



**TRƯỜNG ĐẠI HỌC BÁCH KHOA  
KHOA CÔNG NGHỆ THÔNG TIN**



## **BÁO CÁO PBL5 - ĐỒ ÁN KỸ THUẬT MÁY TÍNH**

### **ĐỀ TÀI: XÂY DỰNG HỆ THỐNG NHẬN DIỆN BIỂN BÁO GIAO THÔNG**

**Doanh nghiệp hỗ trợ hướng dẫn: Enclave**

**Cán bộ doanh nghiệp hướng dẫn: Trần Văn Huy**

**Giảng viên đồng hướng dẫn: TS. Ninh Khánh Duy**

**Sinh viên thực hiện:**

<b>Nhóm</b>	<b>4-CLC</b>	
<b>Họ và tên sinh viên</b>	<b>MSSV</b>	<b>Lớp học phần</b>
<b>Phan Mạnh Cường</b>	<b>102200250</b>	<b>20.15A</b>
<b>Lê Trung Hoàng</b>	<b>102200256</b>	
<b>Nguyễn Kiều Quốc</b>	<b>102200282</b>	
<b>Nguyễn Hoàng Quân</b>	<b>102200281</b>	

**ĐÀ NẴNG, 6/2023**

## TÓM TẮT ĐỒ ÁN

Trong quá trình điều khiển và quản lý giao thông, biển báo giao thông đóng vai trò quan trọng trong việc truyền đạt thông tin, cảnh báo và hướng dẫn cho người tham gia giao thông. Tuy nhiên, việc nhận diện và hiểu đúng các biển báo giao thông đôi khi có thể gặp khó khăn, đặc biệt là trong tình huống giao thông phức tạp hoặc khi gặp phải những biển báo mới. Vì vậy, trong đồ án này, chúng em sẽ thực hiện xây dựng một hệ thống phát hiện và nhận dạng biển báo giao thông trong thời gian thực, từ đó cung cấp thông tin chính xác và kịp thời cho người dùng.

Để đạt được mục tiêu này, chúng em đã nghiên cứu và áp dụng các phương pháp và thuật toán nhận diện ảnh, kết hợp với các công nghệ tiên tiến như học sâu (deep learning) và xử lý ảnh số. Đồng thời, chúng em cũng tập trung vào việc xây dựng một cơ sở dữ liệu đầy đủ và đa dạng, chứa thông tin về các biển báo giao thông thường gặp trong thực tế.

Sau khi nghiên cứu và thử nghiệm, hệ thống đã hoạt động tốt. Tuy nhiên, còn một số điểm thiếu sót nên hệ thống vẫn chưa hoàn hảo. Chúng em sẽ tiếp tục phát triển và hoàn thiện trong tương lai.

## BẢNG PHÂN CÔNG NHIỆM VỤ

Sinh viên	Các nhiệm vụ	Kết quả
<b>Lê Trung Hoàng</b>	Phân công công việc, đảm bảo tiến độ đồ án	Đã hoàn thành
	Thu thập và gán nhãn dữ liệu	Đã hoàn thành
	Tìm kiếm và chuẩn bị phần cứng	Đã hoàn thành
	Xây dựng ứng dụng desktop app	Đã hoàn thành
	Triển khai và kiểm thử hệ thống	Đã hoàn thành
	Viết báo cáo và làm slide	Đã hoàn thành
<b>Nguyễn Kiều Quốc</b>	Thu thập và gán nhãn dữ liệu	Đã hoàn thành
	Nghiên cứu mô hình nhận dạng	Đã hoàn thành
	Huấn luyện mô hình	Đã hoàn thành
	Viết báo cáo và làm slide	Đã hoàn thành
<b>Nguyễn Hoàng Quân</b>	Thu thập và gán nhãn dữ liệu	Đã hoàn thành
	Nghiên cứu mô hình nhận dạng	Đã hoàn thành
	Huấn luyện mô hình	Đã hoàn thành
	Viết báo cáo và làm slide	Đã hoàn thành
<b>Phan Mạnh Cường</b>	Thu thập và gán nhãn dữ liệu	Đã hoàn thành
	Tăng cường dữ liệu	Đã hoàn thành
	Hỗ trợ xây dựng ứng dụng desktop app	Đã hoàn thành
	Viết báo cáo và làm slide	Đã hoàn thành

# MỤC LỤC

TÓM TẮT ĐỒ ÁN .....	2
BẢNG PHÂN CÔNG NHIỆM VỤ .....	3
MỤC LỤC .....	4
DANH SÁCH HÌNH ẢNH .....	6
DANH SÁCH CÁC BẢNG .....	8
1. GIỚI THIỆU .....	9
1.1. Thực trạng sản phẩm .....	9
1.2. Các vấn đề cần giải quyết .....	9
1.2.1. Phân tích bài toán nhận diện biển báo .....	9
1.2.2. Các vấn đề cần giải quyết .....	10
1.3. Đề xuất giải pháp tổng quan .....	10
2. GIẢI PHÁP .....	11
2.1. Giải pháp phần cứng .....	11
2.1.1. Sơ đồ hoạt động tổng quan hệ thống .....	11
2.1.2. Linh kiện sử dụng .....	11
2.2. Giải pháp nhận diện biển báo .....	15
2.2.1. Giới thiệu về YOLO .....	15
2.2.2. Kiến trúc YOLOv5 .....	15
2.2.3. Backbone .....	16
2.2.4. Neck .....	19
2.2.5. Head .....	21
2.2.6. Hàm kích hoạt (Activation function) .....	22
2.2.7. Loss function .....	23
2.2.8. Loss scalling .....	24
2.2.9. Hàm tối ưu hóa (Optimization function) .....	24
2.2.10. Batch normalization .....	24
2.2.10. Anchor box .....	25
2.2.11. Tiến hóa siêu tham số (Hyperparameters Evolution) .....	25
2.3. Giải pháp phần mềm .....	27

2.3.1. Phát biểu bài toán .....	27
2.3.2. Sơ đồ usecase.....	28
3. KẾT QUẢ.....	28
3.1. Nhận diện biến báo .....	28
3.1.1. Tập dữ liệu.....	28
3.1.2. Huấn luyện.....	32
3.1.3. Các công thức đánh giá kết quả.....	33
3.3. Kết quả mô hình .....	36
3.3.1. Kết quả định tính .....	36
3.3.2. Kết quả định lượng .....	36
3.4. Kết quả thực nghiệm.....	38
3.4. Kết quả phần mềm.....	40
4. KẾT LUẬN .....	41
4.1. Đánh giá.....	41
4.1.1. Kết quả đạt được.....	41
4.1.2. Hạn chế .....	41
4.2. Hướng phát triển.....	42
5. DANH MỤC TÀI LIỆU THAM KHẢO .....	42

## DANH SÁCH HÌNH ẢNH

Hình 1: Sơ đồ hệ thống nhận dạng đối tượng.....	9
Hình 2: Sơ đồ hoạt động tổng quan hệ thống .....	11
Hình 3: Quá trình thực hiện giải pháp .....	15
Hình 4: Kiến trúc YOLOv5 .....	16
Hình 5: Kiến trúc CSPNet .....	17
Hình 6: Kiến trúc Darknet53 .....	18
Hình 7: Kiến trúc CPP-Darknet53.....	19
Hình 8: Kiến trúc SPPF .....	20
Hình 9: Kiến trúc PAN .....	20
Hình 10: Output của YOLOv5 .....	21
Hình 11: Công thức tính tọa độ bounding box .....	22
Hình 12: Hàm SiLU và hàm ReLU .....	23
Hình 13: Giải thuật di truyền.....	26
Hình 14: Hyperparameters của YOLOv5.....	27
Hình 15: Sơ đồ usecase hệ thống.....	28
Hình 16: Phân bố dữ liệu trước khi tăng cường .....	31
Hình 17: Phân bố dữ liệu sau khi tăng cường .....	32
Hình 18: Cấu trúc thư mục dataset.....	32
Hình 19: Hyperparameters tối ưu .....	33
Hình 20: Intersection Over Union (IOU) .....	34
Hình 21: Confusion matrix .....	35
Hình 22: Công thức tính AP .....	35
Hình 23: Công thức tính mAP.....	36
Hình 24: Kết quả nhận diện biên báo .....	36
Hình 25: Giao diện ứng dụng desktop.....	40
Hình 26: Giao diện nhận diện biên báo .....	41



## DANH SÁCH CÁC BẢNG

Bảng 1: Đề xuất giải pháp tổng quan .....	10
Bảng 2: Các linh kiện sử dụng.....	11
Bảng 3: Chi phí linh kiện.....	14
Bảng 4: Bảng các class trong dataset .....	29
Bảng : Kết quả mô hình nhận diện biển báo .....	36
Bảng 6: Thống kê khoảng cách nhận dạng.....	38



# 1. GIỚI THIỆU

## 1.1. Thực trạng sản phẩm

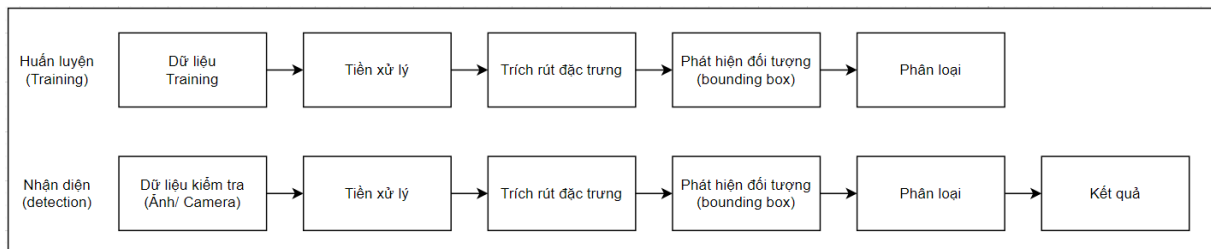
Hiện nay, sản phẩm về nhận diện biển báo giao thông trong thời gian thực đã đạt được sự phát triển đáng kể và được áp dụng trong nhiều ứng dụng thực tế. Nhiều sản phẩm đã đạt được độ chính xác cao trong việc nhận dạng các biển báo thông qua sự kết hợp giữa các công nghệ học máy và học sâu, đồng thời cải thiện đáng kể về tốc độ xử lý trong thời gian thực, đáp ứng yêu cầu của các ứng dụng thực tế như xe tự hành, giám sát giao thông, v.v.

Tuy nhiên, mặc dù đã có sự phát triển tích cực, sản phẩm nhận diện biển báo giao thông trong thời gian thực vẫn đối mặt với một số thách thức. Điều kiện ánh sáng yếu, biển báo bị che khuất, đa dạng biển báo trong các quốc gia khác nhau, và các tình huống phức tạp như biển báo di động hay biển báo nằm trong khuôn hình động là những thách thức cần được giải quyết để cải thiện hiệu suất và độ tin cậy của hệ thống

## 1.2. Các vấn đề cần giải quyết

### 1.2.1. Phân tích bài toán nhận diện biển báo

Một hệ thống nhận dạng đối tượng thông thường gồm các bước sau đây:



*Hình 1: Sơ đồ hệ thống nhận dạng đối tượng*

- **Tiền xử lý:** Bước này nhằm mục đích lọc nhiễu, nâng cao chất lượng ảnh, trong bước này bao gồm các bước : Căn chỉnh ảnh, chuẩn hóa ánh sáng
- **Trích rút đặc trưng:** Ảnh sẽ được đưa qua 1 mạng neural network (thường là 1 mạng CNN) để trích rút đặc trưng
- **Phát hiện đối tượng và phân loại:** Ở bước này 1 phương pháp phát hiện và phân loại vật thể như SSD, YOLO, Faster R-CNN, ... được sử dụng. Kết quả là một danh sách các bounding box chứa đối tượng được xác định và định vị trong ảnh, cùng với class predictions và confidence scores tương ứng

### 1.2.2. Các vấn đề cần giải quyết

- Cần có các thiết bị phần cứng để thu nhận dữ liệu.
- Cần thu thập và gán nhãn dữ liệu
- Phát hiện và nhận diện nhiều loại biển báo.
- Cần ứng dụng tương tác để người dùng có thể điều khiển camera, kiểm tra kết quả.
- Hệ thống chạy theo thời gian thực

### 1.3. Đề xuất giải pháp tổng quan

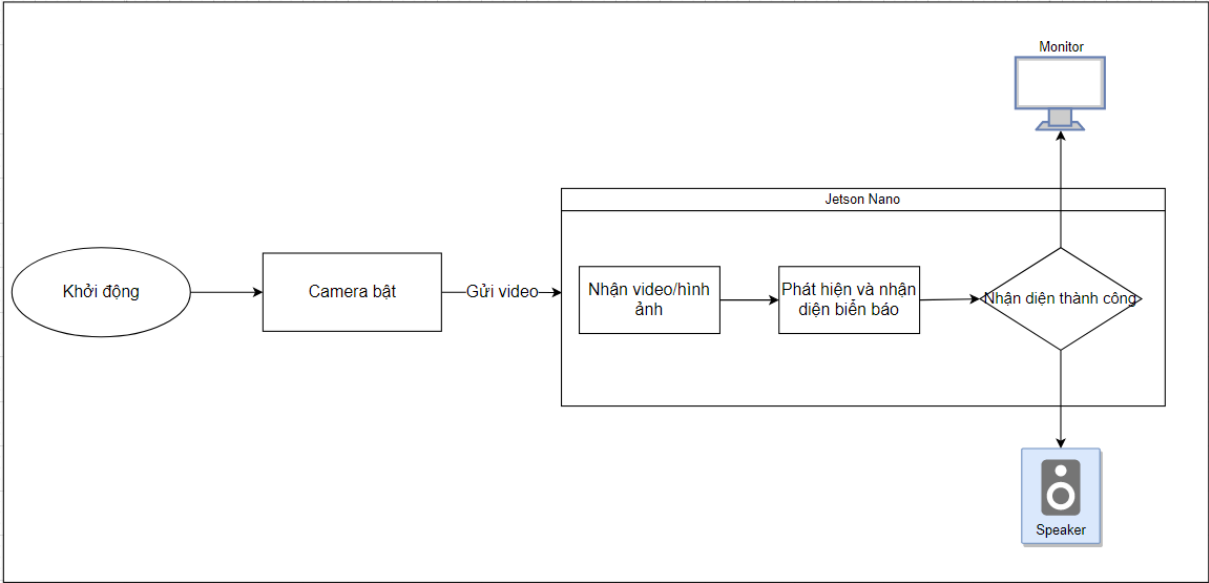
*Bảng 1: Đề xuất giải pháp tổng quan*

Vấn đề	Giải pháp đề xuất
Phần cứng	Jetson Nano Developer Kit B01 Camera IMX477-160 12.3MP Màn hình
Dữ liệu	Thu thập từ các nguồn dữ liệu có sẵn công khai Tự thu thập và gán nhãn
Nhận diện biển báo	Xây dựng và huấn luyện model nhận diện biển báo Thử nghiệm với các model: YOLOv5, YOLOv6, YOLOv7, ... Huấn luyện trên Google Colab
Ứng dụng	Xây dựng ứng dụng desktop Có chức năng bật tắt camera, sử dụng ảnh/video. Hiển thị kết quả lên màn hình và phát cảnh báo qua loa

2. GIẢI PHÁP

2.1. Giải pháp phần cứng

2.1.1. Sơ đồ hoạt động tổng quan hệ thống






Hình 2: Sơ đồ hoạt động tổng quan hệ thống



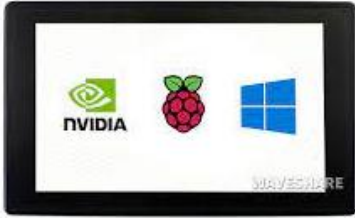

Hệ thống bao gồm Jetson nano và webcam dùng để chụp ảnh, màn hình để hiển thị kết quả và loa để phát thông báo cho người dùng

2.1.2. Linh kiện sử dụng

Bảng 2: Các linh kiện sử dụng

Tên linh kiện	Hình ảnh	Thông số kỹ thuật
Jetson Nano Developer Kit B01		Model: NVIDIA Jetson Nano Developer Kit B01 GPU: 128-core Maxwell CPU: Quad-core ARM A57 @ 1.43 GHz Bộ nhớ: 4 GB 64-bit LPDDR4 25.6 GB/s Video Encode: 4K @ 30   4x 1080p @ 30   9x 720p @ 30 (H.264/H.265) Video Decode: 4K @ 60   2x 4K @ 30   8x 1080p @ 30   18x 720p @ 30 (H.264/H.265) Camera: 2x MIPI CSI-2 DPHY lanes

		<p>Kết nối: Gigabit Ethernet, M.2 Key E</p> <p>Display: HDMI and display port</p> <p>USB: 4x USB 3.0, USB 2.0 Micro-B</p> <p>Cơ khí: 69 mm x 45 mm, 260-pin edge connector</p>
<p>Camera</p> <p>IMX477-160</p> <p>12.3MP</p>		<p>Cảm biến:</p> <ul style="list-style-type: none"> <li>- Model: Sony IMX477</li> <li>- Độ phân giải: 12.3MP, 4056 × 3040</li> <li>- Chiều dài đường chéo CMOS: 7.9mm</li> <li>- pixel size: 1.55μm (H) × 1.55μm (V)</li> </ul> <p>Camera:</p> <ul style="list-style-type: none"> <li>- Ống kính (F): 2.2</li> <li>- Độ dài trọng tâm (BFL): 5.52</li> <li>- Góc nhìn (FOV): 160°(D) 118°(H) 87°(V)</li> <li>- Độ không chính xác: &lt;16%</li> </ul> <p>Mức điện áp: 3.3V</p>
<p>Jetson Nano</p> <p>Wifi Intel</p> <p>AC8265</p>		<p>Chip: Intel 8265AC</p> <p>Băng tần: 2.4GHz / 5GHz</p> <p>Tốc độ: 300Mbps / 867Mbps</p> <p>Giao thức WiFi: 802.11ac</p> <p>Phiên bản Bluetooth: 4.2</p> <p>Giao diện NIC: NGFF (M.2)</p> <p>Giao diện ăng-ten: Đầu nối IPEX</p> <p>Hệ điều hành được hỗ trợ: Linux, Windows 10 / 8.1 / 8/7</p> <p>Kích thước: 22mm × 30 mm × 2,4mm</p>

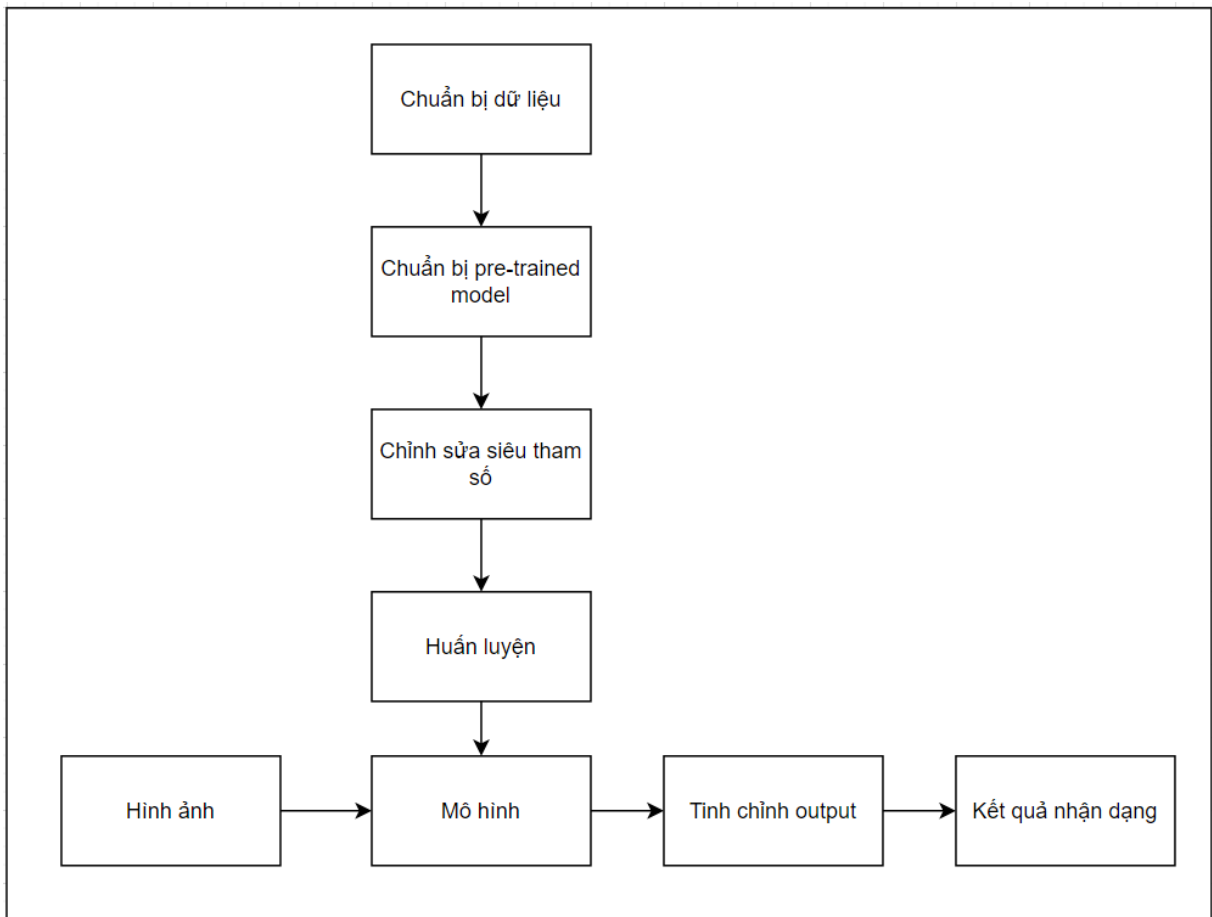
UPS Power Module B		Điện áp đầu ra: 5V Bộ nạp điện: 8.4V 2A Bus điều khiển: 12C Kích thước: 100mm x 79 mm Lỗ gắn: 2.5mm Công suất hiện tại: 4 pin
Pin Lishen 18650		Kích thước: 18 x 65 mm Điện áp: 3.7 V Dung lượng: 2000mAh Nội trở: 12-15mΩ Trọng lượng: 45g
Màn hình		Kích thước: 9 inch Sự giải: 2560x1600 Giao diện hiển thị: Mini HDMI Bảng hiển thị: IPS Cổng cảm ứng: USB Âm thanh: Hi-Fi Speaker Đầu ra audio: 3.5mm Jack
Quạt tản nhiệt		Tương thích: Máy Tính AI NVIDIA Jetson Nano B01 (2 Camera CSI Ports) / Jetson Nano Dev Kit With 16GB EMMC / Jetson Nano 2GB Điện áp sử dụng: 5VDC Kích thước: 40 x 40mm

Case bảo vệ Jetson Nano		Kích thước: 100 x 130 x 80 mm Chất liệu: nhựa tổng hợp
----------------------------	---	---

*Bảng 3: Chi phí linh kiện*

Tên linh kiện	Giá tiền gốc (đ)	Ghi chú
1 x Jetson Nano Developer Kit B01	4,700,000	Mượn
1 x Camera IMX477-160 12.3MP	2,200,000	Mua cũ (-50%)
1 x Jetson Nano Wifi Intel AC8265	450,000	Mua
1 x UPS Power Module B	750,000	Mua
4 x Pin Lishen 18650	30,000	Mua
1 x Màn hình cảm ứng	3,650,000	Mượn
1 x Quạt tản nhiệt	65,000	Mua
1 x Case bảo vệ	500,000	Tự thiết kế và in 3D
		<b>Thành tiền:</b> <b>2,895,000</b>

## 2.2. Giải pháp nhận diện biển báo



Hình 3: Quá trình thực hiện giải pháp

### 2.2.1. Giới thiệu về YOLO

YOLO là một mô hình sử dụng cho việc phát hiện, nhận dạng, phân loại đối tượng. YOLO được xây dựng dựa trên mạng nơron tích chập (CNN) và đã được phát triển qua nhiều phiên bản khác nhau chẳng hạn như YOLOv5, YOLOv6, YOLOv7. Trong đồ án này nhóm sẽ tập trung phân tích chủ yếu kiến trúc của YOLOv5 và so sánh hiệu suất với các mô hình YOLO khác.

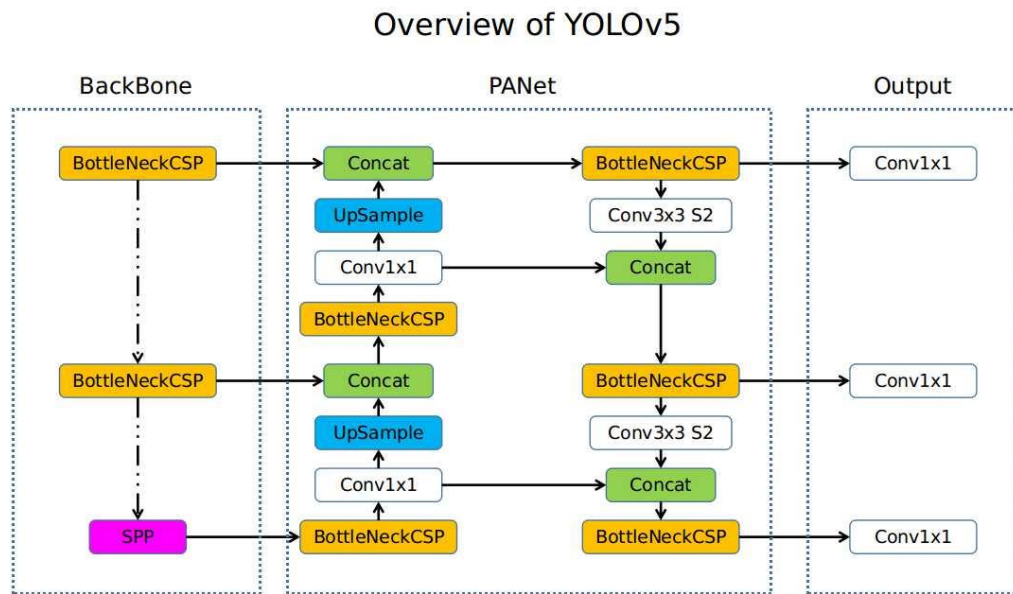
Mô hình YOLOv5 áp dụng một số thuật toán phát hiện vật thể nhanh, tối ưu hóa các phép tính toán giúp tăng tốc độ nhận diện và tăng độ chính xác.

### 2.2.2. Kiến trúc YOLOv5

Cấu trúc nhận diện vật thể của YOLOv5 có 3 phần:

- **Backbone:** Ở phần Backbone, YOLOv5 sử dụng một mô hình pre-trained của một mô hình học chuyển (transfer learning) để trích rút các đặc trưng. Các mô hình học chuyển thường là VGG16, Resnet-50,... Mô hình học chuyển được áp dụng trong YOLOv5 là CSP-Darknet53

- **Neck:** Ở phần Neck, YOLOv5 sử dụng 1 biến thể của Spatial Pyramid Pooling (SPP) là SPP-Fast (SPPF) để đưa các bản đồ đặc trưng về cùng kích thước. Đồng thời, sử dụng Path Aggregation Network (PAN) để để tích hợp các đặc trưng từ các tầng khác nhau và tạo ra các đặc trưng tốt hơn cho việc phát hiện vật thể
- **Head:** Ở phần Head, YOLOv5 sử dụng thuật toán YOLO để thực hiện dự đoán bounding box và class



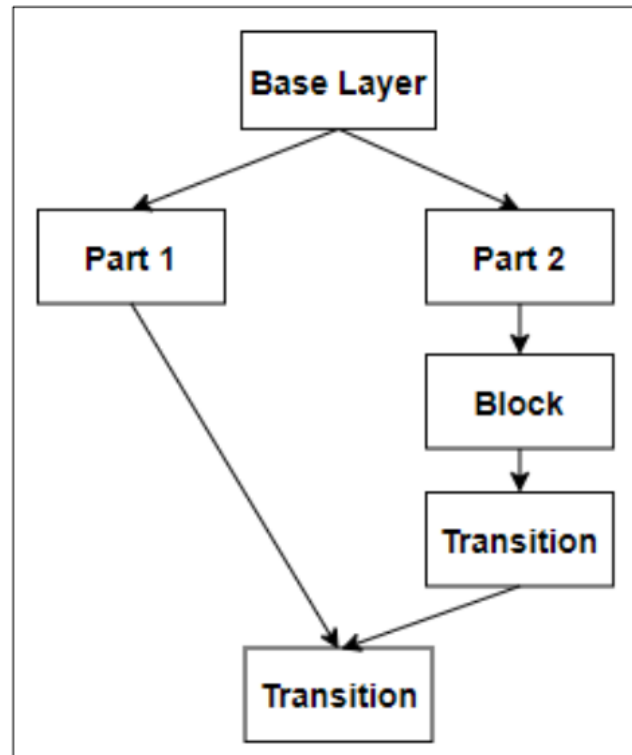
*Hình 4: Kiến trúc YOLOv5*

### 2.2.3. Backbone

Backbone cho một bộ phát hiện đối tượng là bộ phân lớp đã được huấn luyện trước trên tập dữ liệu ImageNet nhằm trích rút đặc trưng. Trong YOLOv5, phần Backbone sử dụng mạng CSP-Darknet53. CSP-Darknet53 là mạng nơ-ron tích chập và là xương sống cho phần phát hiện vật thể. Mạng này được xây dựng dựa trên CSPNet và Darknet53.



### ❖ CSPNet (Cross Stage Partial Network)



Hình 5: Kiến trúc CSPNet

Ý tưởng chính của CSPNet được thể hiện như Hình. Thay vì chỉ có một đường đi từ đầu tới cuối, CSPNet chia thành 2 đường đi. Nhờ việc chia làm 2 đường như vậy, ta sẽ loại bỏ được việc tính toán lại gradient (đạo hàm), nhờ đó có thể tăng tốc độ trong training. Hơn nữa, việc tách làm 2 nhánh, với mỗi nhánh là một phần được lấy từ feature map trước đó, nên số lượng tham số cũng giảm đi đáng kể, từ đó tăng tốc trong cả quá trình inference chứ không chỉ trong training.

### ❖ Darknet53

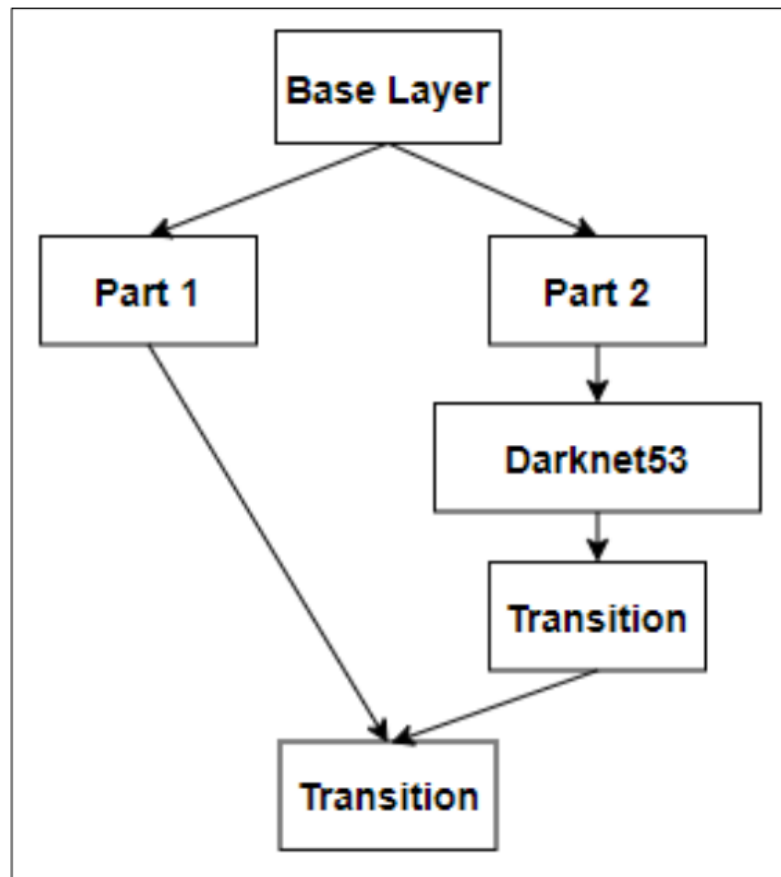
Darknet53 là mạng sử dụng trong trích rút đặc trưng, xây dựng dựa trên mạng CNN. Mạng này được áp dụng trong YOLOv3, đáp ứng mục đích phát hiện và nhận diện vật thể. Kiến trúc CSPDarknet53 sử dụng tổng cộng 52 lớp tích chập, 23 lớp residual

	Type	Filters	Size	Output
	Convolutional	32	$3 \times 3$	$256 \times 256$
	Convolutional	64	$3 \times 3 / 2$	$128 \times 128$
1x	Convolutional	32	$1 \times 1$	
	Convolutional	64	$3 \times 3$	
	Residual			$128 \times 128$
	Convolutional	128	$3 \times 3 / 2$	$64 \times 64$
2x	Convolutional	64	$1 \times 1$	
	Convolutional	128	$3 \times 3$	
	Residual			$64 \times 64$
	Convolutional	256	$3 \times 3 / 2$	$32 \times 32$
8x	Convolutional	128	$1 \times 1$	
	Convolutional	256	$3 \times 3$	
	Residual			$32 \times 32$
	Convolutional	512	$3 \times 3 / 2$	$16 \times 16$
8x	Convolutional	256	$1 \times 1$	
	Convolutional	512	$3 \times 3$	
	Residual			$16 \times 16$
	Convolutional	1024	$3 \times 3 / 2$	$8 \times 8$
4x	Convolutional	512	$1 \times 1$	
	Convolutional	1024	$3 \times 3$	
	Residual			$8 \times 8$
	Avgpool		Global	
	Connected		1000	
	Softmax			

Hình 6: Kiến trúc Darknet53

### ❖ CSP-Darknet53

Khi mạng CSP kết hợp với Darknet53, khối block của CSP sẽ được thay bằng Darknet53, tạo thành một mạng mới, giúp tăng cường khả năng học và xử lý của Darknet53 hơn



Hình 7: Kiến trúc CPP-Darknet53

#### 2.2.4. Neck

Sau khi thông qua backbone, dữ liệu được chuyển vào neck để tích hợp các đặc trưng từ các tầng khác nhau và tạo ra các đặc trưng tốt hơn cho việc phát hiện vật thể. Trong phần neck, YOLOv5 sử dụng khối SPPF và PAN

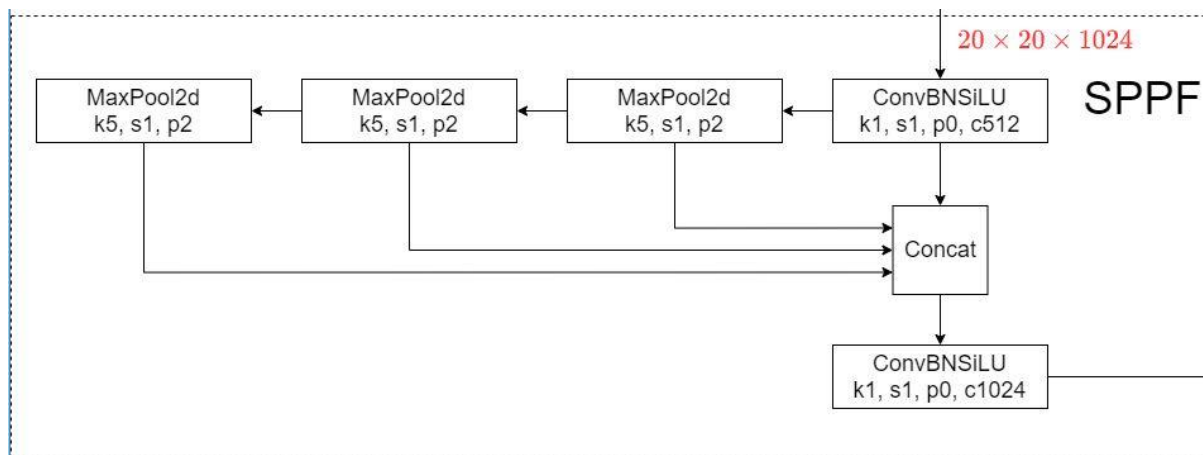
##### ❖ Spatial Pyramid Pooling – Fast (SPPF)

Trong YOLOv5, SPPF giúp đưa kích thước các bản đồ đặc trưng về cùng một kích thước cố định. Thông thường, với các mạng có lớp Fully-Connected, kích thước ảnh đầu vào cần phải được giữ nguyên. Tuy nhiên, với SPPF, ảnh đầu vào đã có thể có kích thước đa dạng hơn mà không cần phải giữ nguyên nữa, một vector có độ dài cố định vẫn được sinh ra khi đi qua SPPF.

SPPF được áp dụng sau các lớp trích xuất đặc trưng của mạng nơ-ron tích chập (CNN). Nó thực hiện việc tổng hợp thông tin đặc trưng từ các kích thước không đổi nhưng không gian khác nhau trong ảnh. Cụ thể, SPPF chia nhỏ đầu vào thành các ô lưới với các kích thước khác nhau và thực hiện max pooling trên từng ô lưới. Quá trình này tạo ra các vector đặc trưng có kích thước cố định cho mỗi ô lưới.

Sau khi thực hiện max pooling trên các ô lưới với các kích thước khác nhau, các vector đặc trưng được thu thập lại và kết hợp thành một vector đặc trưng duy nhất bằng cách sử dụng các phép gộp (merge) và một lớp Fully Connected (FC). Quá trình này giúp tổng hợp thông tin từ các kích thước không đổi nhưng không gian khác nhau trong ảnh và tạo ra một vector đặc trưng tổng quát cho toàn bộ ảnh.

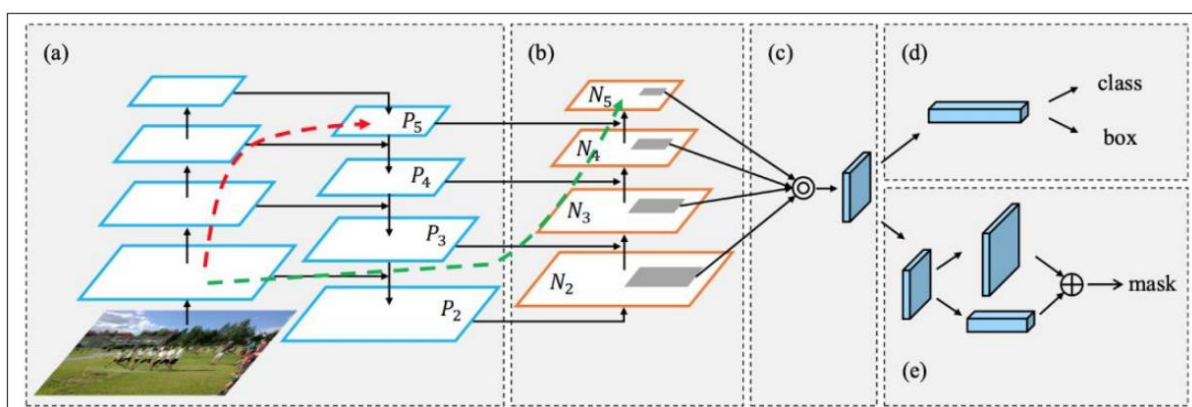
Kiến trúc của SPPF được thể hiện ở hình dưới:



Hình 8: Kiến trúc SPPF

#### ❖ Path Aggregation Network (PAN)

Mạng học sâu càng sâu thì càng làm mất mát thông tin do khi hình ảnh đi qua các lớp của mạng nơ-ron, độ phức tạp của đặc trưng tăng, độ phân giải hình ảnh giảm. Vì vậy, sẽ rất khó để phát hiện được vật thể kích thước nhỏ. Trong YOLOv5, PANet được sử dụng để cải thiện khả năng phát hiện vật thể của mô hình và giải quyết vấn đề mất mát thông tin trong quá trình truyền ngược (back propagation) của mạng



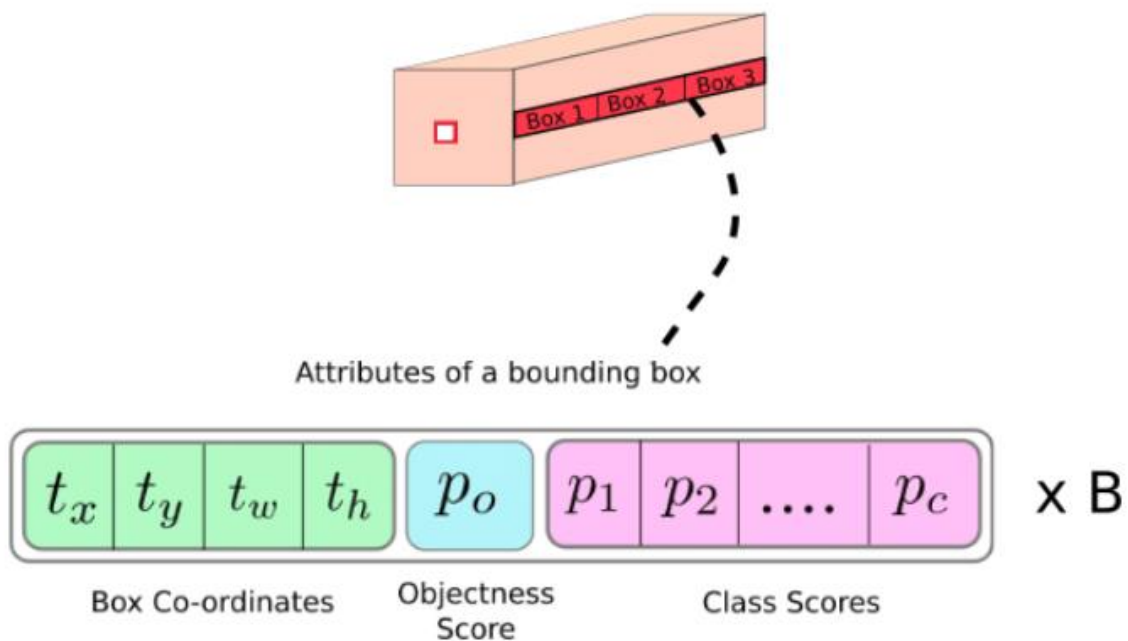
Hình 9: Kiến trúc PAN

PANet được sử dụng để tăng cường kết nối giữa các tầng trong mạng neural để chia sẻ thông tin và đặc trưng từ các tầng khác nhau. Nó bao gồm hai thành phần chính: PANet-topdown (a) và PANet-bottomup (b).

- **PANet-topdown**: Thành phần này tạo ra các kết nối từ các tầng ở phía sau lên các tầng phía trước. Nó sử dụng các phép gộp (merge) và upsample để chia sẻ thông tin đặc trưng từ các tầng sau đến các tầng trước. Quá trình này giúp cải thiện việc truyền ngược và chia sẻ thông tin giữa các tầng trong mạng.
- **PANet-bottomup**: Thành phần này tạo ra các kết nối từ các tầng ở phía trước xuống các tầng phía sau. Nó sử dụng các phép gộp (merge) để kết hợp thông tin từ các tầng trước và cung cấp thông tin bổ sung cho các tầng sau. Quá trình này giúp cải thiện khả năng phát hiện vật thể và đảm bảo rằng các đặc trưng từ các tầng trước được truyền đến các tầng sau một cách tốt nhất

### 2.2.5. Head

Ở phần Head, để thực hiện dự đoán bounding box và class, YOLOv5 được thực hiện giống YOLOv3 và YOLOv4. Nó bao gồm ba lớp tích chập dự đoán vị trí của các bounding box  $(t_x, t_y, t_w, t_h)$ , điểm số  $p_o$  và các lớp  $(p_1, p_2, \dots, p_c)$ .



Hình 10: Output của YOLOv5

Trong đó,  $(t_x, t_y, t_w, t_h)$  không phải giá trị thực của bounding box, mà chỉ là giá trị offset của bounding box so với 1 anchor box cho trước. Anchor box này có kích thước  $(p_w, p_h)$  được định nghĩa sẵn.

Từ  $(t_x, t_y, t_w, t_h)$  ta cần bước biến đổi để tính ra giá trị tọa độ thật của bounding box  $(b_x, b_y, b_w, b_h)$  dựa trên  $(p_w, p_h)$  theo công thức dưới đây:

$$\begin{aligned}
b_x &= 2\sigma(t_x) - 0.5 + c_x \\
b_y &= 2\sigma(t_y) - 0.5 + c_y \\
b_w &= p_w (2\sigma(t_w))^2 \\
b_h &= p_h (2\sigma(t_h))^2
\end{aligned}$$

Hình 11: Công thức tính tọa độ bounding box

Trong đó,  $\sigma$  là hàm sigmoid, dùng để ép miền giá trị về khoảng (0, 1).

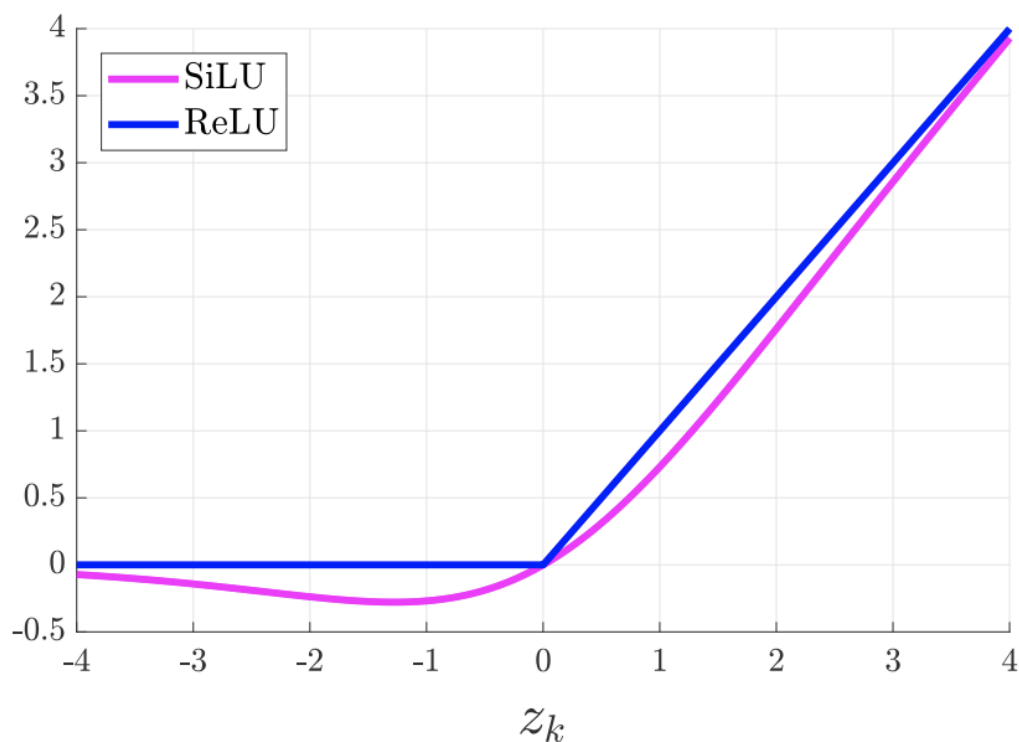
#### 2.2.6. Hàm kích hoạt (Activation function)

YOLOv5 sử dụng hàm kích hoạt sigmoid linear units (SiLU), còn được gọi là hàm Swish, là một hàm kích hoạt trơn, liên tục, tự điều chỉnh và phi tuyến tính được định nghĩa với công thức:

$$f(x) = x\sigma(x)$$

Trong đó:  $\sigma(x) = \frac{1}{1+e^{-x}}$  là hàm sigmoid thông thường

Hàm SiLU tự động điều chỉnh độ nhảy của đầu ra dựa trên giá trị đầu vào. Khi đầu vào  $x$  gần với 0, hàm SiLU tương tự như hàm tuyến tính, tăng khả năng truyền dẫn gradient hiệu quả. Khi đầu vào  $x$  xa khỏi 0, hàm SiLU xấp xỉ hàm sigmoid, giới hạn đầu ra trong khoảng (0, 1) và giúp kiểm soát giá trị lớn của đầu ra. Hàm SiLU có khả năng tính toán hiệu quả hơn hàm ReLU do chỉ sử dụng một lượng nhỏ các phép tính hàm số và đạo hàm.



Hình 12: Hàm SiLU và hàm ReLU

### 2.2.7. Loss function

Thay vì sử dụng một loss function duy nhất, YOLOv5 sử dụng compound loss để tính toán tổng của các thành phần loss function khác nhau để đo lường sự sai khác toàn diện giữa giá trị dự đoán và giá trị thực tế. Compound loss tính toán dựa trên các thành phần bao gồm :

Classes loss (BCE loss) : dùng để đo lường sự khác biệt trong việc phân loại đối tượng. YOLOv5 sử dụng hàm Binary Cross-Entropy (BCE loss) để tính toán sai số giữa xác suất dự đoán và xác suất thực tế của các lớp đối tượng.

Objectness loss (BCE loss) : đánh giá sự khác biệt về sự hiện diện của đối tượng. YOLOv5 cũng sử dụng BCE loss để tính toán sai số giữa score dự đoán và score thực tế về sự hiện diện của đối tượng.

Location loss (CIoU loss): đo lường sự sai khác trong việc xác định vị trí của bounding box. Nó sử dụng CIoU loss (một biến thể của IoU loss) để tính toán sai số về tọa độ và kích thước của bounding box dự đoán và bounding box thực tế.

Ba thành phần này cùng đóng góp vào compound loss của YOLOv5 và được sử dụng để tối ưu hóa mô hình trong huấn luyện.

$$Loss = \lambda_1 L_{cls} + \lambda_2 L_{obj} + \lambda_3 L_{loc}$$

### 2.2.8. Loss scalling

YOLOv5 có sử dụng 3 đầu ra từ PAN Neck, để phát hiện objects tại 3 scale khác nhau. Tuy nhiên, bởi sự ảnh hưởng của các object tại mỗi scale đến Objectness Loss là khác nhau, do đó, công thức của Objectness Loss được thay đổi như sau:

$$L_{obj} = 4.0 \cdot L_{obj}^{small} + 1.0 \cdot L_{obj}^{medium} + 0.4 \cdot L_{obj}^{large}$$

### 2.2.9. Hàm tối ưu hóa (Optimization function)

YOLOv5 sử dụng hàm tối ưu hóa là Stochastic Gradient Descent (SGD). SGD là một biến thể của thuật toán Gradient Descent được sử dụng để tối ưu hóa các mô hình máy học. Nó giải quyết sự kém hiệu quả trong tính toán của các phương pháp Gradient Descent truyền thống khi xử lý các tập dữ liệu lớn trong các mô hình máy học.

Trong SGD, thay vì sử dụng toàn bộ tập dữ liệu cho mỗi lần lặp, chỉ một mẫu ngẫu nhiên duy nhất (hoặc một lô nhỏ) được chọn để tính toán độ dốc và cập nhật các tham số mô hình. Lựa chọn ngẫu nhiên này đưa tính ngẫu nhiên vào quá trình tối ưu hóa, đồng thời chi phí tính toán cho mỗi lần lặp giảm đáng kể so với các phương pháp Gradient Descent truyền thống yêu cầu xử lý toàn bộ tập dữ liệu.

Công thức cập nhật trọng số trong SGD là:

$$\theta_{t+1} = \theta_t - \alpha * \nabla J(\theta_t)$$

Trong đó:

- $\theta_{t+1}$  là vector tham số cập nhật tại thời điểm t+1
- $\theta_t$  là vector tham số hiện tại tại thời điểm t
- $\alpha$  là learning rate (tốc độ học)
- $\nabla J(\theta_t)$  là gradient của hàm mất mát  $J(\theta)$  theo các tham số  $\theta$  tại thời điểm t

### 2.2.10. Batch normalization

Batch Normalization là một phương pháp hiệu quả khi huấn luyện một mô hình mạng nơron. Mục tiêu của phương pháp này chính là việc muốn chuẩn hóa các đặc trưng (đầu ra của mỗi layer sau khi đi qua các hàm kích hoạt) có trung bình gần bằng 0 và độ lệch chuẩn gần bằng 1. Sau khi chuẩn hóa, giá trị được nhân với một tham số tỷ lệ (scale) và cộng với một tham số dịch chuyển (shift). Hai tham số này cho phép mô hình học



cách điều chỉnh lại đầu ra chuẩn hóa để phù hợp với yêu cầu của bài toán. Trong YOLOv5, sau khi các đặc trưng qua hàm SiLU sẽ được chuẩn hóa

Batch Normalization giúp cải thiện tốc độ học và ổn định quá trình huấn luyện bằng cách giảm vấn đề biến mất độ dốc (vanishing gradient) và khởi động ban đầu (initialization). Nó cũng có khả năng chống overfitting và giúp mô hình tự học được các đặc trưng quan trọng hơn.

#### **2.2.10. Anchor box**

Anchor box được sử dụng để giúp mô hình dự đoán bounding box của đối tượng trong ảnh. YOLOv5 sử dụng Auto Anchor, một kỹ thuật tự động tìm kiếm các giá trị Anchor Box tối ưu cho mô trong quá trình huấn luyện.

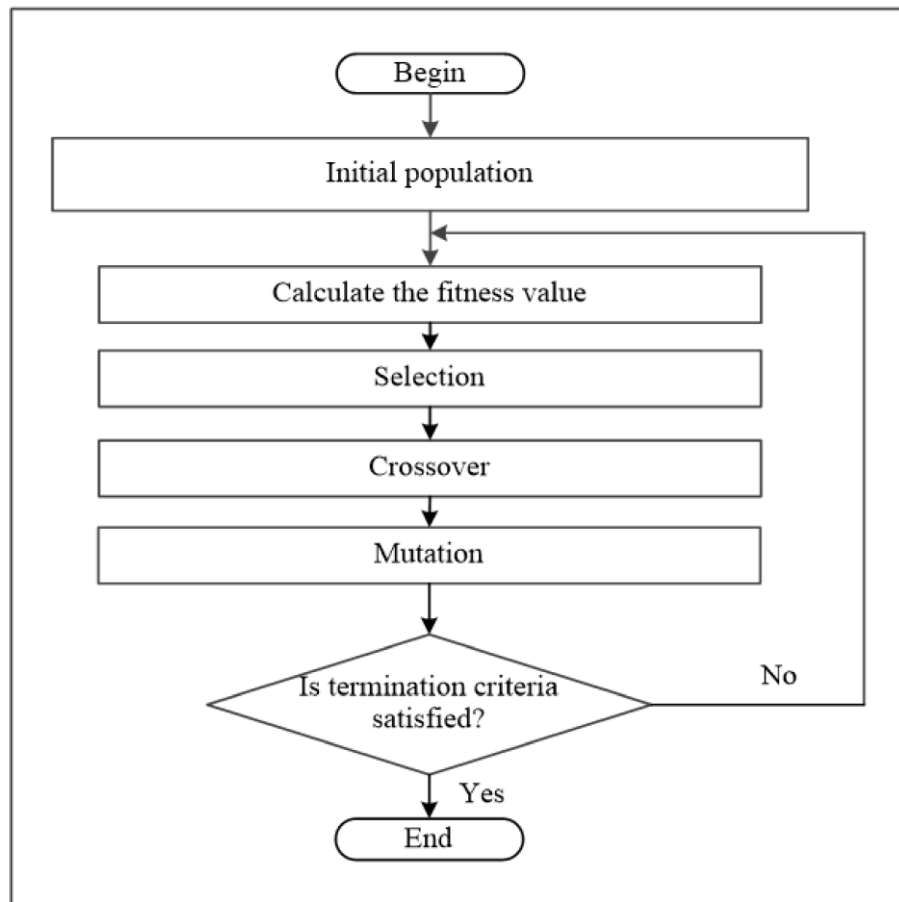
Quá trình Auto Anchor gồm các bước sau:

- Chuẩn bị dữ liệu huấn luyện: Các bounding box của các đối tượng trong tập huấn luyện được cung cấp. Các bounding box này chứa thông tin về tọa độ (x, y) của góc trái trên cùng và (w, h) của chiều rộng và chiều cao của đối tượng.
- Tính toán trung bình kích thước: Từ tập huấn luyện, trung bình kích thước của các bounding box được tính toán.
- K-means clustering: Áp dụng thuật toán K-means clustering trên tập huấn luyện để phân cụm các bounding box thành các nhóm dựa trên tương đồng về kích thước. Số lượng cluster (nhóm) thường được chọn trước.
- Xác định anchor box: Dựa trên các centroid (trung tâm) của các cluster thu được từ bước trước, các anchor box cuối cùng được xác định. Các anchor box được xác định bằng cách tính toán chiều rộng và chiều cao dựa trên kích thước trung bình và tỉ lệ được cố định.

#### **2.2.11. Tiến hóa siêu tham số (Hyperparameters Evolution)**

Siêu tham số (hyperparameters) là các tham số không được học trong quá trình huấn luyện mô hình, nhưng có ảnh hưởng đến hiệu suất và học tập của mô hình. Cách cấu hình và lựa chọn các siêu tham số có thể ảnh hưởng đến tốc độ hội tụ, độ chính xác, khả năng tổng quát hóa và sự ổn định của mô hình. Tuy nhiên, việc tìm kiếm các siêu tham số tối ưu có thể là một thách thức. Các phương pháp truyền thống như grid search có thể khó sử dụng vì ba nguyên nhân. Thứ nhất, không gian tìm kiếm lớn. Thứ hai,

không biết được sự tương quan giữa các chiều. Thứ ba, chi phí xác định điểm phù hợp tại mỗi điểm cao. Điều này làm cho giải thuật di truyền trở thành ứng cử viên phù hợp cho việc tìm kiếm hyperparameter.



Hình 13: Giải thuật di truyền

Giải thuật di truyền gồm 5 giai đoạn (phase) chính: khởi tạo quần thể ban đầu (initial population), tính điểm phù hợp (fitness score), chọn lọc (selection), trao đổi chéo (cross-over), đột biến (mutation). Giải thuật kết thúc khi tìm thấy solution đủ tốt hoặc đến một thế hệ nhất định.

- **Quần thể ban đầu**

```

lr0: 0.01 # initial learning rate (SGD=1E-2, Adam=1E-3)
lrf: 0.2 # final OneCycleLR learning rate (lr0 * lrf)
momentum: 0.937 # SGD momentum/Adam beta1
weight_decay: 0.0005 # optimizer weight decay 5e-4
warmup_epochs: 3.0 # warmup epochs (fractions ok)
warmup_momentum: 0.8 # warmup initial momentum
warmup_bias_lr: 0.1 # warmup initial bias lr
box: 0.05 # box loss gain
cls: 0.5 # cls loss gain
cls_pw: 1.0 # cls BCELoss positive_weight
obj: 1.0 # obj loss gain (scale with pixels)
obj_pw: 1.0 # obj BCELoss positive_weight
iou_t: 0.20 # IoU training threshold
anchor_t: 4.0 # anchor-multiple threshold
# anchors: 3 # anchors per output layer (0 to ignore)
fl_gamma: 0.0 # focal loss gamma (efficientDet default gamma=1.5)
hsv_h: 0.015 # image HSV-Hue augmentation (fraction)
hsv_s: 0.7 # image HSV-Saturation augmentation (fraction)
hsv_v: 0.4 # image HSV-Value augmentation (fraction)
degrees: 0.0 # image rotation (+/- deg)
translate: 0.1 # image translation (+/- fraction)
scale: 0.5 # image scale (+/- gain)
shear: 0.0 # image shear (+/- deg)
perspective: 0.0 # image perspective (+/- fraction), range 0-0.001
flipud: 0.0 # image flip up-down (probability)
fliplr: 0.5 # image flip left-right (probability)
mosaic: 1.0 # image mosaic (probability)
mixup: 0.0 # image mixup (probability)
copy_paste: 0.0 # segment copy-paste (probability)

```

Hình 14: Hyperparameters của YOLOv5

Quần thể ban đầu chỉ có 1 cá thể. Nếu ngay từ đầu cá thể được chọn đã tốt (điểm phù hợp cao) thì sẽ có khả năng tìm ra các cá thể tốt hơn. Do đó các gen của cá thể này được tác giả lựa chọn thay vì sinh ngẫu nhiên.

- **Điểm phù hợp**

Điểm phù hợp là mục tiêu cần tối ưu sao cho đạt giá trị lớn nhất. Trong YOLOv5 điểm phù hợp được định nghĩa bằng tổng của: 10% của mAP@0.5 và 90% của mAP@0.5:0.95

- **Tiến hóa**

Các toán tử di truyền chính là trao đổi chéo và đột biến. Trong đó, đột biến được sử dụng với xác suất 80% và phương sai 0.04 để tạo cá thể mới từ những cha mẹ tốt nhất từ tất cả các thế hệ trước

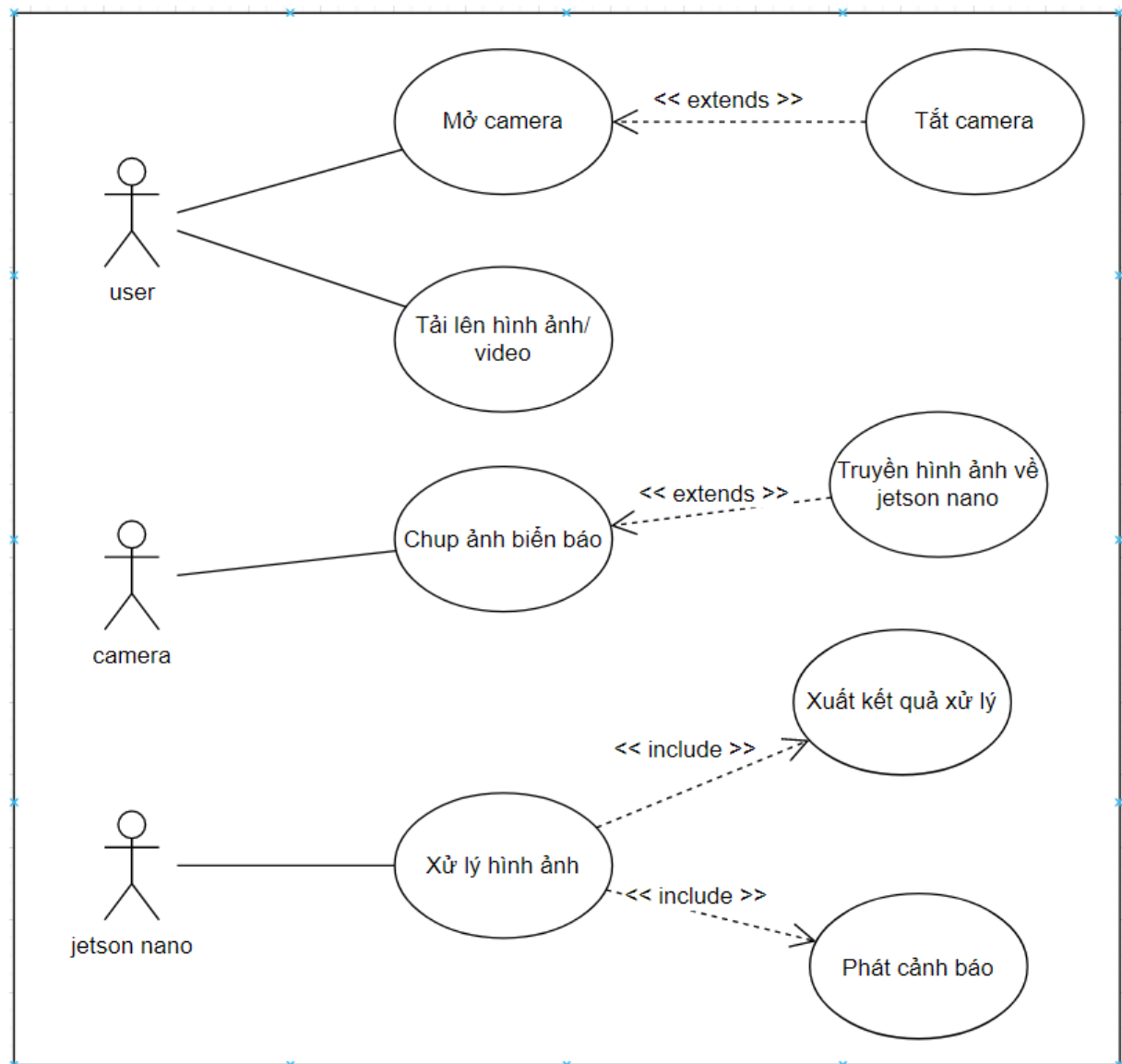
## 2.3. Giải pháp phần mềm

### 2.3.1. Phát biểu bài toán

Xây dựng ứng dụng desktop cho phép người dùng tải lên một hình ảnh hoặc sử dụng camera trên máy tính để phát hiện và định vị các biển báo trong ảnh.

- Đầu vào: Hình ảnh từ nguồn dữ liệu (tải lên từ máy tính hoặc từ camera).
- Đầu ra: Hình ảnh gốc với các biển báo được đánh dấu bằng hình chữ nhật và nhãn của các biển báo được phát hiện.
- Công nghệ sử dụng: Python Tkinter.
- Đối tượng người dùng: Người tham gia giao thông.

### 2.3.2. Sơ đồ usecase



Hình 15: Sơ đồ usecase hệ thống

## 3. KẾT QUẢ

### 3.1. Nhận diện biển báo

#### 3.1.1. Tập dữ liệu

##### a. Nguồn gốc và cách thức thu thập

Trong phần huấn luyện mô hình nhận diện biển báo giao thông, tập dữ liệu là các biển báo giao thông của hệ thống giao thông đường bộ Việt Nam.

Tập dữ liệu được thu thập từ 2 nguồn:


- Nguồn 1: Bộ dữ liệu có sẵn công khai.
- Nguồn 2: Dữ liệu tự thu thập bằng cách chụp hình các biển báo giao thông trong khu vực thành phố Đà Nẵng.

Dữ liệu sau đó được tổng hợp lên Roboflow và thực hiện gán nhãn.


## b. Thông tin tập dữ liệu

- Bộ dữ liệu trước khi chọn:
  - Nguồn: <https://universe.roboflow.com/quoc-nguyen-sgjwa/traffic-sign-model-9xhqf>
  - Tập dữ liệu gồm có 3245 hình ảnh biển báo giao thông.
  - Sau khi gán nhãn, bao gồm 5316 nhãn tương ứng với 30 class (30 biển báo).
- Bộ dữ liệu sau khi chọn:
  - Nguồn: <https://universe.roboflow.com/pbl-uj4wt/pbl5-6p2ne>
  - Lược bỏ bớt class (biển báo), tập trung vào các class có nhiều instance. Từ đó, tập trung bổ sung thêm dữ liệu cho các class đó.
  - Mô tả: Tập dữ liệu gồm có 3773 hình ảnh biển báo, trong đó 350 hình ảnh được sử dụng lại từ các bộ dữ liệu có sẵn công khai, 3423 hình ảnh còn lại được nhóm chúng em tự thu thập bằng cách chụp hình biển báo trên đường phố Đà Nẵng. Sau khi thu thập, dữ liệu sẽ được upload lên RoboFlow và thực hiện gán nhãn. Sau khi gán nhãn, tập dữ liệu gồm có 12 class tương ứng 12 biển báo thường gặp với 5366 nhãn.
  - Class:

*Bảng 4: Bảng các class trong dataset*

Class	Ý nghĩa	Số nhãn	Hình ảnh
131a	Cấm đỗ xe	1056	

102	Cấm đi ngược chiều	698	
106b	Cấm xe tải	456	
128	Cấm bóp còi	272	
130	Cấm dừng và đỗ	472	
123a	Cấm rẽ trái	361	
123b	Cấm rẽ phải	381	
127	Tốc độ tối đa cho phép (40 km/h)	364	
207b	Giao nhau với đường không ưu tiên	464	
225	Chú ý trẻ em	274	
245a	Đi chậm	221	

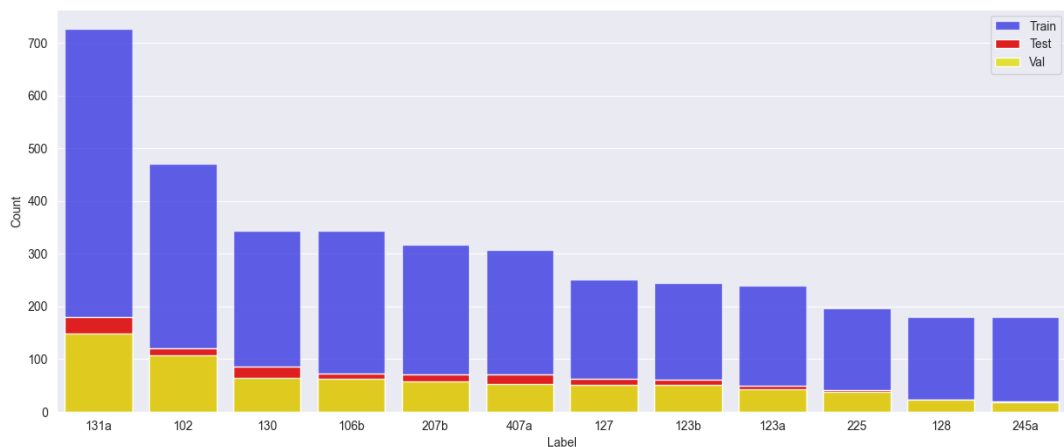
407a	Đường 1 chiều	347	
------	---------------	-----	---

### c. Tiền xử lý

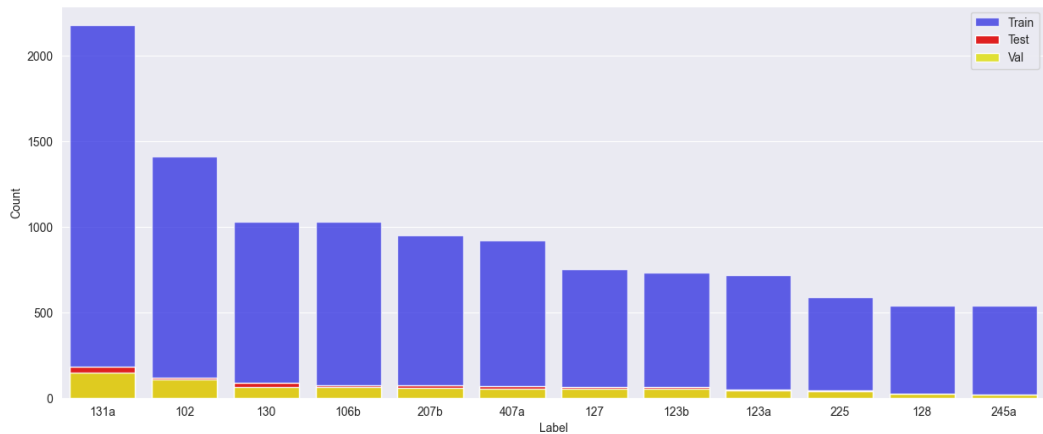
Đầu tiên dữ liệu sẽ được chuẩn hoá về cùng một kích thước 640x640. Sau đó, dữ liệu sẽ được chia thành 3 tập huấn luyện: 70%, đánh giá: 15%, kiểm thử: 15%. Để tăng tính đa dạng và khả năng tổng quát hoá của mô hình, nhóm đã áp dụng các kĩ thuật tăng cường dữ liệu trên tập huấn luyện:

- **Hue:** điều chỉnh màu sắc của hình ảnh. Giá trị được chỉ định là từ  $-25^\circ$  đến  $+25^\circ$ . Thay đổi màu sắc giúp tăng tính đa dạng của dữ liệu huấn luyện.
- **Brightness:** điều chỉnh độ sáng của hình ảnh. Giá trị được chỉ định là từ  $-25\%$  đến  $+25\%$
- **Blur:** áp dụng hiệu ứng mờ lên hình ảnh.

Kết quả là tập huấn luyện đã tăng từ 3773 hình ảnh lên 8977 hình ảnh, tỉ lệ sau cùng của tập dữ liệu là huấn luyện: 87 (%), đánh giá: 6 (%), kiểm thử: 7 (%)

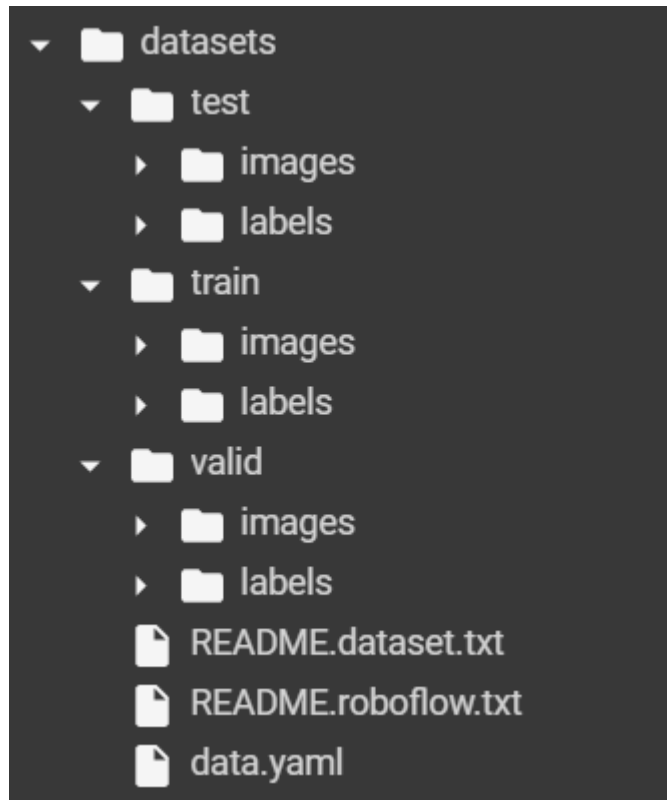


Hình 16: Phân bố dữ liệu trước khi tăng cường



Hình 17: Phân bố dữ liệu sau khi tăng cường

Sau khi tăng cường và phân chia tập dữ liệu, dữ liệu sẽ được chuyển đổi về định dạng dữ liệu của YOLOv5:



Hình 18: Cấu trúc thư mục dataset

### 3.1.2. Huấn luyện

Trong quá trình huấn luyện mô hình yolo, việc sử dụng GPU là cần thiết để cải thiện tốc độ xử lý và hiệu suất của mô hình. Vì thế nhóm sử dụng tài nguyên GPU từ Google Colab nhằm tiết kiệm chi phí và tăng tốc độ huấn luyện mô hình.

#### a. Tiến hóa siêu tham số

Tiếp theo, nhóm tiến hành tìm kiếm bộ siêu tham số tối ưu cho mô hình tốt nhất sử dụng thuật toán di truyền với 100 thế hệ, mỗi thế hệ 10 lần lặp:



```
lr0: 0.01061
lrf: 0.01
momentum: 0.9417
weight_decay: 0.00046
warmup_epochs: 3.0
warmup_momentum: 0.82301
warmup_bias_lr: 0.1
box: 0.04985
cls: 0.5
cls_pw: 0.99655
obj: 1.058
obj_pw: 1.0
iou_t: 0.2
anchor_t: 3.5781
fl_gamma: 0.0
hsv_h: 0.01465
hsv_s: 0.65497
hsv_v: 0.42314
degrees: 0.0
translate: 0.10011
scale: 0.5002
shear: 0.0
perspective: 0.0
flipud: 0.0
fliplr: 0.0
mosaic: 0.0
mixup: 0.0
copy_paste: 0.0
anchors: 3.0
```

*Hình 19: Hyperparameters tối ưu*

#### **b. Cấu hình huấn luyện**

- GPU: Tesla T4
- epochs: 50
- batch size: 16
- Pre-trained model: yolov5n/ yolov5s/yolov6/yolov7

#### **3.1.3. Các công thức đánh giá kết quả**


Để tiến hành đánh giá hiệu suất mô hình, nhóm sử dụng mAP (mean Average Precision). mAP là độ đo được sử dụng rất phổ biến cho các bài toán Object Detection. Công thức mAP dựa trên các chỉ số phụ sau đây

- Intersection Over Union( IOU)
- Confusion Matrix

- Precision
- Recall

### **IOU (Intersection over union)**

IOU là chỉ số đánh giá được sử dụng để đánh giá độ chính xác của phát hiện đối tượng dựa trên tập dữ liệu cụ thể. IOU được tính bằng:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


*Hình 20: Intersection Over Union (IOU)*

Trong đó Area of Overlap là diện tích phần giao nhau giữa predicted bounding box với growth-truth bounding box, còn Area of Union là diện tích phần hợp giữa predicted bounding box với growth-truth bounding box. Nếu IOU lớn hơn 1 ngưỡng (thường bằng 0.5) thì prediction được đánh giá là tốt.

### **Confusion Matrix**

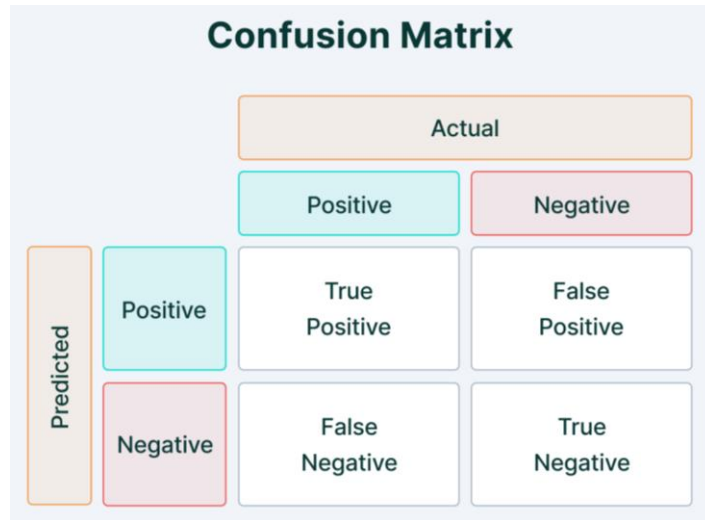
Để tạo một confusion matrix, ta cần 4 thuộc tính :

**True Positive (TP)** : Một kết quả phát hiện đúng (IOU  $\geq$  ngưỡng).

**False Positive (FP)** : Một kết quả phát hiện sai (IOU  $<$  ngưỡng).

**True Negative (TN)** : Không có biển báo và không phát hiện ra biển báo

**False Negative (FN)** : Có biển báo nhưng không phát hiện ra.



Hình 21: Confusion matrix

### Precision

Precision là độ đo đánh giá độ tin cậy của kết luận đưa ra (tỷ lệ nhận diện đúng vật thể trên tập đã đánh nhãn).

$$Precision = \frac{TP}{TP + FP}$$

**Recall:** Là độ đo đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (tỷ lệ nhận diện đúng vật thể trên tổng số vật thể mà mô hình nhận diện).

$$Recall = \frac{TP}{TP + FN}$$

### Average Precision (AP)

Cách khác để so sánh hiệu suất của các bộ phát hiện khuôn mặt là tính diện tích của đồ thị đường cong (UAC) của Precision x Recall curve. Đường cong AP thường là đường zigzag lên rồi xuống. Trong thực tế, AP là độ chính xác được tính trung bình trên tất cả các giá trị recall giữa 0 và 1

$$AP = \sum_{k=0}^{k=n-1} [Recalls(k) - Recalls(k+1)] * Precisions(k)$$

**$Recalls(n) = 0, Precisions(n) = 1$**   
 **$n = \text{Number of thresholds.}$**

Hình 22: Công thức tính AP

### Mean Average Precision (mAP)

mAP được tính bằng trung bình tổng các giá trị AP của các class:

$$mAP = \frac{1}{n} \sum_{k=1}^{k=n} AP_k$$

$AP_k = \text{the AP of class } k$   
 $n = \text{the number of classes}$

Hình 23: Công thức tính mAP

### 3.3. Kết quả mô hình

#### 3.3.1. Kết quả định tính

Kết quả mô hình nhận diện có độ chính xác khá tốt. Tuy nhiên, vẫn còn 1 số trường hợp biển báo không nhận diện được do khoảng cách xa và một số trường hợp biển báo nhận diện chưa chính xác (nhầm lẫn với biển báo khác hoặc nhầm lẫn với môi trường)



Hình 24: Kết quả nhận diện biển báo

#### 3.3.2. Kết quả định lượng

Bảng 5: Kết quả mô hình nhận diện biển báo trên tập test

Model	Param (M)	Image size	Precision	Recall	mAP@.5	mAP@.5:.95	FPS
YOLOv5n	1.775	640	86.6	77.5	85.8	68.5	8
YOLOv5s	7.042	640	92.9	79.7	88.8	73.3	6
YOLOv5n + hyperparameters evolution	1.775	640	91.0	79	87.1	70.7	8

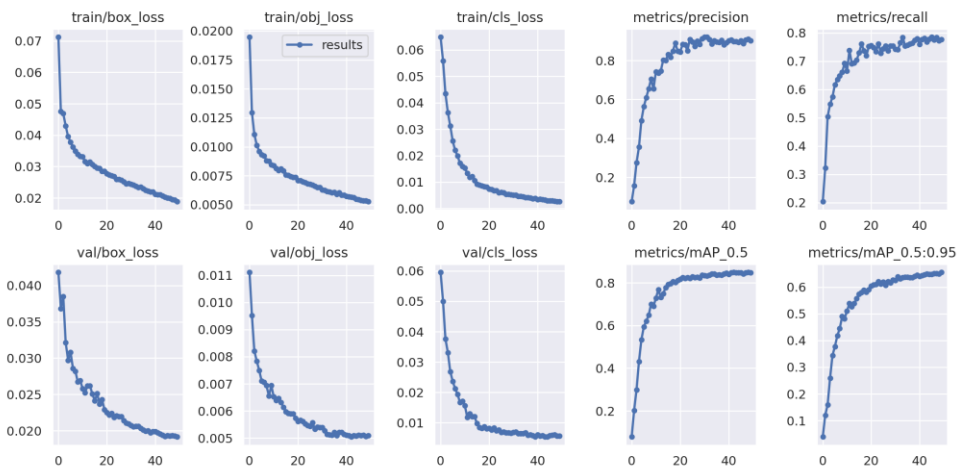
YOLOv6n	4.63	640	87.1	76.6	80.1	63.5	6
YOLOv6s	18.51	640	89.3	82	84.3	67	6
YOLOv7-tiny	6.037	640	89.7	75.7	79.2	61.1	6.5

Nhận xét:

- Ta thấy YOLOv5n sử dụng ít tham số nhất (1.775M). Điều này cho thấy YOLOv5n có cấu trúc nhỏ gọn hơn, tốn ít tài nguyên và thời gian huấn luyện hơn so với các mô hình khác.
- YOLOv5s có Precision cao nhất (92.9%) với cả 3 chỉ số Precision, Recall, Mean Average Precision đều cao, chứng tỏ YOLOv5s có khả năng dự đoán chính xác các đối tượng hơn, tuy nhiên tốc độ nhận dạng của YOLOv5s lại thấp hơn so với YOLOv5n.
- Áp dụng thuật toán di truyền tìm kiếm bộ siêu tham số tối ưu cho YOLOv5n giúp tăng độ chính xác của mô hình.

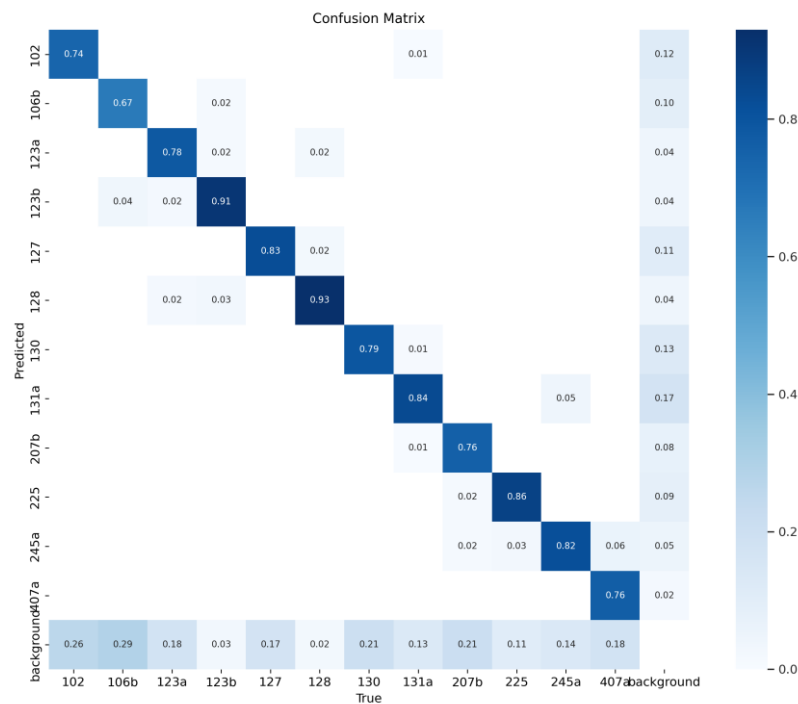
Trong quá trình thực nghiệm, nhóm nhận thấy rằng, sử dụng YOLOv5n+hyp\_evolve là cân bằng nhất về độ chính xác cũng như tốc độ nhận diện biến báo và có kích thước nhỏ phù hợp để triển khai trên jetson.

### 3.3.3. Kết quả huấn luyện của yolov5n + hyp\_evolve



Hình 25: Kết quả nhận diện biến báo

Độ chính xác và giá trị hàm mất mát của mô hình là khá tốt trong quá trình huấn luyện, có thể thấy việc mô hình hội tụ nhanh từ 0 tới 20 epoch



### 3.4. Kết quả thực nghiệm

Thống kê khoảng cách thực tế tối đa của hệ thống trong việc nhận diện đúng biển báo giao thông vào các buổi (sáng, chiều, tối).

Bảng 6: Thống kê khoảng cách nhận dạng

Class	Thời điểm	Lần 1	Lần 2	Lần 3	Trung bình
131a	Sáng	22	21.3	22.5	21.9
	Chiều	22.5	23	21.2	22.2
	Tối	19.6	18.5	18.3	18.8
102	Sáng	30.3	32.5	31	31.3
	Chiều	22.5	24.2	23.6	23.4
	Tối	20.1	19.5	20	19.9
106b	Sáng	17.2	18.1	17.5	17.6
	Chiều	22.6	21	20.7	21.4
	Tối	17.3	16.5	17	16.9
128	Sáng	26.3	25.6	26	26
	Chiều	21.8	22.3	22	22
	Tối	20.4	19.6	19.5	19.8
	Sáng	20.5	21.1	20.7	20.8

<b>130</b>	Chiều	16.2	17.6	16.5	<i>16.7</i>
	Tối	16.5	15.7	15.3	<i>15.8</i>
<b>123a</b>	Sáng	22.5	22.6	22	<i>22.4</i>
	Chiều	20.4	21	20.8	<i>20.7</i>
	Tối	16.8	17.5	17.3	<i>17.2</i>
<b>123b</b>	Sáng	27.5	27.7	27.2	<i>27.4</i>
	Chiều	22.6	23.3	23	<i>23</i>
	Tối	21	21.2	20.7	<i>21</i>
<b>127</b>	Sáng	27.6	28.1	28.2	<i>28</i>
	Chiều	26.7	26.5	26.1	<i>26.4</i>
	Tối	21.7	21.5	21	<i>21.4</i>
<b>207b</b>	Sáng	20.2	20.7	21	<i>20.6</i>
	Chiều	17.6	18.2	17	<i>17.6</i>
	Tối	16.5	16.7	17	<i>16.7</i>
<b>225</b>	Sáng	28.5	29	28.7	<i>28.7</i>
	Chiều	26.7	27	26.1	<i>26.6</i>
	Tối	22	21.5	20.6	<i>21.4</i>
<b>245a</b>	Sáng	26.8	26.5	26	<i>26.4</i>
	Chiều	22.5	23	22.4	<i>22.6</i>
	Tối	20.3	20.4	20	<i>20.2</i>
<b>407a</b>	Sáng	21.4	21.7	22	<i>21.7</i>
	Chiều	20.2	19.5	19.6	<i>19.8</i>
	Tối	16.8	17.2	17	<i>17</i>

(Đơn vị: mét)

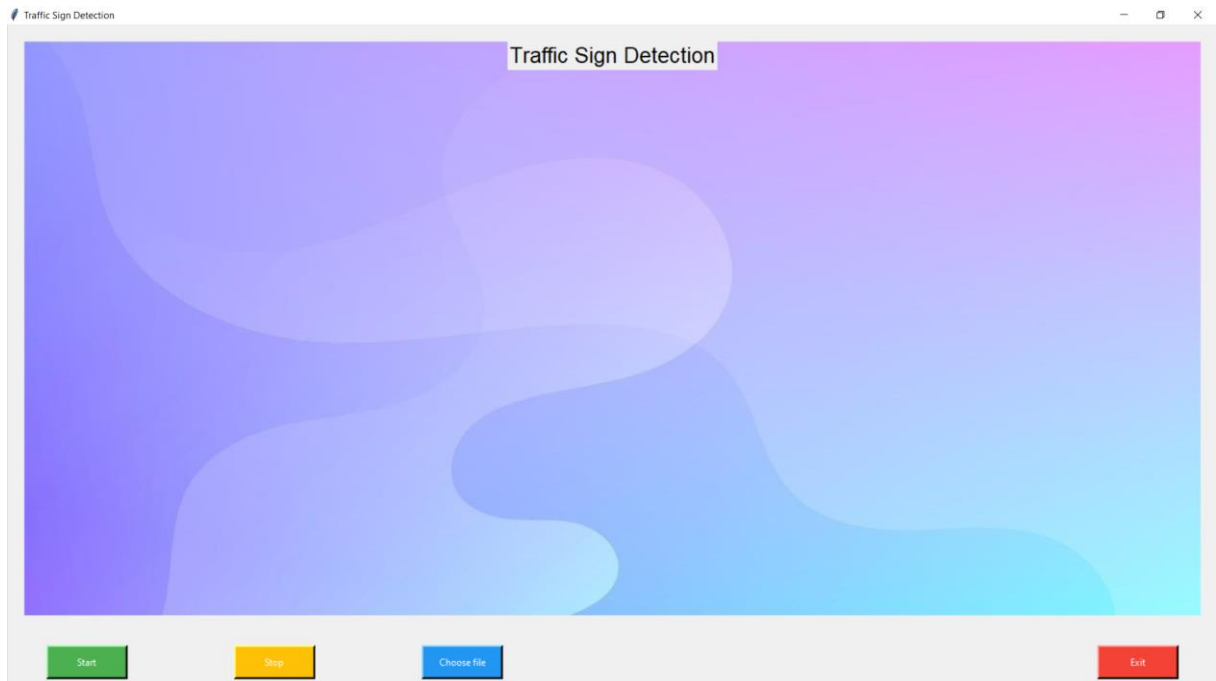
○ Nhận xét:

- Hầu hết các biển báo được nhận diện với khoảng cách tối đa theo sáng > râm > tối (trừ biển báo 106b râm > sáng).

Lý do có sự khác nhau giữa các buổi là tập dữ liệu của các biển báo buổi sáng nhiều hơn so với buổi chiều và tối.

- Các biển báo 102, 127, 123a, 123b nhận diện với khoảng cách xa. Lý do là có tập dữ liệu lớn.
- Các biển báo 245a, 225, 128 nhận diện với khoảng cách xa dù tập dữ liệu không lớn. Lý do là vì dữ liệu đa dạng và các biển báo này có hình dáng khác so với các biển báo còn lại.
- Biển báo 130 và 131a đều là hai biển báo có tập dữ liệu lớn nhưng khoảng cách tối đa nhận diện hai biển báo lại không lớn. Lý do là hai biển này khá tương đồng dẫn đến nhầm lẫn nhiều trong lúc nhận dạng.
- Khoảng cách tối đa nhận diện các biển báo vào sáng [18, 31] (m).
- Khoảng cách tối đa nhận diện các biển báo vào chiều [16, 27] (m).
- Khoảng cách tối đa nhận diện các biển báo vào tối [15, 21] (m).
  - ⇒ Vào buổi sáng hoặc chiều, người tham gia giao thông với tốc độ từ 25 đến 40 km/h thì có thể nhận diện đa số chính xác biển báo.
  - ⇒ Vào buổi tối, người tham gia giao thông với tốc độ từ 20 đến 30 km/h thì có thể nhận diện đa số chính xác biển báo.

### 3.4. Kết quả phần mềm



Hình 26: Giao diện ứng dụng desktop





Hình 27: Giao diện nhận diện biển báo

## 4. KẾT LUẬN

### 4.1. Đánh giá

#### 4.1.1. Kết quả đạt được

- Tập dữ liệu: Thu thập được 12 class với dữ liệu đa dạng (vị trí, kích thước, sáng tối, đơn lẻ, hỗn hợp,...)
- Mô hình: Áp dụng được mô hình YOLO vào bài toán thực hiện nhận diện biển báo giao thông.
- Phần mềm: Đủ các chức năng cơ bản để người dùng có thể thao tác sử dụng để nhận diện biển báo và quan sát sự nhận diện đó.

#### 4.1.2. Hạn chế

- Phần cứng: Thời gian 1 lần sử dụng liên tục chỉ có 1 tiếng không được nhiều. Sử dụng xong là phải sạc nguồn lại cho phần cứng khá lâu. Vì vậy, không phù hợp với quãng đường xa với thời gian di chuyển lớn.
- Mô hình: Mô hình nhận diện hiện tại chưa hoàn thiện. Tốc độ thực nghiệm chưa được tốt (fps khá thấp).
- Phần mềm: chỉ dừng mức cơ bản, đủ dùng; chưa có nhiều chức năng mở rộng.

## 4.2. Hướng phát triển

- Tập dữ liệu: Thu thập nhiều biển báo (class) hơn với nhiều kiểu dữ liệu đa dạng.
- Phần cứng: Sử dụng camera có độ phân giải tốt hơn để có những bức hình ở độ phân giải cao. Nâng cấp phần cứng của bộ vi xử lý, cải thiện tốc độ nhận diện biển báo.
- Mô hình: Tiếp tục huấn luyện mô hình để nâng cao độ chính xác nhận diện. Ngoài ra, tìm hiểu và xây dựng nhiều mô hình khác để tìm ra mô hình thích hợp với bài toán.
- App: Xây dựng app có giao diện đẹp hơn. Đồng thời, mở rộng ra nhiều chức năng phụ (đánh giá, học về các loại biển báo,..)

Hiện nay, hệ thống chỉ mới là hệ thống nhận diện biển báo giao thông đơn thuần. Sau này, có thể mở rộng tích hợp các ứng dụng có liên quan đến map, đường đi. Hơn nữa, phát triển hệ thống không chỉ nhận diện biển báo giao thông ở Việt Nam mà còn cả nước ngoài.

## 5. DANH MỤC TÀI LIỆU THAM KHẢO

- [1] J. Redmon, S. Divvala, R. Girshick & A. Farhadi “You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition”, 2016
- [2] D. Snegireva, A. Perkova “Traffic Sign Recognition Application Using YOLOv5 Architecture”, 2021
- [3] Glenn Jocher, “YOLOv5 Ultralytics Docs” , 2023
- [4] C. Wang, A. Bochkovskiy, H. Liao, “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”, 2022
- [5] Nguyen Tung Thanh, “Giải thuật di truyền và ứng dụng trong YOLOv5”, 2021
- [6] S. Liu, L. Qi, H. Qin, J. Shi, J. Jia, “Path Aggregation Network for Instance Segmentation”, 2018.
- [7] K. He, X. Zhang, S. Ren, J., “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”, 2015.

[8] C. Wang, H. Liao, I. Yeh, Y. Wu, P. Chen, J. Hsieh, “CSPNet: A new backbone that can enhance learning capability of CNN”, 2019.