ quy định giao thông tại khu vực cho phép. Thiết bị quay được gắn cố định để tránh rung lắc và đảm bảo góc nhìn ổn định phía trước.

Sau khi quá trình quay video hoàn tất, các khung hình (frame) được trích xuất từ video để làm dữ liệu huấn luyện. Việc cắt ảnh từ video giúp đảm bảo các hình ảnh có độ liên tục, phản ánh điều kiện thực tế như ánh sáng, vật cản, và các loại biển báo giao thông trong nhiều tình huống khác nhau (ban ngày, chiều tối).

Quy trình trích xuất ảnh từ video được thực hiện theo tỷ lệ cố định (ví dụ: trích xuất 1 ảnh mỗi 30 frame, tương ứng với 1 ảnh mỗi giâ**y**.), nhằm tránh thu thập quá nhiều hình ảnh giống nhau và tăng tính đa dạng của tập dữ liệu.

Các lần thu thập dữ liệu:

Bảng 2.1.1. Thông tin các đợt thu thập dữ liệu

|  |  |  |  |
| --- | --- | --- | --- |
| Lần thu thập | Ngày thu thập | Số lượng ảnh sau khi lọc | Mục đích |
| 1 | 19/1/2025 | 1522 | Train & val |
| 2 | 23/2/2025 | 2929 | Train & val |
| 3 | 9/3/2025 | 1154 | Train & val |
| 4 | 20/4/2025 | 2.538 | Train & val |
| 5 | 25/4/2025 | Không cắt | Test |

### 2.1.2. Tiền xữ lí dữ liệu

Xóa những ảnh bị mờ và bị khuất:  
Các ảnh bị mất nét (blurred) hoặc bị vật thể khác che khuất đối tượng chính cần được loại bỏ khỏi tập dữ liệu. Những ảnh này không chỉ làm giảm chất lượng huấn luyện mà còn gây nhiễu cho mô hình khi nhận diện.  
Ngoài ra, có thể sử dụng một số kỹ thuật để phát hiện ảnh mờ tự động như tính Laplacian Variance – giá trị phương sai càng thấp thì ảnh càng mờ.

Tăng độ sáng cho ảnh tối:  
Nếu một số ảnh hơi tối nhưng vẫn nhìn rõ đối tượng, có thể cải thiện bằng các kỹ thuật:

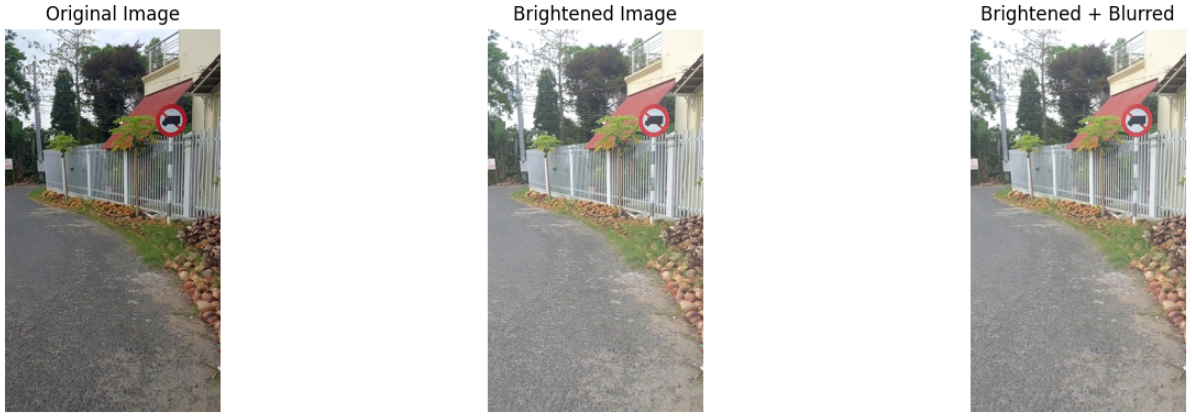
Gamma Correction: Điều chỉnh độ sáng phi tuyến tính.

*Hình 2. Điều chỉnh độ sáng*

Histogram Equalization: Tăng độ tương phản cho ảnh.

*Hình 3. Tăng độ tương phản*

Sử dụng kernel làm mờ nhẹ kết hợp với tăng sáng để giảm nhiễu khi làm sáng ảnh.



*Hình 4. Tăng sáng và giảm nhiễu*

Chuyển đổi định dạng ảnh:

Đảm bảo ảnh có định dạng thống nhất (JPG/PNG), không bị lỗi, và có thể đọc được bằng thư viện xử lý ảnh.

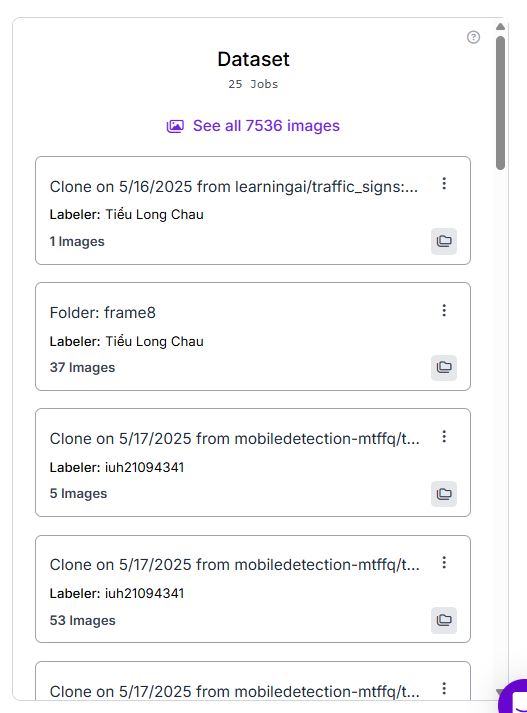
### 2.1.3. Công cụ label dữ liệu

Nhóm em chọn web Roboflow để lable dữ liệu.

Lý do chọn:

Web có giao diện đẹp, thân thiện cho người sử dụng.

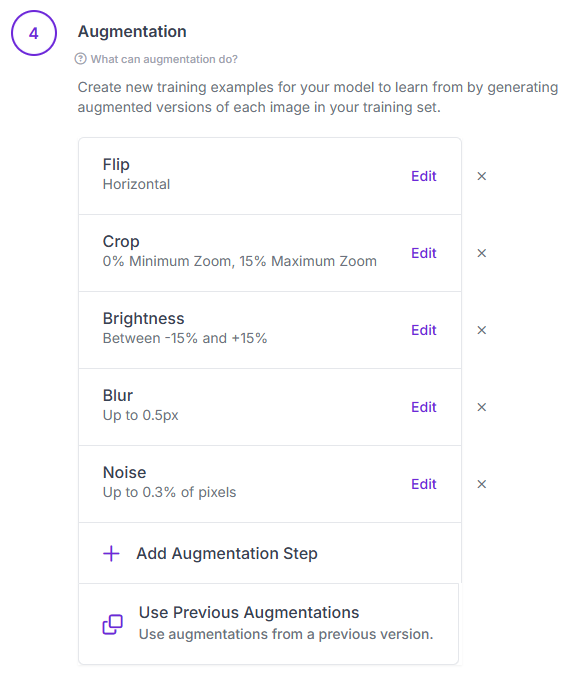
Có hỗ trợ chia công việc cho thành viên trong nhóm, dễ dàng quản lý tiến trình hoàn thành dự án.



*Hình 5. bộ datase*

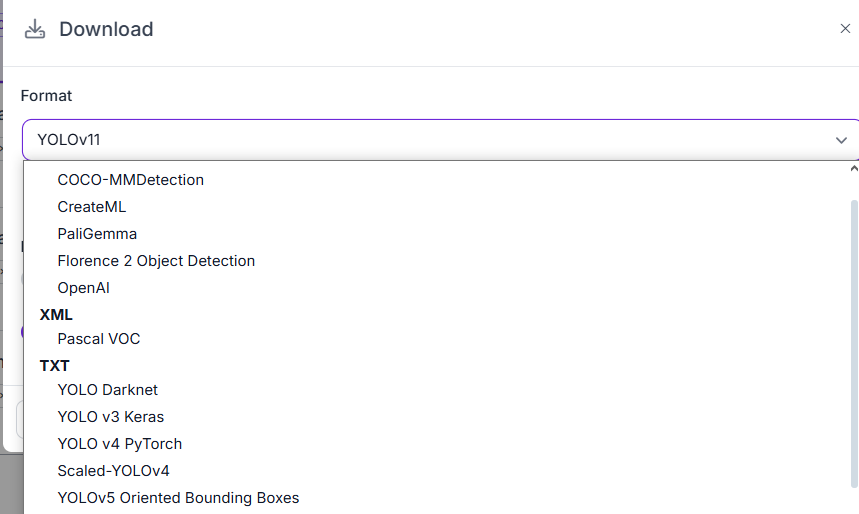
Thao tác dễ dàng với các phím tắt, giúp công việc trở nên trơn tru hơn.

Khi xuất ra data, robolow sẽ có thêm tiến trình làm giàu dữ liệu.



*Hình 6. Chỉnh tham số xuất ảnh*

Khi lable xong, web sẽ đưa ra data có định dạng phù hợp với model YoLo, rất thuận tiện khi đưa vào quá trình train mà không cần thao tác gì thêm.

 *Hình 7. Dowload dữ liệu*

Quy tắt gán nhãn dữ liệu:

Khi gán nhãn, cần bao bouding box vừa với đối tượng, không bị lệch.

Cần xác định chính xác nhãn cho đối tượng, tránh trường hợp gán nhãn sai để tránh việc dữ liệu train bị nhiễu dẫn đến model học sai.

*Hình 7. Đánh label*

Sau khi gán nhãn xong, web sẽ xuất ra folder chứa ảnh, foler chứa file txt.

Trong đó file txt có cấu trúc

|  |
| --- |
| <ID> <X\_center> <Y\_center> <Width> <Height> |

VD:



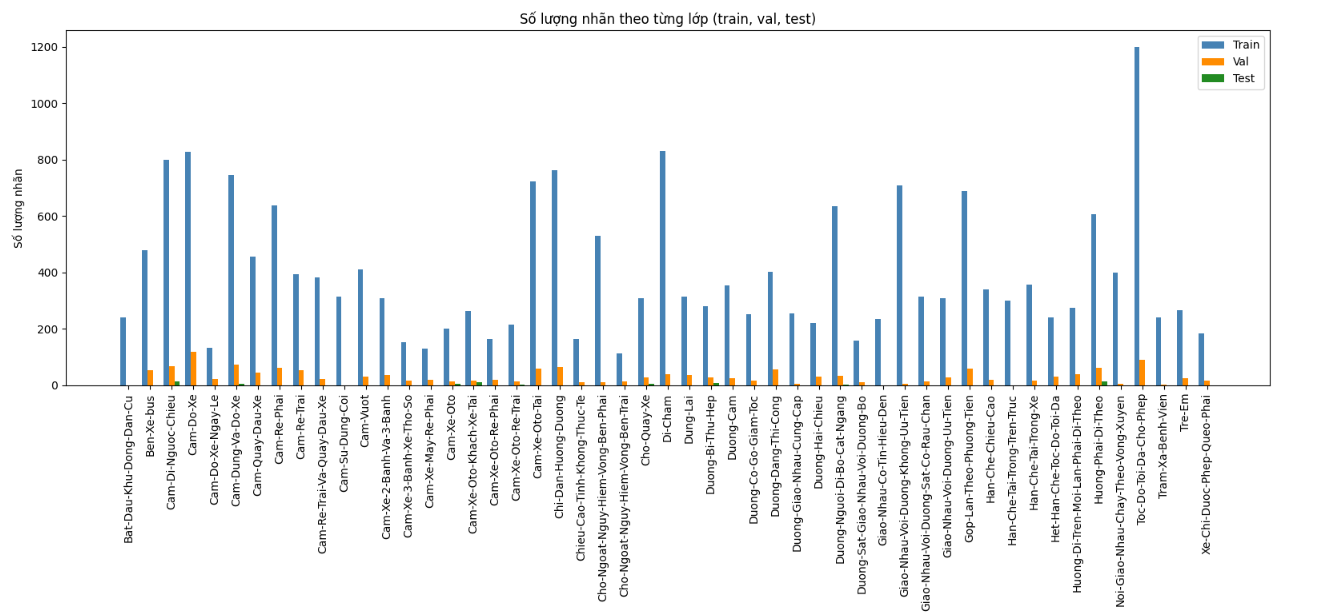
* + ID là thứ tự của nhãn đối tượng cần phân lớp.
  + X\_center: hoành độ của tâm đối tượng.
  + Y\_center: tung độ của đối tượng.
  + Width: Chiều rộng của bouding box.
  + Height: Chiều cao của bouding box.
  + Công thức tính X\_center, Y\_center.



### 2.1.4. Kết quả thu thập dữ liệu

Sau nhiều lần thu thập data đầu tiên, có tất cả 8143 ảnh, thuộc vào 61 class. Nhưng do có những class số lượng ảnh dưới 100, những ảnh label không đúng quy tắc ban đầu đề ra nên chúng em tiến hành loại bỏ. Sau cùng thu được 7536 ảnh thuộc 51 class bên dưới:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Tên biển báo | Hình ảnh | ID | Tên biển báo | Hình ảnh |
| 1 | Bắt đàu khu đông dân cư |  | 27 | Dừng lại |  |
| 2 | Bến xe bus |  | 28 | Đường bị thu hẹp |  |
| 3 | Cấm đi ngược chiêu |  | 29 | Đường cấm |  |
| 4 | Cấm đỗ xe |  | 30 | Đường có gồ giảm tốc |  |
| 5 | Cấm đỗ xe ngày lẻ |  | 31 | Đường đang thi công |  |
| 6 | Cấm dừng và đỗ xe |  | 32 | Đường giao nhau cùng cấp |  |
| 7 | Cấm quay đầu xe |  | 33 | Đường hai chiều |  |
| 8 | Cấm rẻ phải |  | 34 | Đường người đi bộ cắt ngang |  |
| 9 | Cấm rẻ trái |  | 35 | Nơi đường sắt giao vuông góc với đường bộ |  |
| 10 | Cấm rẻ trái và quay đầu xe |  | 36 | Giao nhau có tín hiệu đèn |  |
| 11 | Cấm sử dụng còi |  | 37 | Giao nhau với đường không ưu tiên |  |
| 12 | Cấm vượt |  | 38 | Giao nhau với đường săt có ràu chắn |  |
| 13 | Cấm xe 2 bánh và 3 bánh |  | 39 | Giao nhau với đường ưu tiên |  |
| 14 | Cấm xe 3 bánh và xe thô sơ |  | 40 | Gộp làn theo phương tiện |  |
| 15 | Cấm xe máy rẻ phải |  | 41 | Hạn chế chiều cao |  |
| 16 | Cấm xe oto |  | 42 | Hạn chế tải trọng trên trục |  |
| 17 | Cấm xe Oto khách và xe tải |  | 43 | Hạn chế tải trọng xe |  |
| 18 | Cấm xe oto rẻ phải |  | 44 | Hết hạn chế tốc độ tối đa |  |
| 19 | Cấm xe oto rẻ trái |  | 45 | Hướng đi trên mỗi làn phải đi theo |  |
| 20 | Cấm xe oto tải |  | 46 | Hướng phải đi theo |  |
| 21 | Chỉ dấn hướng đường |  | 47 | Nơi giao nhau chạy theo vòng xuyên |  |
| 22 | Chiều cao tĩnh không thực tế |  | 48 | Tốc độ tối da cho phép |  |
| 23 | Chỗ ngoặt nguy hiểm vòng bên phải |  | 49 | Trạm xá/ bệnh viên |  |
| 24 | Chỗ ngoặt nguy hiểm vòng bên trái |  | 50 | Trẻ em |  |
| 25 | Chỗ quay xe |  | 51 | Xe chỉ được phép quẹo phải |  |
| 26 | Đi chậm |  |  |  |  |

*Hình 8. Tổng số lượng nhãn lớp*

Dựa trên biểu đồ thống kê số lượng nhãn theo từng lớp trong ba tập dữ liệu (train, val, test), có thể nhận thấy rằng bộ dữ liệu được phân bố khá đều giữa các lớp. Cụ thể, tất cả các lớp trong tập huấn luyện (train) đều có số lượng ảnh lớn hơn 100, đảm bảo đủ dữ liệu để mô hình học được đặc trưng của từng lớp một cách hiệu quả. Mặc dù số lượng nhãn trong các tập kiểm tra (test) và tập xác thực (val) thấp hơn so với tập huấn luyện, nhưng vẫn phân bố rộng khắp trên hầu hết các lớp, đảm bảo tính đại diện cho việc đánh giá và hiệu chỉnh mô hình. Điều này cho thấy bộ dữ liệu có chất lượng tốt và phân bố hợp lý, là nền tảng thuận lợi cho quá trình huấn luyện và đánh giá mô hình nhận dạng biển báo giao thông.

# 3. Huấn luyện và đánh giá model

## 3.1 Hướng tiếp cận và chọn model

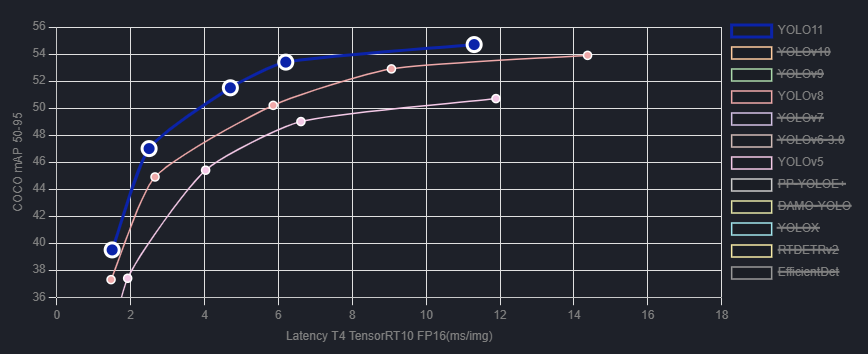
Các thuật toán phát hiện đối tượng hiện đại có thể chia thành hai nhóm chính:

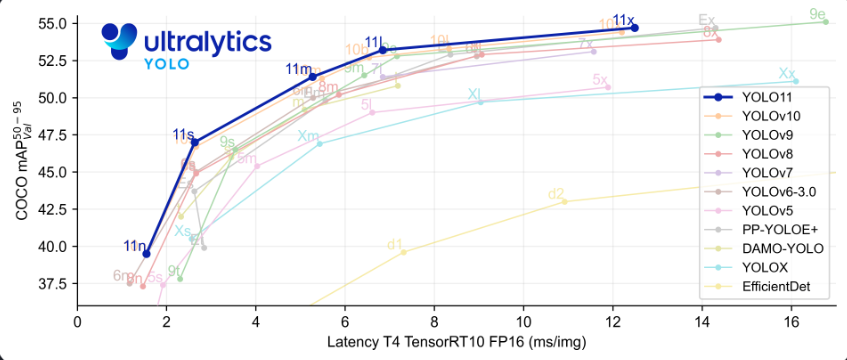
Nhóm dựa trên Region Proposal như R-CNN, Fast R-CNN, và Faster R-CNN  
Những mô hình này có độ chính xác cao nhờ cơ chế tách riêng bước đề xuất vùng (region proposal) và phân loại. Ví dụ, theo bài báo Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identication, Faster R-CNN đạt mAP 87.96%, nhưng tốc độ xử lý FPS thấp do kiến trúc nhiều giai đoạn (multi-stage).

Nhóm mô hình một bước (single-stage) như YOLO (You Only Look Once): Mặc dù độ chính xác thấp hơn một chút (YOLOv3 đạt mAP 80.17%), nhưng lại có tốc độ xử lý rất cao – gấp 8 lần so với Faster R-CNN – điều này rất quan trọng đối với các ứng dụng thời gian thực.

Dựa vào đặc điểm của bài toán chúng em đặt ra, yếu tố tốc độ nhận diện có vai trò quan trọng hơn so với độ chính xác. Do đó chúng em quyết định dùng YOLO để thực hiện bài toán này. Hai phiên bản chúng em chọn là YOLOv5, YOLOv8, YOLO11.

## 3.2. Tổng quan YOLOv5, YOLOv8, YOLO11





### 3.2.1 YOLOv5

Tác giả chính: Glenn Jocher.

Ngày phát hành: 2020-06-26

YOLOv5 là một sản phẩm nổi tiếng dựa vào tốc độ xử lý và độ chính xác khi mới vừa ra mắt. Mô hình được nhiều nhà nghiên cứu tin tưởng và áp dụng vào nhiều lĩnh vực trong cuộc sống

YOLOv5 được xây dựng với kiến trúc chính: xương sống CSPDarknet53 để trích xuất đặc trưng, mạng tổng hợp PANet giúp kết hợp thông tin ở nhiều cấp độ, và phần đầu phát hiện YOLOv5 để dự đoán các hộp giới hạn. Mô hình áp dụng phương pháp dựa trên mỏ neo (anchor-based) để phát hiện đối tượng. Nhờ thiết kế tối ưu, YOLOv5 cho phép huấn luyện và suy luận nhanh chóng, đồng thời có sẵn nhiều phiên bản kích thước khác nhau như n, s, m, l, x — giúp linh hoạt điều chỉnh theo yêu cầu về tài nguyên và hiệu suất. Ngoài ra, Ultralytics cung cấp tài liệu chi tiết và giao diện thân thiện, hỗ trợ người dùng dễ dàng trong quá trình huấn luyện và triển khai mô hình.

### 3.2.2 YOLOv8

Tác giả: Glenn Jocher, Ayush Chaurasia và Jing Qiu

Ngày: 2023-01-10

Ultralytics YOLOv8, ra mắt vào tháng 1 năm 2023, là phiên bản mới nhất trong dòng mô hình YOLO của Ultralytics tại thời điểm công bố. Đây là một framework đa năng, hỗ trợ đầy đủ các nhiệm vụ về thị giác máy tính như phát hiện đối tượng, phân đoạn, phân loại, ước lượng tư thế và hộp giới hạn định hướng (OBB).

Về kiến trúc và tính năng nổi bật: YOLOv8 sử dụng một mạng backbone hoàn toàn mới, kết hợp với đầu phát hiện không cần anchor hiện đại và hàm mất mát được cải tiến. Kiến trúc mô-đun linh hoạt giúp người dùng dễ dàng tùy chỉnh và mở rộng theo nhu cầu. Thiết kế này giúp nâng cao cả độ chính xác lẫn hiệu suất xử lý trong nhiều bài toán thị giác khác nhau. Ngoài ra, Ultralytics cũng tập trung vào trải nghiệm người dùng khi cung cấp tài liệu chi tiết, API thân thiện và tích hợp sẵn với Ultralytics HUB, giúp quy trình làm việc trở nên dễ dàng và hiệu quả hơn

### 3.2.3 YOLO11

Tác giả: Glenn Jocher, Ayush Chaurasia và Jing Qiu

Ngày: 2024-09-27

Ultralytics YOLO11. Trích xuất đặc trưng mạnh mẽ: YOLO11 sử dụng kiến trúc backbone và neck được nâng cấp, giúp tăng khả năng trích xuất đặc trưng, từ đó cải thiện độ chính xác trong việc phát hiện đối tượng và xử lý các tác vụ thị giác phức tạp.

Tối ưu hóa về tốc độ và hiệu suất: Với kiến trúc được thiết kế tinh gọn và quy trình huấn luyện được tối ưu, YOLO11 mang lại tốc độ xử lý nhanh chóng đồng thời duy trì sự cân bằng tốt giữa độ chính xác và hiệu quả tính toán.

Hiệu suất cao với ít tham số hơn: Nhờ cải tiến trong thiết kế, mô hình YOLO11m đạt được độ chính xác trung bình (mAP) cao hơn trên bộ dữ liệu COCO, đồng thời sử dụng ít hơn 22% tham số so với YOLOv8m — giúp tiết kiệm tài nguyên mà không làm giảm chất lượng dự đoán.

Thích nghi linh hoạt với nhiều nền tảng: YOLO11 có thể triển khai mượt mà trên nhiều môi trường khác nhau, từ thiết bị biên (edge devices) cho đến nền tảng đám mây và hệ thống sử dụng GPU NVIDIA, đảm bảo khả năng ứng dụng đa dạng.

Hỗ trợ nhiều tác vụ thị giác: Không chỉ dừng lại ở phát hiện đối tượng, YOLO11 còn hỗ trợ các tác vụ khác như phân đoạn instance, phân loại hình ảnh, ước lượng tư thế và nhận diện đối tượng theo hướng (OBB), đáp ứng nhu cầu đa dạng trong các ứng dụng AI thị giác.

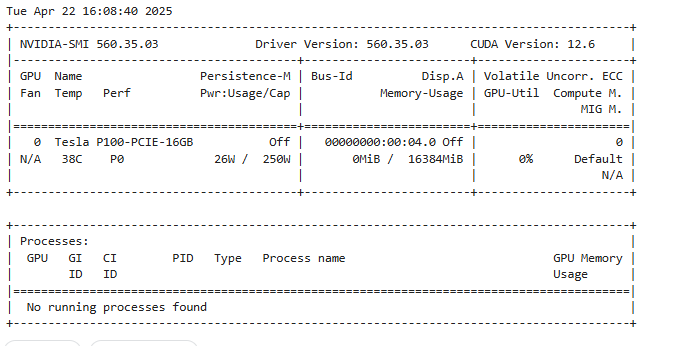
## 3.3 Các bước tiến hành huấn luyên

### 3.3.1 Nền tảng

Nhóm em chọn 2 nền tảng web Kaggle và máy tính cá nhân có tích hợp CUDA sử dụng GPU để train mô hình

#### 3.3.1.1 Train trên Kaggle

Kaggle là một nền tảng trực tuyến do Google sở hữu, cung cấp một môi trường lập trình dựa trên đám mây hoàn toàn miễn phí, đặc biệt dành cho các nhà khoa học dữ liệu và nhà nghiên cứu. Kaggle Notebooks cho phép người dùng huấn luyện và đánh giá các mô hình học máy trực tiếp trên GPU hoặc TPU miễn phí. Một điểm nổi bật của Kaggle là môi trường đã được cấu hình sẵn với hàng trăm thư viện phổ biến trong lĩnh vực machine learning và deep learning, giúp người dùng tiết kiệm thời gian cài đặt và dễ dàng bắt đầu dự án.

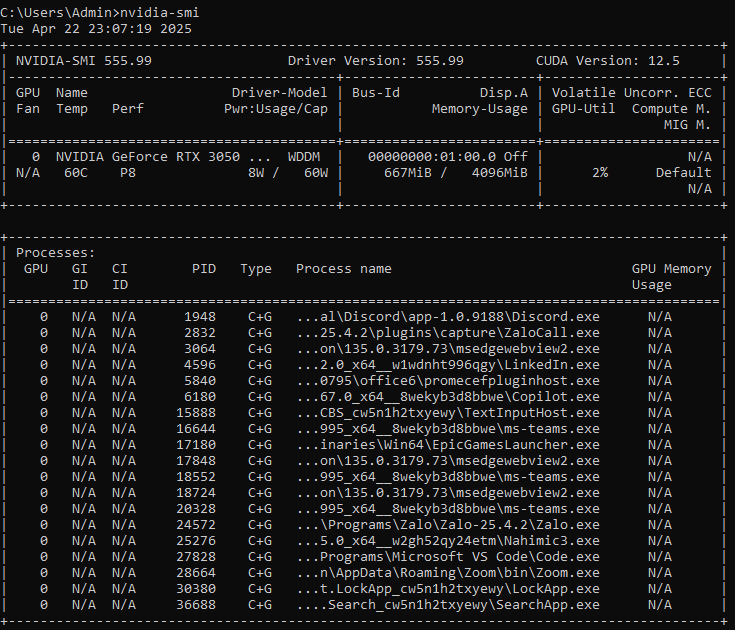


*Hình 8. GPU trên kaggle*

Do Kaggle hỗ trợ trực tiếp việc tải và sử dụng dataset thông qua thư viện kaggle hoặc từ kho dữ liệu của nền tảng, quá trình chuẩn bị dữ liệu diễn ra nhanh chóng và thuận tiện (15 phút). Bên cạnh đó, dữ liệu khi được tải lên sẽ được lưu giữ trong suốt phiên làm việc mà không bị mất đi như trên Colab. Vì vậy, chúng em lựa chọn lưu trữ và xử lý dữ liệu trực tiếp trên Kaggle để đảm bảo tính ổn định và tiết kiệm thời gian khi huấn luyện mô hình.

#### 3.3.1.2 Train trên máy cá nhân

Thông số chính:



*Hình 9. GPU trên máy tính*

Phần cứng: card đồ họa RTX3050 - 4G

Phiên bản cuda: 12.4

Phiên bản drive Nvidia: 555.99

Phiên bản Ultralytics: 8.3.94

### 3.3.2 YOLOv5

Để bắt đầu quá trình huấn luyện, chúng em sử dụng **pretrained weights** YOLOv5n.pt – đây là trọng số đã được huấn luyện trước trên tập dữ liệu COCO, chứa hơn 80 lớp đối tượng phổ biến. Việc sử dụng pretrained weights giúp rút ngắn thời gian huấn luyện so với việc train mô hình từ đầu, đồng thời tận dụng được các đặc trưng cơ bản đã học trước đó như biên cạnh, hình dạng, hoa văn, v.v.

Khi sử dụng pretrained weights, lớp đầu ra (dùng để phân loại) sẽ được tự động điều chỉnh lại để phù hợp với số lượng lớp của bài toán hiện tại. Điều này giúp mô hình nhanh chóng học các đặc trưng cụ thể hơn của tập dữ liệu tùy chỉnh mà không cần khởi tạo lại toàn bộ từ đầu.

Các bước thực hiện huấn luyện:

1. **Clone source code YOLOv5:** Chúng em clone repo chính thức của Ultralytics từ GitHub (https://github.com/ultralytics/YOLOv5) để sử dụng các tiện ích huấn luyện, validate và inference sẵn có.
2. **Chuẩn bị dữ liệu:**

Thư mục images/train và images/val chứa ảnh dùng cho huấn luyện và kiểm tra.

Thư mục labels/train và labels/val chứa các file .txt mô tả bounding boxes cho từng ảnh, theo định dạng YOLO.

File data.yaml được cấu hình để chỉ ra:

* + - Đường dẫn đến tập train/val
    - Số lớp (nc)
    - Tên các lớp (names)

1. **Cấu hình tham số huấn luyện:**
   * Sử dụng model: YOLOv5s.pt (phiên bản nhẹ, phù hợp với tài nguyên GPU hiện có).
   * Epochs = 30
   * Batch size = 4
   * Image size = 640
   * Device = 'cuda' (GPU)
2. **Lệnh train:**

|  |
| --- |
| python train.py --img 640 --batch 8--epochs 30--data data.yaml --weights YOLOv5m.pt --name YOLOv5-custom |

1. **Theo dõi quá trình training:**

Loss, mAP, Precision, Recall được theo dõi tự động qua TensorBoard hoặc các biểu đồ do Ultralytics sinh ra.

Trọng số mô hình sau mỗi epoch được lưu tại thư mục runs/train/YOLOv5-custom/weights/.

### 3.3.3 YOLOv8

Bước chuẩn bị dữ liệu train tương tự như YOLOv5 So với YOLOv8, quá trình huấn luyện YOLOv8n có một số điểm khác biệt quan trọng. Đầu tiên, YOLOv8 sử dụng cú pháp dòng lệnh mới do chính Ultralytics phát triển, đơn giản và dễ sử dụng hơn. Nếu như YOLOv5 cần chạy các file riêng như train.py, val.py, thì với YOLOv8, tất cả đều được tích hợp trong một lệnh duy nhất thông qua CLI yolo.

Ngoài ra, YOLOv8 được cài đặt thông qua gói ultralytics bằng lệnh pip install ultralytics, thay vì phải clone thủ công repo như YOLOv5. Các output khi huấn luyện YOLOv8 cũng được lưu tại thư mục runs/detect/... thay vì runs/train/... như trước.

Về mặt kiến trúc, YOLOv8 áp dụng phương pháp anchor-free thay cho anchor-based như ở YOLOv5, giúp đơn giản hóa mô hình và tăng hiệu suất xử lý. Không chỉ dừng lại ở phát hiện đối tượng, YOLOv8 còn hỗ trợ nhiều tác vụ khác như phân đoạn ảnh, phân loại, ước lượng tư thế và hộp giới hạn định hướng (OBB).

Lệnh train:

|  |
| --- |
| from ultralytics import YOLO  model = YOLO("YOLOv8m.yaml") # hoặc YOLOv8n.pt để fine-tune từ pretrained  model.train(data="/kaggle/input/YOLOv8/Datav8/data.yaml", epochs=30, imgsz=640, batch=8) |

### 3.3.4 YOLO11

Cũng giống như YOLOv5 và YOLOv8 Bước chuẩn bị dữ liệu train tương tự. Nhưng so với YOLOv5 và YOLOv8, YOLO11 có những cải tiến rõ rệt về tối ưu hóa số lượng tham số, giúp tăng độ chính xác mà không làm giảm hiệu suất. Khác với YOLOv5, YOLO11 sử dụng một kiến trúc xương sống mới, tối ưu hóa các lớp mạng để cải thiện khả năng tính toán và phát hiện đối tượng chính xác hơn, đặc biệt trong các tác vụ phức tạp.

Bên cạnh đó, YOLO11 được thiết kế để dễ dàng triển khai trên nhiều môi trường khác nhau, bao gồm các hệ thống biên, đám mây và các nền tảng hỗ trợ GPU. Với sự tối ưu hóa này, YOLO11 có thể đạt được hiệu suất vượt trội, đặc biệt trong các tình huống yêu cầu khả năng phát hiện đối tượng chi tiết và chính xác hơn.

Lệnh train:

|  |
| --- |
| from ultralytics import YOLO  model = YOLO("YOLO11m.yaml")  model.train(data="/kaggle/input/yolov1/Datav1/data.yaml", epochs=30, imgsz=640, batch=8) |

### 3.3.5 Cấu Trúc Dự Án Huấn Luyện YOLO

Khi chuẩn bị để huấn luyện một mô hình YOLO (bao gồm YOLO11, YOLOv8, YOLOv5), bạn cần tổ chức các tệp và thư mục như sau:

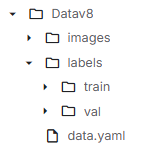
|  |
| --- |
| /project\_folder/  ├── /images/ # Chứa ảnh huấn luyện  │ ├── train/ # Ảnh huấn luyện  │ ├── val/ # Ảnh kiểm tra  │ └── test/ # Ảnh kiểm tra (tùy chọn)  ├── /labels/ # Chứa nhãn (labels) cho các ảnh  │ ├── train/ # Nhãn cho ảnh huấn luyện  │ ├── val/ # Nhãn cho ảnh kiểm tra  │ └── test/ # Nhãn cho ảnh kiểm tra (tùy chọn)  ├── data.yaml # File cấu hình cho tập dữ liệu  ├── YOLO11n.yaml # Cấu hình mô hình YOLO11  ├── YOLO11n.pt # Trọng số đã huấn luyện hoặc pre-trained weight  ├── train.py # Script huấn luyện mô hình  └── val.py # Script kiểm tra mô hình |

File cấu hình dữ liệu (data.yaml):

Đây là tệp quan trọng giúp YOLO hiểu cách thức tổ chức dữ liệu. Nó chỉ ra các lớp cần phân loại, cũng như các đường dẫn đến ảnh huấn luyện, kiểm tra và nhãn.

|  |
| --- |
| train: /project\_folder/images/train # Đường dẫn đến ảnh huấn luyện  val: /project\_folder/images/val # Đường dẫn đến ảnh kiểm tra  test: /project\_folder/images/test # Đường dẫn đến ảnh kiểm tra (tùy chọn)  nc: 5 # Số lượng lớp (classes)  names: ['class1', 'class2', 'class3', 'class4', 'class5'] # Tên các lớp (classes) |

Một số hình ảnh về đặt file bài làm chúng em:



*Hình 10. Bố cục đặt file*



*Hình 11. Thông in file yaml*

## 3.4. Phương thức đánh giá mô hình

Sau khi thực hiện các bước train, để có thể nhìn nhận một cách khách quan về sự tốt xấu, mạnh yếu của từng mô hình, cần phải có những tiêu chí đánh giá và độ đo cụ thể để làm rõ điều đó.

Đối với mô hình YoLo và bài toán phát hiện đối tượng (Oject Detection), có nhiều phương pháp và độ đo để đánh giá mô hình. Trong bài toán này, nhóm em quyết định sử dụng độ đo mAP50 để đánh giá mức độ chính xác của bouding box và đánh giá độ chính xác trong việc phân lớp. Ngoài ra còn có độ đo FPS để đánh giá tốc độ của mô hình trong thời gian thực

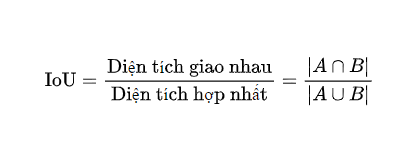
### 3.4.1 Độ đo mAP-50

Trong bài toán Object detection có 2 tiêu chí

đánh giá chính: Bouding box chuẩn xác và phân lớp chính xác. mAP-50 rất phù hợp để thể hiện 2 tiêu chí đánh giá này.

#### 3.4.1.1 IoU -50

Là độ đo độ trùng nhau giữa **bounding box dự đoán** và **bounding box thật (ground truth)**.

Tính bằng công thức:  


Nếu IoU ≥ 0.5, thì đó là True Positive (TP).

Nếu IoU < 0.5 → False Positive (FP).  
Nếu có ground truth không được phát hiện → False Negative (FN).

#### 3.4.1.2 Precision và Recall

Là diện tích dưới đường cong **Precision - Recall** cho một lớp (label) cụ thể.

**Precision**: Thể hiện độ tin cậy của model. Tức là tỷ lệ dự đoán đúng trong số tất cả dự đoán.

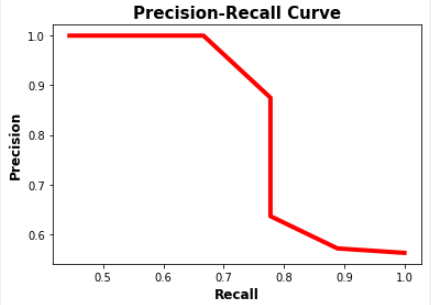
Precision =TP/(TP+FP)

**Recall**: Thể hiện độ nhạy của model. Tức là tỷ lệ dự đoán đúng trong số tất cả đối tượng thật.

Recall = TP/(TP+ FN)

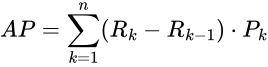
#### 3.4.1.3 Average Precision (AP)

Precision và Recall thay đổi theo từng giá trị của ngưỡng Confidence. Để theo dõi toàn bộ sự thay đổi này, ta sử dụng **Precision-Recall Curve** — đường cong thể hiện các cặp giá trị (Recall, Precision) tương ứng với mỗi mức threshold.



*Hình 11. biểu đồ Recall, Precision*

Để đánh giá chất lượng mô hình dựa trên đường cong này, ta sử dụng chỉ số **AUC (Area Under the Curve)** – diện tích dưới đường cong. Đối với Precision-Recall Curve, AUC còn được gọi là **Average Precision (AP)** và được tính theo công thức:



Trong đó R và P lần lượt là Recall và Precision tại mức thứ k

Khi AP càng cao, điều đó cho thấy cả R và P đều tốt ngay cả trên những mức khác nhau, do đó model được đánh giá tốt

Khi AP thấp, R và P thấp do đó model không được tốt

#### 3.4.1.4 mAP-50 (mean Average Precision)

Được định nghĩa là trung bình tất cả các độ đo AP trên tất cả các class. Vì lấy ngưỡng IoU là 0.5 nên được gọi là mAP-50.

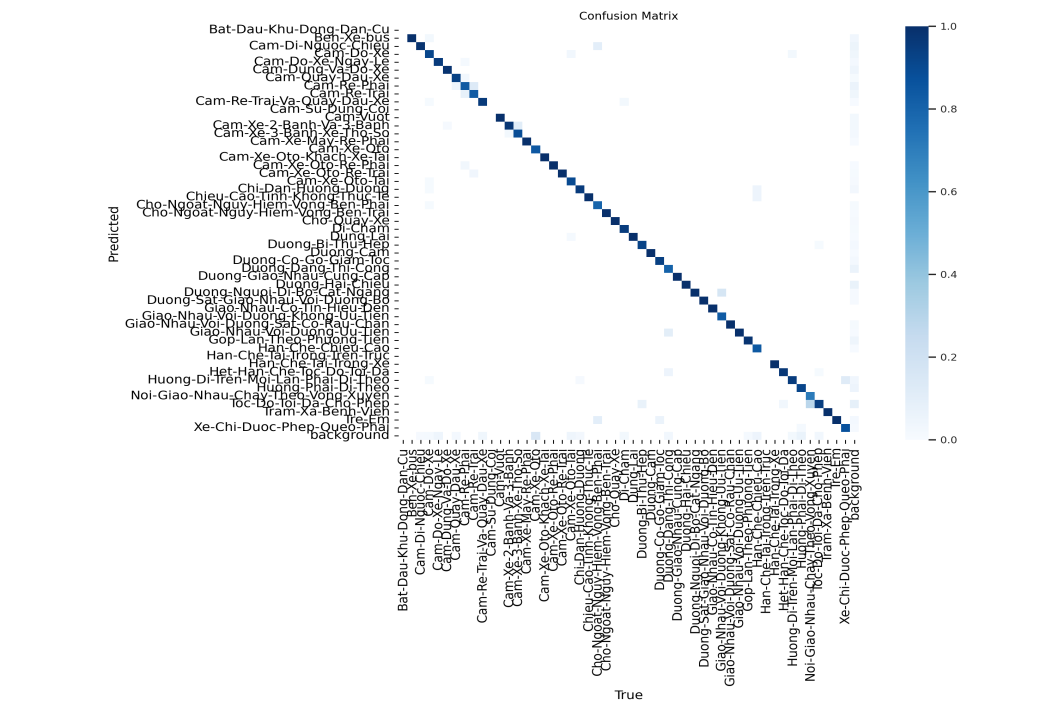
Công thức:



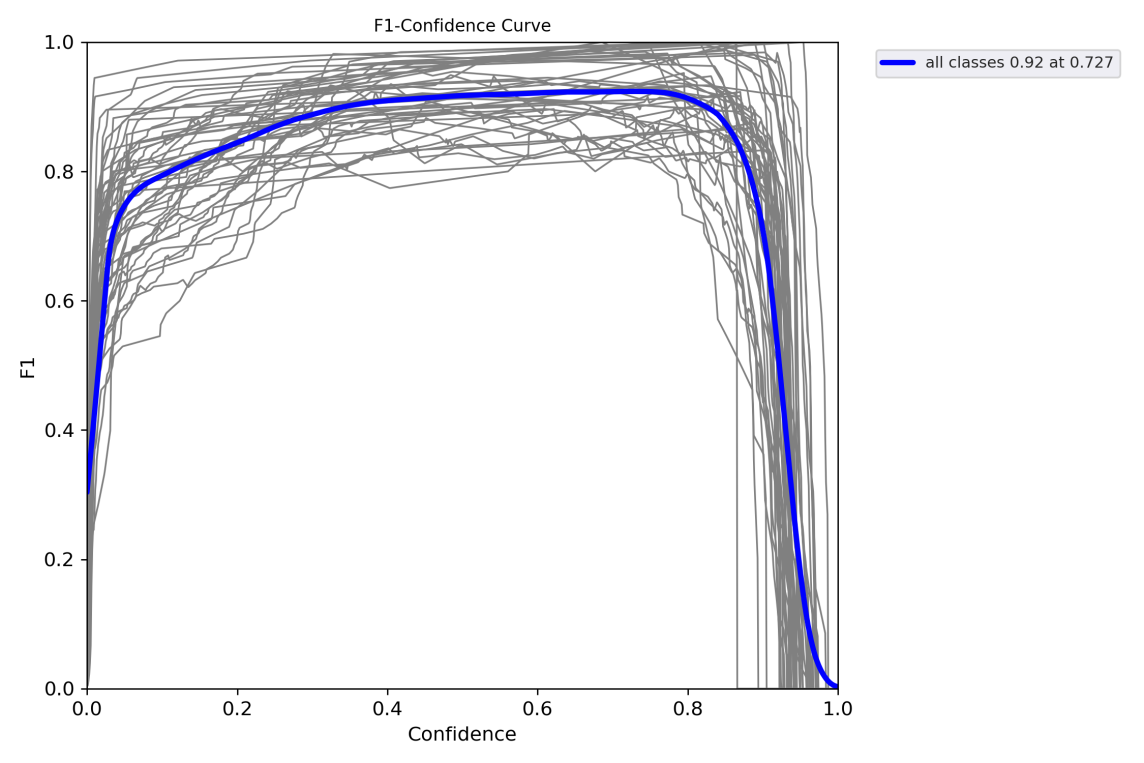
Vì thế công thức này thể hiện tốt mức độ chính xác của mô hình YoLo trong bài toán phát hiện đối tượng

### 3.5 Đánh giá các mô hình

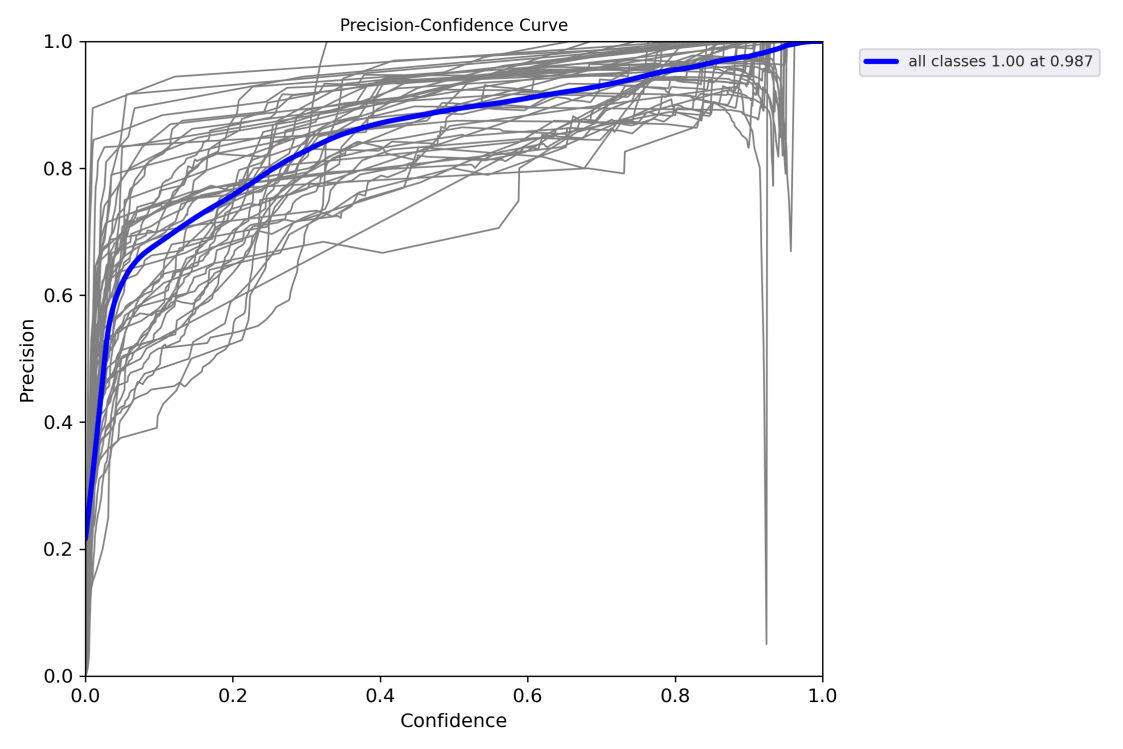
#### 3.5.1 YOLOv5

****

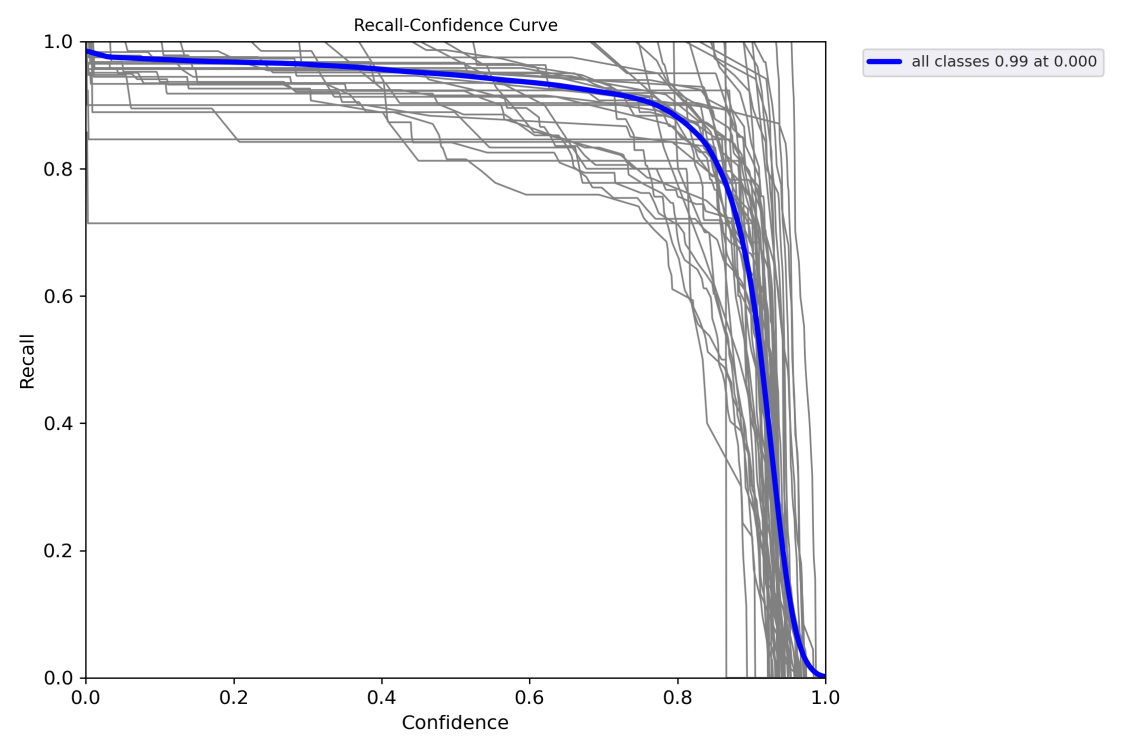
*Hình 11. Biểu đồ* Confusion Matrix Normalized của YOLOv5n

****

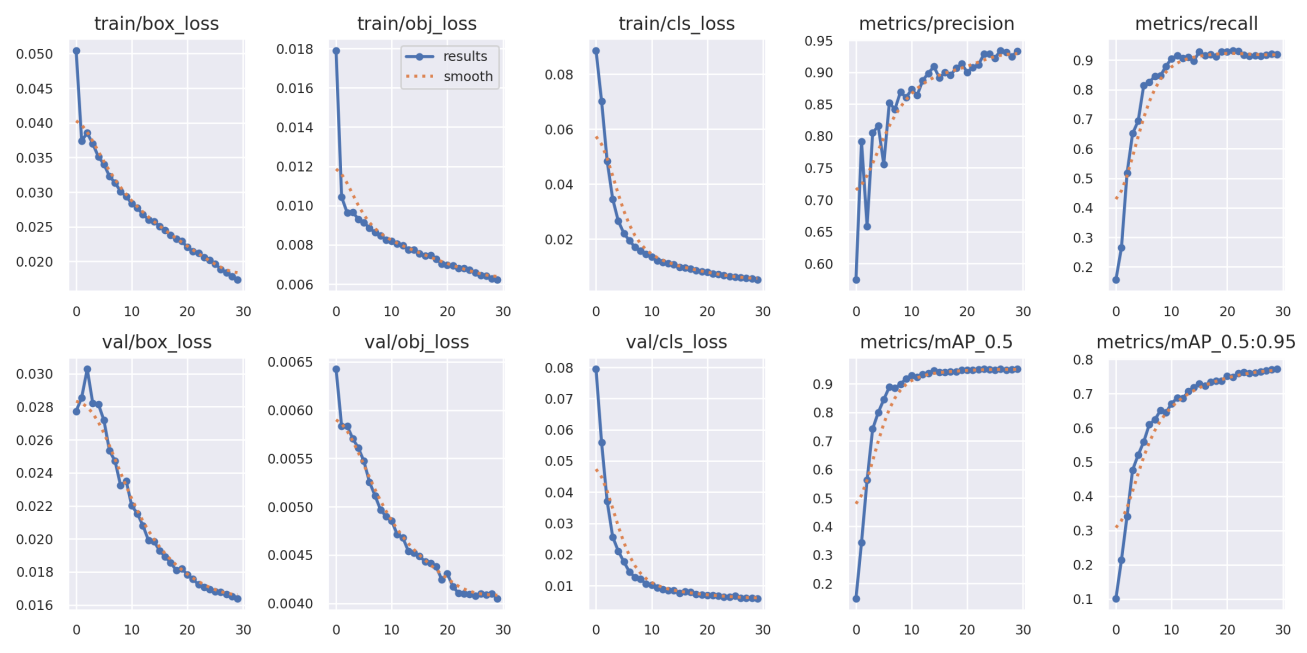
*Hình 12. Biểu đồ F1 của YOLOv5n*

****

*Hình 13. Biểu đồ Precision- confident của YOLOv5n*

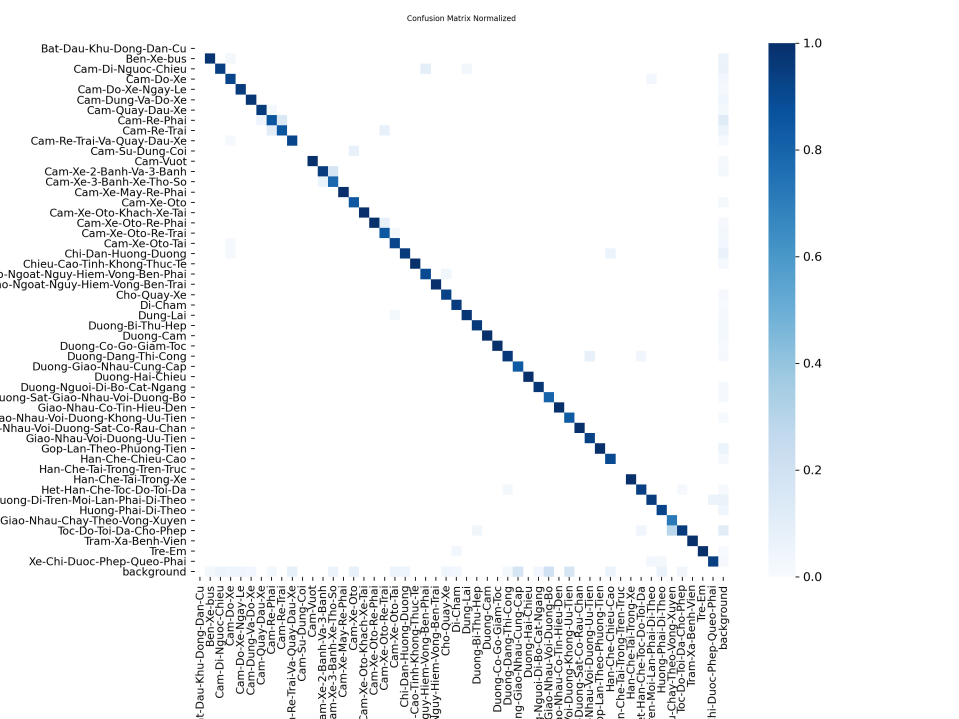
****

*Hình 14. Biểu đồ Recall-Confidence của YOLOv5n*

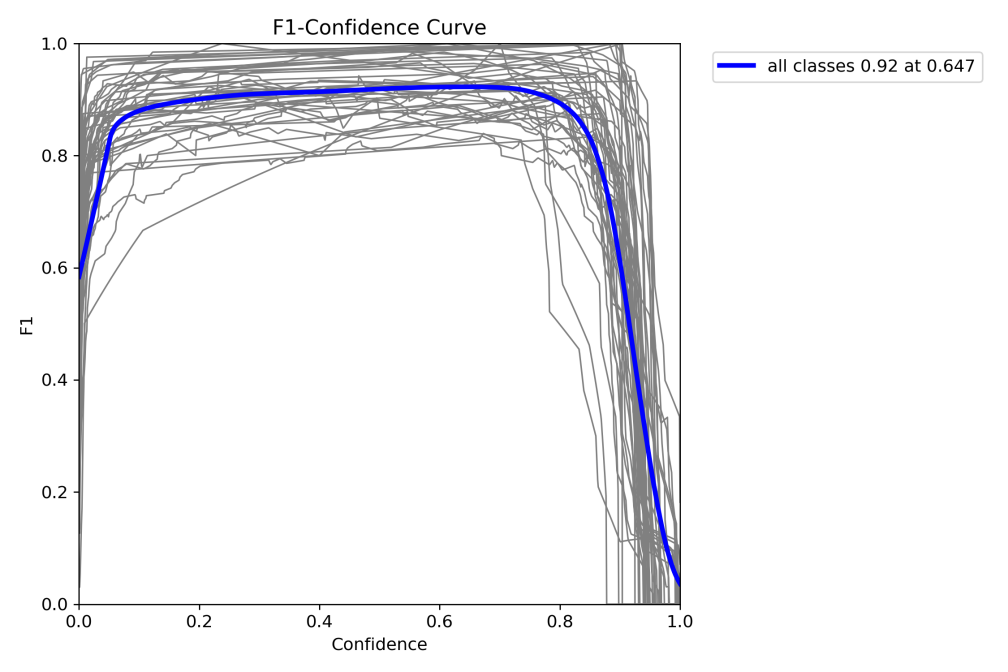
****

*Hình 15. Biểu đồ Resut của YOLOv5n*

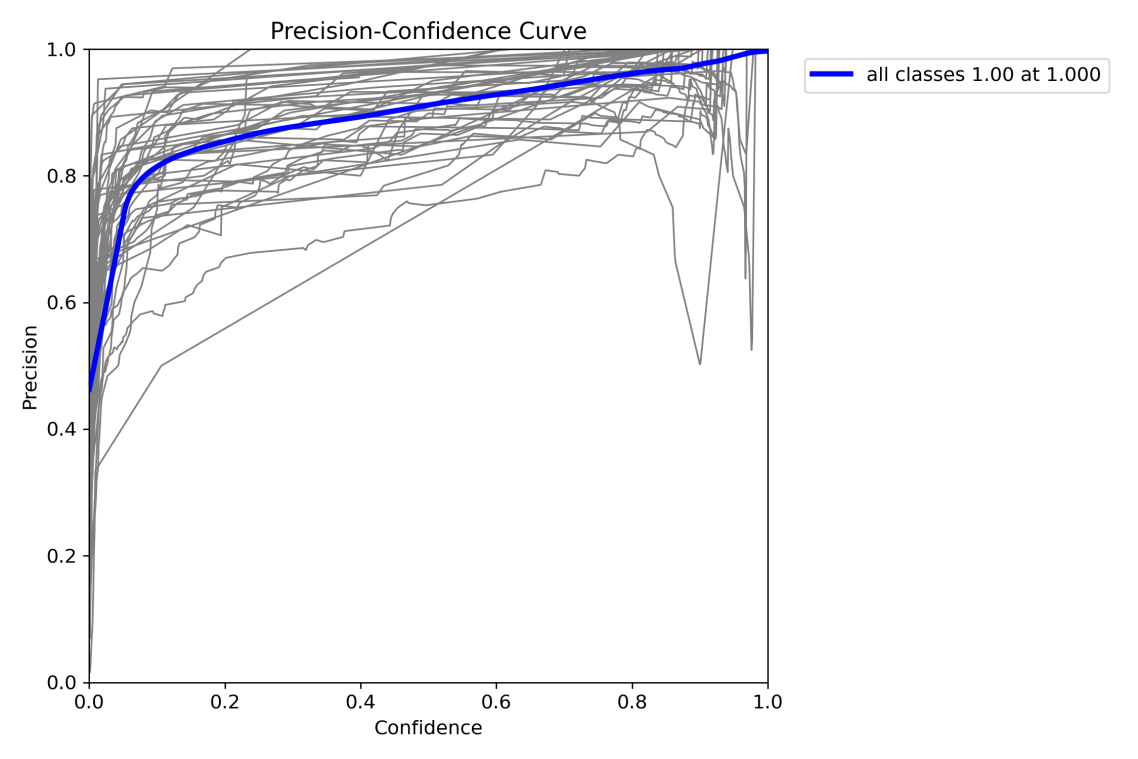
#### 3.5.2 YOLOv8

****

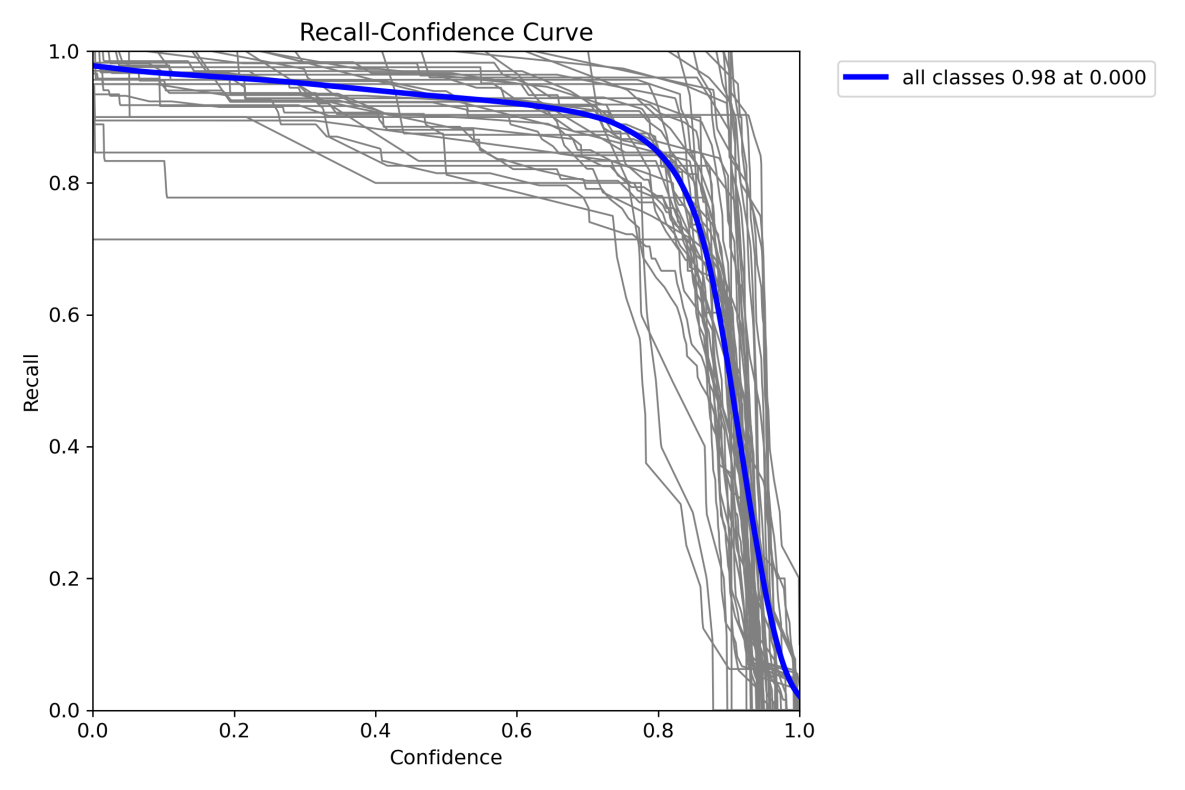
*Hình 16. Biểu đồ* Confusion Matrix Normalized của YOLOv8n

****

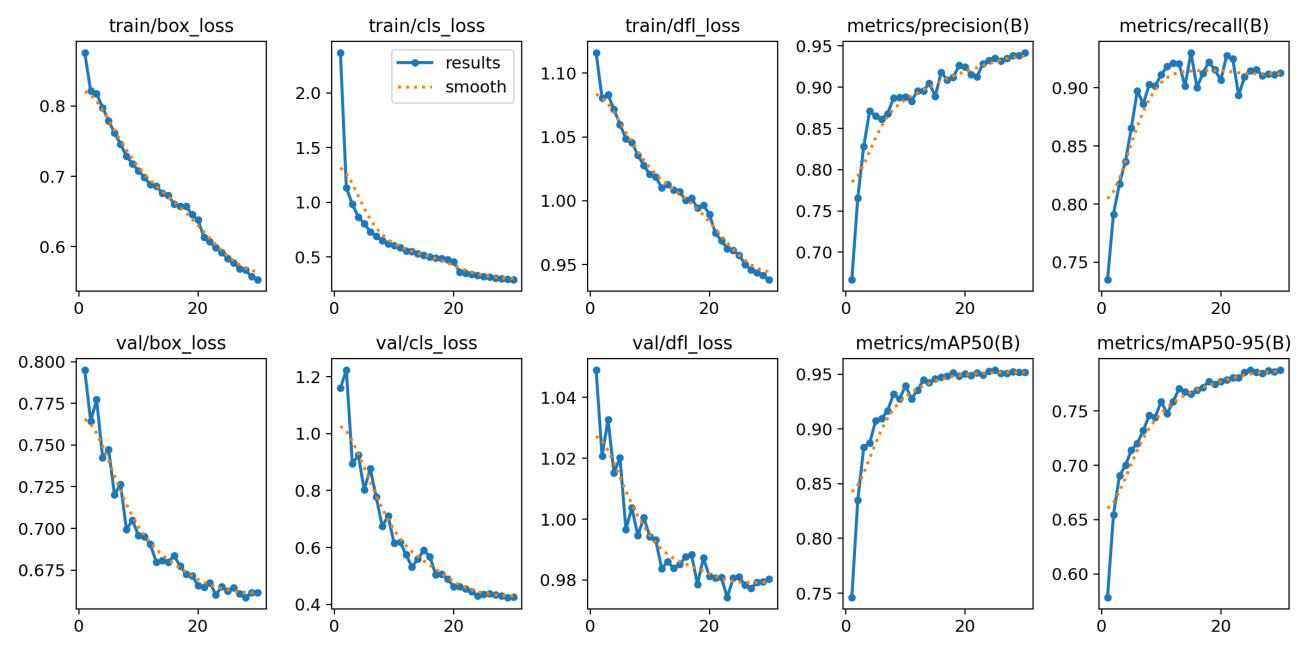
*Hình 17. Biểu đồ F1 của YOLOv8n*

****

*Hình 18. Biểu đồ Precision- confident của YOLOv8n*

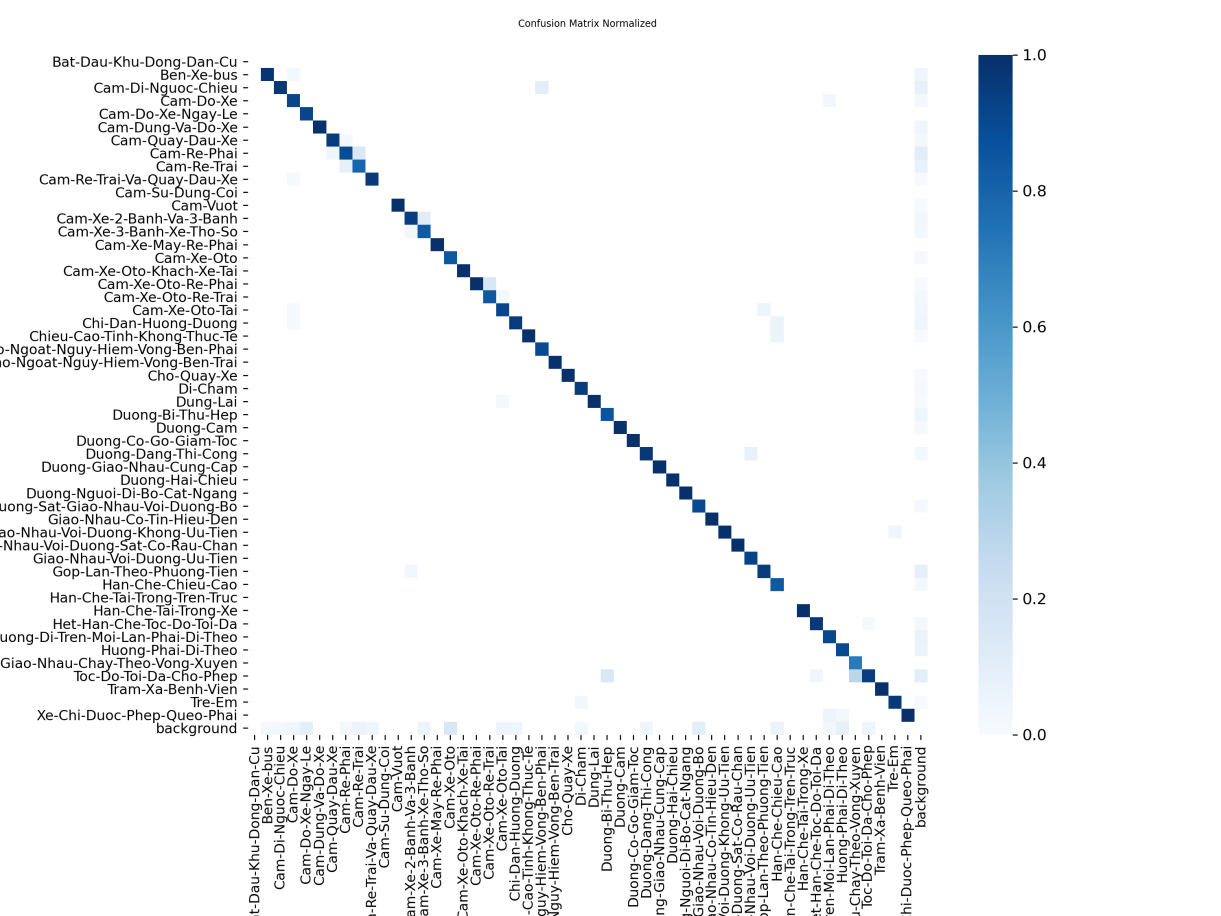
****

*Hình 19. Biểu đồ Recall-Confidence của YOLOv5n*

****

*Hình 20. Biểu đồ Resut của YOLOv5n*

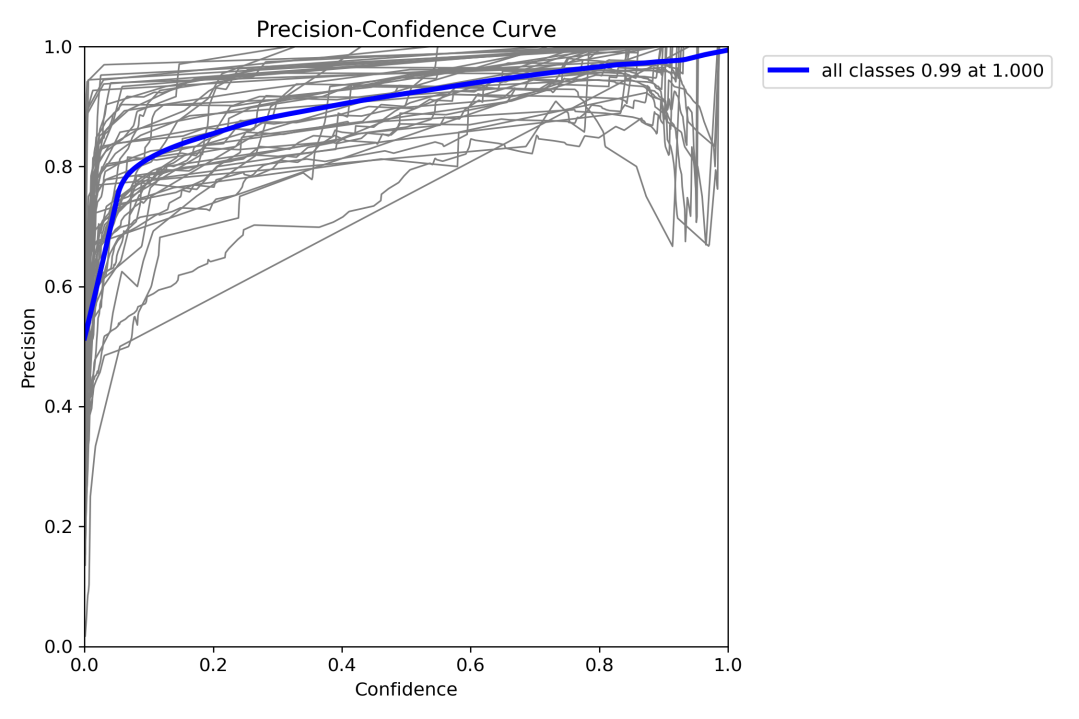
#### 3.5.3 YOLO11

****

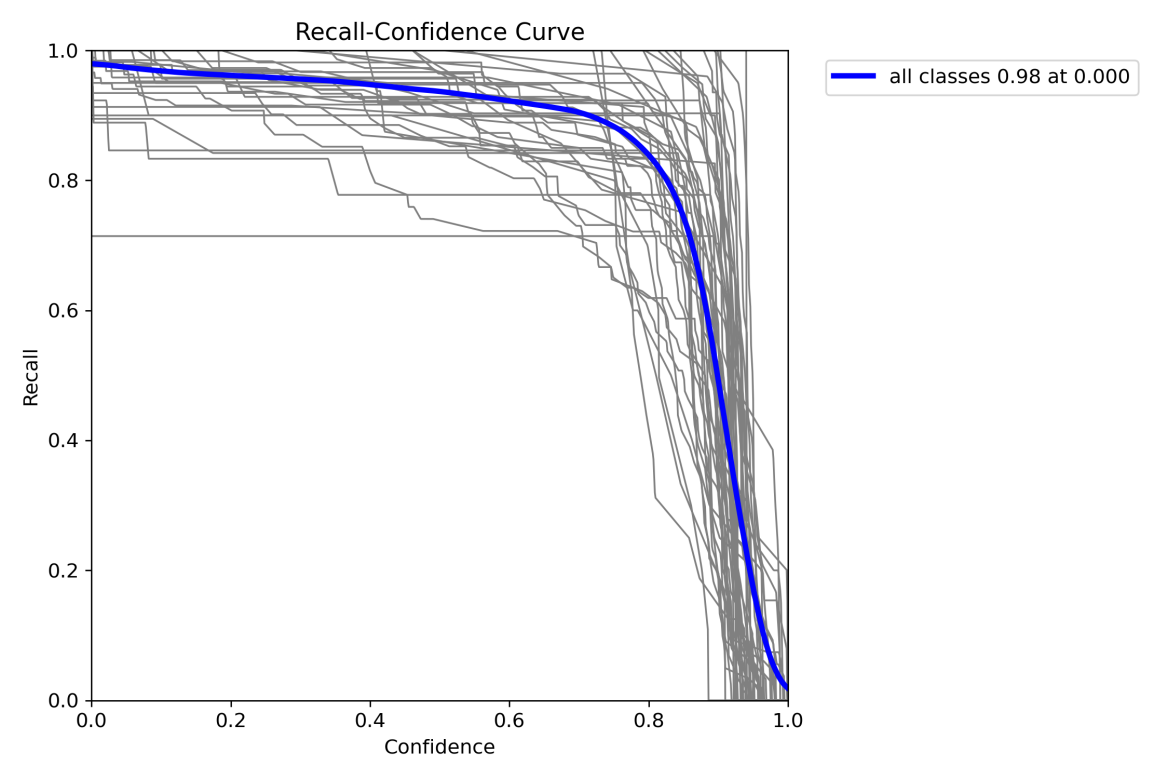
*Hình 21. Biểu đồ* Confusion Matrix Normalized của YOLO11n

****

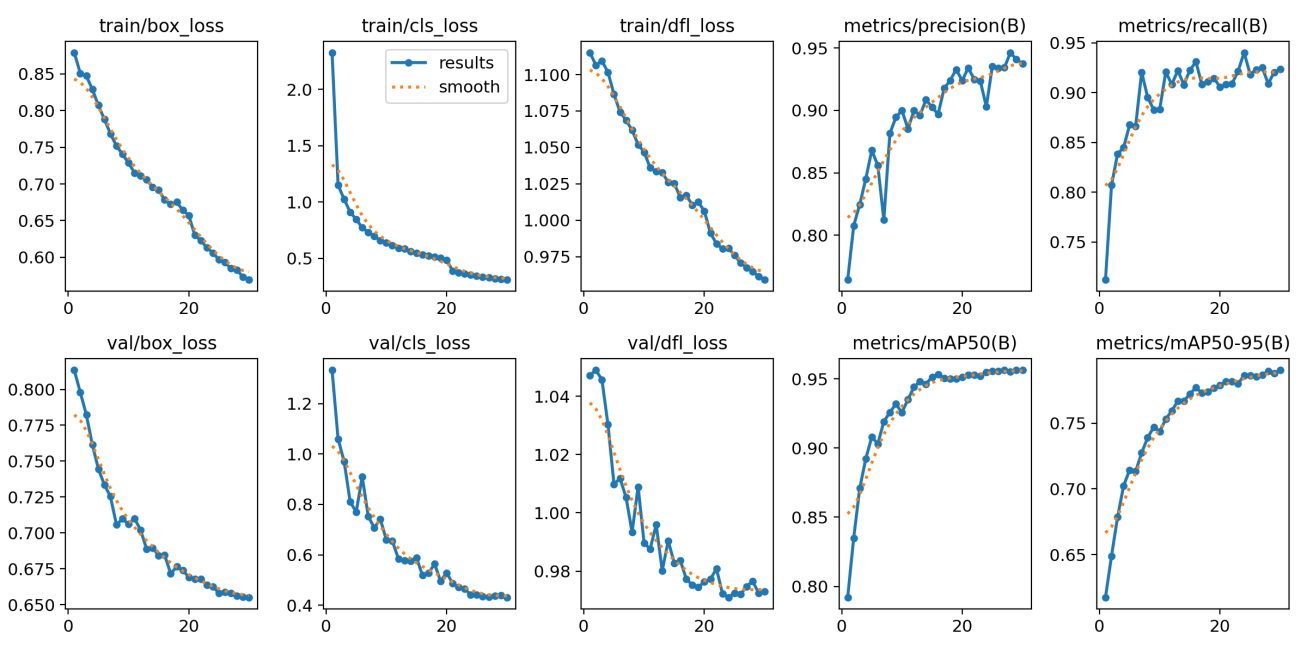
*Hình 22. Biểu đồ F1 của YOLO11n*

****

*Hình 23. Biểu đồ Precision- confident của YOLO11n*

****

*Hình 24. Biểu đồ Recall-Confidence của YOLO11n*

****

*Hình 25. Biểu đồ Resut của YOLO11n*

#### 3.5.4 so sánh 3 mô hình

|  |  |  |  |
| --- | --- | --- | --- |
|  | **metrics/precision(B)** | **metrics/recall(B)** | **metrics/mAP50(B)** |
| **YOLOV5** | 0.934 | 0.913 | 0.951 |
| **YOLOV8** | 0.934 | 0.914 | 0.953 |
| **YOLO11** | 0.937 | 0.923 | 0.956 |

Mô hình **YOLOv5** là một trong những phiên bản phổ biến và được sử dụng rộng rãi nhất trong các bài toán phát hiện đối tượng nhờ vào kiến trúc nhẹ, tốc độ xử lý nhanh và độ chính xác cao. Theo kết quả đánh giá, YOLOv5 đạt độ chính xác (precision) là **0.934**, độ bao phủ (recall) là **0.913** và mAP@50 là **0.951**. Những chỉ số này thể hiện YOLOv5 có hiệu suất tốt và ổn định trong việc phát hiện đối tượng.

Mô hình **YOLOv8** là phiên bản cải tiến hiện đại hơn so với YOLOv5, sử dụng kiến trúc **anchor-free** và tích hợp nhiều tối ưu hóa mới nhằm nâng cao hiệu suất và độ linh hoạt. Kết quả đánh giá cho thấy YOLOv8 đạt precision là **0.934**, recall là **0.914** và mAP@50 là **0.953** – nhỉnh hơn một chút so với YOLOv5. Điều này chứng tỏ YOLOv8 không chỉ giữ được độ chính xác mà còn cải thiện khả năng bao phủ và tổng thể hiệu suất.

Mô hình **YOLO11** là phiên bản mới được phát triển với mục tiêu cải thiện độ chính xác trong việc phát hiện đối tượng. Trong kết quả thu được, YOLO11 đạt precision cao nhất là **0.937**, recall là **0.923** và mAP@50 lên tới **0.956** – đều là những giá trị cao nhất trong ba mô hình được so sánh. Điều này cho thấy YOLO11 là mô hình có hiệu suất phát hiện tốt nhất trong bài toán cụ thể này.

Vậy mô hình YOLO11 là tốt nhất trong ba mô hình vừa làm trên. Mô hình YOLOV5 ổn định, có thể dùng để phát triển app và các game cần phiên bản nhẹ. Yolov8 thì hiện đại, phù hợp khi cần mở rộng các cải tiến phát triển sau này.

# 4. Ứng dụng

## 4.1 Ý tưởng

Áp dụng mô hình yolo để train vào app điện thoại để nhận diện biển báo thông qua camera

Sử dụng thư viện CameraX xử lý luồng video kết hợp với TF-Lite

## 4.2 Cách thức thực hiện.

### **4.2.1 Chuyển đổi model**

Chuyển đổi mô hình yolo dưới dạng tflite để đưa vào app android để chạy model.

Code chuyển:

|  |
| --- |
| !python export.py --weights /kaggle/working/YOLOv5/runs/train/YOLOv5\_train/weights/best.pt --img 640 --device 0 --include *tflite* |

Model nhận được



Xác định các thông số mô hình (output, input)



Output: [1, 25200, 63] lần lượt là [batchsize, anchor\_box, class + 5]. 5 giá trị cộng thêm với class chính là tọa độ x,y; chiều dài(h) và chiều rộng (w); 1 giá trị xác suất của lớp.

Input: [1, 640, 640, 3] lần lượt là batchsize (1 ảnh); 640x640: kích thước mô hình; 3 là số kênh màu (RGB).

### 4.2.2 Xây dựng app android bằng phần mềm android studio

1. CameraX: Cung cấp luồng previewView để thu nhận ảnh từ camera và callback mỗi khung hình.
2. TFLite Interpreter: Tải và chạy mô hình .tflite với kích thước input 640×640.
3. Tiền xử lý (Preprocessing):
   * Resize khung hình về kích thước mô hình yêu cầu.
   * Chuyển ảnh sang ByteBuffer theo chuẩn float32, normalize.
4. Hậu xử lý (Postprocessing):
   * Áp dụng OBJ\_THRESH và CONF\_THRESH để lọc box.
   * Non-Max Suppression với IOU\_THRESH để loại bỏ box trùng lặp.
   * Giới hạn số box trả về tối đa MAX\_DET.
5. Hiển thị: Dùng OverlayView để vẽ bounding box và tên đối tượng lên khung camera.

### 4.2.3 Kết quả thực nghiệm

****

OBJ: ngưỡng độ tin cậy tính bouding box là đối tượng

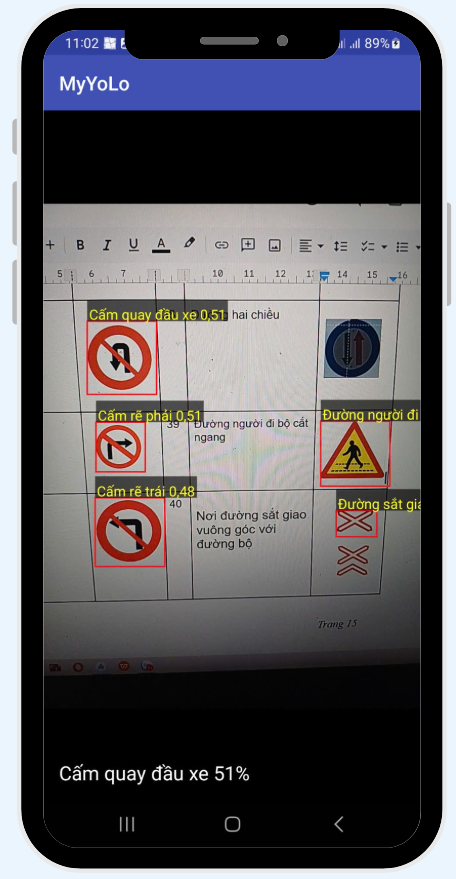
CONF: ngưỡng độ tin cậy tổng quát (Objectness x Class\_confident)

IoU: ngưỡng giữ lại các bouding box có độ IoU cao, tránh trùng lặp bouding box.

MAX\_DET: giới hạn đối tượng nhận diện

SKIP\_FRAMES: tỉ lệ bỏ qua khung hình, càng thấp độ trễ càng tốt.

Sau nhiều lần thử nghiệm thay đổi các tham số ngưỡng này. Nhóm em cho ra kết quả tốt nhất



## 4.3. Đánh giá và tối ưu

Hiệu năng: Tốt, nhưng model còn nhầm lẫn một số lớp.

Mở rộng:

Cải thiện chất lượng mô hình tflite để cho ra kết quả tốt nhất.

Cải thiện về giao diện người dùng, thêm nhiều tính năng như: chọn được phân loại…

Tài liệu Tham khảo

Cục Cảnh sát giao thông – Bộ Công an. (2019). *Thống kê phương tiện giao thông đăng ký đến năm 2019*. Truy cập tại:<http://www.csgt.vn>

an, L., Huangfu, T., Wu, L., & Chen, W. (2021). *Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identification*. The Third Affiliated Hospital of Southern Medical University. Research Article. <https://assets-eu.researchsquare.com/files/rs-668895/v1_covered.pdf?c=1631875157>

Ultralytics. *YOLOv8 Models*. Ultralytics, [https://docs.ultralytics.com/vi/models/YOLOv8/#performance-metrics](https://docs.ultralytics.com/vi/models/yolov8/#performance-metrics). Accessed 22 Apr. 2025.ci

Ultralytics. YOLO11 Models. Ultralytics, <https://docs.ultralytics.com/vi/models/yolo11/>. Accessed 18 May.

Ultralytics. YOLOv5 Models. Ultralytics, [https://docs.ultralytics.com/vi/models/YOLOv5/](https://docs.ultralytics.com/vi/models/yolov5/). Accessed 18 May. 2025.

Ultralytics. TFLite Integration. Ultralytics, <https://docs.ultralytics.com/integrations/tflite/>. Accessed 18 May. 2025.

Singla, Aarohi. Object Detection Android App. GitHub, <https://github.com/AarohiSingla/Object-Detection-Android-App> Accessed 18 May. 2025.