

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

**KHÓA LUẬN TỐT NGHIỆP**

**XÁC ĐỊNH ĐỐI TƯỢNG VÀ TÍNH  
KÍCH THƯỚC CỦA ĐỐI TƯỢNG TRÊN  
BĂNG CHUYỀN**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Lã Nhu Hải

SINH VIÊN THỰC HIỆN: Trà Nguyễn Gia Khánh – 21DH110837

Nguyễn Viết Lượng – 21DH113863

TP. HỒ CHÍ MINH – THÁNG 08 – NĂM 2024

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH

KHOA CÔNG NGHỆ THÔNG TIN

**KHÓA LUẬN TỐT NGHIỆP**

**XÁC ĐỊNH ĐỐI TƯỢNG VÀ TÍNH  
KÍCH THƯỚC CỦA ĐỐI TƯỢNG TRÊN  
BĂNG CHUYỀN**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Lã Như Hải

SINH VIÊN THỰC HIỆN: Trà Nguyễn Gia Khánh – 21DH110837

Nguyễn Viết Lượng – 21DH113863

TP. HỒ CHÍ MINH – THÁNG 08 – NĂM 2024

## LỜI CẢM ƠN

Trước hết nhóm chúng em xin gửi lời cảm ơn và tri ân sâu sắc và chân thành đến quý thầy cô trong khoa Công Nghệ Thông Tin, Trường Đại học Ngoại ngữ - Tin học TP. Hồ Chí Minh (HUFLIT). Nhờ sự hỗ trợ, hướng dẫn nhiệt tình và nguồn tri thức vô giá từ quý thầy cô chúng em mới có thể hoàn thành khóa luận tốt nghiệp này.

Đặc biệt chúng em xin bày tỏ lòng biết ơn sâu sắc đến thầy ThS.Lã Như Hải người hướng dẫn tận tình của nhóm em trong bài khóa luận tốt nghiệp này và thầy ThS.Tôn Quang Toại người đã cho nhóm em sự giúp đỡ quý báu về kiến thức và bài học và những lần giải đáp những thắc mắc của nhóm em trong quá trình thực hiện khóa luận. Hai thầy không chỉ tận tình chỉ dẫn, truyền đạt kiến thức và kinh nghiệm quý báu, mà còn luôn động viên, khích lệ chúng em vượt qua mọi khó khăn trong quá trình nghiên cứu và hoàn thiện đề tài khóa luận. Sự quan tâm và hỗ trợ từ hai thầy đã giúp chúng em tự tin và kiên trì theo đuổi mục tiêu của mình.

Cuối cùng chúng em muốn gửi lời cảm ơn chân thành đến gia đình và bạn bè, những người luôn bên cạnh và động viên chúng em trong suốt quá trình học tập và nghiên cứu. Sự đồng hành và ủng hộ của mọi người là nguồn động lực vô giá. Giúp chúng em vượt qua mọi khó khăn và thử thách.

Một lần nữa, chúng em xin chân thành cảm ơn!



## LỜI CAM ĐOAN

Chúng em xin cam đoan rằng đề tài "Xác Định Đối Tượng Và Tính Kích Thước Của Đối Tượng Trên Băng Chuyền" là một công trình nghiên cứu độc lập do nhóm chúng em thực hiện dưới sự hướng dẫn tận tình của thầy ThS. Lã Như Hải và sự giúp đỡ quý báu của thầy ThS. Tôn Quang Toại. Chúng em khẳng định rằng không có bất kỳ sự sao chép nào từ các công trình nghiên cứu khác.

Nội dung khóa luận và các kết quả nghiên cứu là thành quả của sự nỗ lực và công sức của nhóm. Mọi số liệu và kết quả trình bày trong khóa luận đều được thu thập và sử dụng một cách trung thực, minh bạch, với nguồn gốc rõ ràng và tài liệu tham khảo được trích dẫn đầy đủ.

Chúng em xin chịu hoàn toàn trách nhiệm về tính chính xác và trung thực của nội dung khóa luận, đồng thời cam kết sẵn sàng chấp nhận mọi hình thức kỷ luật của khoa và nhà trường nếu có bất kỳ vấn đề nào phát sinh liên quan đến tính minh bạch và đạo đức học thuật.



## TÓM TẮT KHÓA LUẬN

Trong lĩnh vực sản xuất công nghiệp việc tích hợp tự động hóa vào các quy trình kiểm tra và phân loại sản phẩm đóng vai trò then chốt trong việc nâng cao hiệu suất sản xuất và đảm bảo chất lượng sản phẩm. Bài toán đặt ra là phát triển một hệ thống thông minh có khả năng tự động nhận diện và đo lường kích thước của các đối tượng di chuyển trên băng chuyền. Hệ thống này không chỉ giúp xác định chính xác vị trí và kích thước của từng sản phẩm mà còn cung cấp giải pháp toàn diện cho việc quản lý chất lượng, tự động phân loại và đếm sản phẩm một cách nhanh chóng và chính xác.

Để giải quyết bài toán trên nhóm em đã sử dụng hai tập dữ liệu: tập dữ liệu có sẵn (tập dữ liệu 1) và tập dữ liệu do nhóm tự thu thập (tập dữ liệu 2). Trên cơ sở của tập dữ liệu 1 nhóm đã thực nghiệm hai phương pháp chính là Machine Learning và Deep Learning và kết hợp hai phương pháp này với thuật toán truy vết đối tượng SORT (Simple Online and Realtime Tracking) để so sánh và tìm ra giải pháp tối ưu nhất và thực hiện nó trên tập dữ liệu thực tế do nhóm tự thu thập. Trong phương pháp Machine Learning mà nhóm tìm hiểu có ba thuật toán bao gồm phát hiện màu , nền đồng nhất và trừ nền nhưng ở đây nhóm chỉ thực nghiệm chính chỉ có hai thuật toán với SORT đó là thuật toán phát hiện màu [1-4] và thuật toán trừ nền [1, 5-7] trong đó nhóm tập trung vào hai kỹ thuật này để tận dụng đặc điểm màu sắc và độ tương phản trong việc xác định và phân loại đối tượng. Trong khi đó, phương pháp Deep Learning sử dụng mô hình YOLOv8 (You Only Look Once) [8-11] để phát hiện đối tượng.

Kết quả thực nghiệm cho thấy hệ thống đề xuất đạt được độ chính xác cao trong việc nhận diện và đo lường kích thước các đối tượng trên băng chuyền. Mô hình YOLOv8 không chỉ cung cấp độ chính xác cao trong việc phát hiện đối tượng mà còn đảm bảo tốc độ xử lý nhanh, phù hợp với yêu cầu thời gian thực. Việc tích hợp thuật toán SORT giúp tăng cường độ tin cậy và giảm thiểu sai sót do sự di chuyển của đối tượng. Phương pháp phát hiện màu sắc và trừ nền tuy có ưu điểm về tốc độ và sử dụng ít tài nguyên, nhưng bị hạn chế bởi các yếu tố môi trường như ánh sáng và nhiễu và có khả năng phân loại đối tượng kém đỏi với phát hiện màu và với trừ nền là không có.

Hệ thống này hứa hẹn mang lại giải pháp tự động hóa hiệu quả cho các quy trình đo lường kích thước, đếm và phân loại sản phẩm trong sản xuất công nghiệp.



## MỤC LỤC

MỤC LỤC.....	I
DANH MỤC HÌNH .....	V
DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT .....	VIII
DANH MỤC CÁC BẢNG.....	IX
DANH MỤC CÁC HÌNH VẼ, BIỂU ĐỒ .....	X
Chương 1. Giới thiệu bài toán.....	1
1.1 Câu hỏi nghiên cứu .....	1
1.2 Tâm quan trọng .....	4
1.2.1 Tính mới .....	4
1.2.2 Tính hữu dụng .....	4
1.3 Giới hạn nghiên cứu .....	5
1.4 Bố cục khóa luận .....	6
Chương 2. Lý thuyết nền tảng.....	8
2.1 Thị giác máy tính .....	8
2.1.1 Quá trình hình thành ảnh.....	8
2.1.2 Dữ liệu ảnh .....	9
2.1.3 Các thao tác trên dữ liệu ảnh .....	9
2.1.4 Một số bài toán về thị giác máy tính.....	10
2.2 Deep learning .....	12
2.2.1 Perceptron .....	12
2.2.2 Mạng neuron đa tầng (Multilayer Perceptron – MLP) .....	17
2.2.3 Mạng neuron tích chập (Convolutional Neural Network - CNN) .....	19

2.3 Các phương pháp đánh giá bài toán .....	24
2.3.1 Ma trận confusion .....	24
2.3.2 Accuracy .....	24
2.3.3 Precision .....	25
2.3.4 Độ nhạy (Recall) .....	25
2.3.5 F1-Score .....	25
2.3.6 IoU.....	26
2.3.7 AP.....	26
2.3.8 mAP.....	27
Chương 3. Các công trình liên quan .....	28
3.1 Object Detection (Phát hiện đối tượng) .....	28
3.1.1 Một số phương pháp Machine Learning .....	28
3.1.1.1 Biên của đối tượng .....	28
3.1.1.2 Không gian màu RGB, HSV .....	29
3.1.1.3 Thuật toán phát hiện đối tượng dựa trên màu .....	30
3.1.1.4 Thuật toán phát hiện đối tượng dựa trên nền đồng nhất .....	31
3.1.1.5 Thuật toán phát hiện đối tượng dựa trên việc trừ nền .....	32
3.1.2 Phương pháp YOLO (Deep Learning).....	34
3.1.2.1 Tổng quan về mô hình YOLO .....	34
3.1.2.2 Sự tiến hóa của YOLO từ v1 đến v7 .....	38
3.1.2.3 Tổng quan về mô hình YOLOv8 .....	55
3.2 Object Tracking (Truy vết đối tượng).....	60
3.2.1 Kalman Filter .....	60

3.2.2 Thuật toán Hungarian.....	66
3.2.3 Thuật toán SORT (Simple Online and Realtime Tracking): .....	68
3.3 Tạo ROI (Region Of Interest) .....	75
3.4 Tính kích thước đối tượng.....	77
<b>Chương 4. Giải pháp đề xuất .....</b>	<b>78</b>
4.1 Phân tích dữ liệu.....	78
4.1.1 Tập dữ liệu 1 .....	78
4.1.2 Tập dữ liệu 2 .....	81
4.2 Mô hình .....	84
4.3 Thực nghiệm .....	86
4.3.1 Phát hiện đối tượng bằng Machine Learning.....	87
4.3.1.1 Thuật toán phát hiện đối tượng dựa trên màu .....	87
4.3.1.2 Thuật toán phát hiện đối tượng dựa trên nền đồng nhất .....	89
4.3.1.3 Thuật toán phát hiện đối tượng dựa trên việc trừ nền .....	90
4.3.1.4 Kết hợp phát hiện đối tượng Machine Learning với SORT .....	91
4.3.2 Phát hiện đối tượng bằng Deep Learning .....	95
4.3.2.1 Phát hiện đối tượng bằng YOLOv8 (Deep Learning).....	95
4.3.2.2 Kết hợp YOLOv8 (Deep Learning) với SORT.....	96
4.3.2.3 So sánh đánh giá tổng quan hai phương pháp.....	105
4.4 Thực nghiệm với tập dữ liệu 2 .....	106
4.4.1.1 Giao diện ứng dụng .....	112
<b>KẾT LUẬN .....</b>	<b>119</b>
<b>TÀI LIỆU THAM KHẢO.....</b>	<b>120</b>

PHỤ LỤC .....	125
1. Code của các thành phần trong SORT .....	125
2. Ví dụ về Kalman Filter.....	130
3. Ví dụ về thuật toán Hungarian .....	134

## DANH MỤC HÌNH

Hình 1. Cấu tạo băng chuyền .....	2
Hình 2. Hình ảnh thực tế băng chuyền.....	2
Hình 3. Quá trình hoạt động của perceptron .....	12
Hình 4. Hàm Identify .....	13
Hình 5. Hàm Step .....	14
Hình 6. Hàm Sigmoid .....	14
Hình 7. Hàm tanh .....	15
Hình 8. Hàm ReLU .....	16
Hình 9. Cấu trúc MLP .....	17
Hình 10. Cấu trúc của CNN cho việc nhận dạng chữ số .....	20
Hình 11. Mô tả cách một phép toán convolution được thực hiện .....	21
Hình 12. Ví dụ về hoạt động của phép toán convolution.....	21
Hình 13. Ví dụ về Stride .....	22
Hình 14. Ví dụ về phép tính max pooling.....	23
Hình 15. Hình ban đầu và hình ảnh với các đường biên được vẽ lên .....	29
Hình 16. Không gian màu RGB .....	29
Hình 17. Không gian màu HSV .....	30
Hình 18.Kiến trúc YOLOv1.....	34
Hình 19. Hàm tính IoU.....	36
Hình 20. Quy trình hoạt động của YOLO .....	37
Hình 21. Quá trình phát triển của YOLO từ v1 đến v7 .....	38
Hình 22. Kiến trúc YOLOv1 .....	39
Hình 23. YOLOv2 xác định nhiều anchor boxes cho mỗi lưới .....	41
Hình 24. Kiến trúc YOLOv3.....	42
Hình 25. Kiến trúc YOLOv4.....	44
Hình 26. Kiến trúc YOLOv5.....	47
Hình 27. Kiến trúc YOLOv6.....	52
Hình 28. Kiến trúc YOLOv8.....	58

Hình 29. Quá trình lắp lại của Kalman Filter.....	65
Hình 30. Minh họa về thuật toán Hungarian.....	67
Hình 31. Biểu đồ độ chính xác và tốc độ giữa những phương pháp.....	69
Hình 32. Ví dụ về phát hiện đối tượng.....	70
Hình 33. Quy trình hoạt động của thuật toán SORT.....	72
Hình 34. Bảng thông số đánh giá các mô hình tracking .....	74
Hình 35. Hình ảnh bộ dữ liệu 1 .....	79
Hình 36. Hình ảnh Object 1 .....	79
Hình 37. Hình ảnh Object 2 .....	79
Hình 38. Hình ảnh Object 3 .....	80
Hình 39. Hình ảnh bộ dữ liệu 2.....	81
Hình 40. Hình ảnh HopKeo1 .....	82
Hình 41. Hình ảnh HopKeo2 .....	82
Hình 42. Hình ảnh MayNgheNhac.....	83
Hình 43. Hình ảnh XitThomMieng.....	83
Hình 44. Kiến trúc YOLOv8.....	84
Hình 45. Vẽ contours chung cho tất cả các màu .....	87
Hình 46. Vẽ contours riêng cho từng màu .....	88
Hình 47. Thực nghiệm thuật toán phát hiện đối tượng dựa trên nền đồng nhất .....	89
Hình 48.Thực nghiệm thuật toán phát hiện đối tượng dựa trên việc trừ nền.....	90
Hình 49. Kết quả khi kết hợp SORT và phát hiện đối tượng dựa trên màu.....	92
Hình 50. Lỗi khi kết hợp thêm ROI vào SORT với phát hiện dựa trên màu .....	92
Hình 51. Kết quả khi thực hiện Sort và thuật toán trừ nền(KNN) .....	94
Hình 52. Kết quả khi thực hiện Sort và thuật toán trừ nền(MOG2) .....	94
Hình 53. Lỗi khi thực hiện SORT, ROI và trừ nền.....	94
Hình 54. Thực nghiệm phát hiện đối tượng bằng YOLOv8 với tập dữ liệu 1.....	96
Hình 55. Kết quả khi kết hợp YOLO với SORT .....	98
Hình 56. So sánh Object 1 và 3 .....	98
Hình 57. Hình ảnh Yolo xác định sai khi Object 3 bị che 1 nửa .....	99

Hình 58. SORT và ROI với Yolov8 và ROI hình chữ nhật có phân loại .....	99
Hình 59. SORT và ROI với Yolov8 và ROI hình chữ nhật không phân loại .....	100
Hình 60. SORT và ROI với Yolov8 và ROI đa giác có phân loại.....	102
Hình 61. SORT và ROI với Yolov8 và ROI đa giác không phân loại.....	102
Hình 62. Hiển thị đánh nhãn 2 cách.....	106
Hình 63. Ma trận nhầm lẫn của mô hình 1.....	107
Hình 64. Ma trận nhầm lẫn của mô hình 2.....	108
Hình 65. Test với mô hình 1 .....	109
Hình 66. Test với mô hình 2 .....	110
Hình 67. Vật thể tham chiếu .....	110
Hình 68. Hình ảnh khi vật thể thực hiện tính kích thước.....	111
Hình 69. Hình ảnh thực nghiệm với tập dữ liệu 2.....	111
Hình 70. Giao diện chọn và đổi Model .....	112
Hình 71. Màn hình báo lỗi khi người dùng không chọn model .....	112
Hình 72. Màn hình chính của ứng dụng.....	113
Hình 73. Chức năng tính toán PPM .....	114
Hình 74. Chức năng theo dõi và tính kích thước đối tượng.....	115
Hình 75. Màn hình khi theo dõi và tính kích thước đối tượng .....	115
Hình 76. Giao diện thuật toán trừ nền.....	116
Hình 77. Giao diện thuật toán phát hiện màu .....	117
Hình 78. Màn hình tạo dữ liệu mới .....	118
Hình 79. Màn hình huấn luyện trực tiếp .....	118
Hình 80. Ví dụ đơn giản về Kalman Filter.....	130
Hình 81. Ví dụ đơn giản về Kalman Filter.....	131

## **DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT**

<b>STT</b>	<b>KÝ HIỆU, CHỮ VIẾT TẮT</b>	<b>Ý NGHĨA</b>
1	NMS	Non-maximum Suppression
2	MLP	Multilayer Perceptron
3	SORT	Simple Online And Realtime Tracking
4	YOLO	You Look Only Once
5	Bbox	Bounding Box (Hộp giới hạn)
6	ROI	Region Of Interest
7	V	Version
8	ML	Machine Learning
9	DL	Deep Learning
10	GPU	Graphics Processing Unit
11	CPU	Central Processing Unit
12	RAM	Random Access Memory
13	PPM	Pixel Per Metric
14	OD	Object Detection
15	OT	Object Tracking

## **DANH MỤC CÁC BẢNG**

STT	BẢNG
1	Bảng ví dụ về ma trận nhầm lẫn
2	Bảng ví dụ về ma trận chi phí
3	Bảng so sánh các phương pháp
4	Bảng độ chính xác của YOLOv8 có phân loại
5	Bảng ví dụ bảng giá trị đo của Kalman Filter
6	Bảng ví dụ thực hiện Kalman Filter
7	Bảng ví dụ thuật toán Hungarian

## **DANH MỤC CÁC HÌNH VẼ, BIỂU ĐỒ**

STT	HÌNH VẼ, BIỂU ĐỒ
1	Workflow của thuật toán phát hiện màu
2	Workflow của thuật toán phát dựa trên nền đồng nhất
3	Workflow của thuật toán phát dựa trên trừ nền
4	Biểu đồ thể hiện quy trình thực hiện giải pháp đề xuất
5	Biểu đồ thể hiện quy trình thực nghiệm
6	Biểu đồ thể hiện quy trình thực hiện phát hiện đối tượng bằng ML
7	Biểu đồ thể hiện quy trình thực hiện phát hiện đối tượng bằng DL

## Chương 1. Giới thiệu bài toán

### 1.1 Câu hỏi nghiên cứu

Trong lĩnh vực sản xuất công nghiệp việc áp dụng tự động hóa vào các khâu kiểm tra và phân loại sản phẩm đóng vai trò cực kỳ quan trọng để nâng cao hiệu quả sản xuất và đảm bảo chất lượng sản phẩm. Để đạt được điều này bài toán đặt ra là phát triển một hệ thống thông minh có khả năng tự động nhận diện và đo lường kích thước của các đối tượng di chuyển trên băng chuyền. Hệ thống này sẽ giúp xác định vị trí và kích thước của từng sản phẩm qua đó cung cấp một giải pháp toàn diện cho việc quản lý chất lượng và phân loại và đếm sản phẩm một cách tự động, nhanh chóng và chính xác.

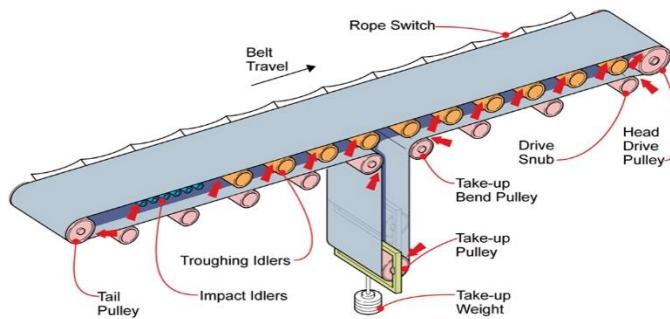
Mục tiêu cuối cùng là tối ưu hóa các quy trình kiểm soát chất lượng và giảm thiểu chi phí lao động, đồng thời nâng cao độ chính xác và giảm thiểu lỗi sản phẩm. Bằng cách này công nghệ sẽ góp phần quan trọng vào việc tạo ra một môi trường sản xuất hiện đại, thông minh và hiệu quả.

Trong nghiên cứu này, nhóm không chỉ tập trung vào việc phát triển hệ thống nhận diện và đo lường kích thước sản phẩm mà còn đặc biệt quan tâm đến cấu trúc và nguyên lý hoạt động của băng chuyền vì đây là yếu tố then chốt giúp tối ưu hóa quá trình kiểm soát chất lượng sản phẩm.

Cấu trúc của Băng Chuyền:

- **Băng Tải:** Là bộ phận chính thức tạo nên bề mặt chuyền động thường được làm từ cao su, PVC hoặc các vật liệu tổng hợp có độ bền cao và khả năng chịu lực tốt [12].
- **Khung băng chuyền:** Thường được chế tạo từ thép không gỉ hoặc nhôm, khung băng chuyền cung cấp sự vững chắc cần thiết để hỗ trợ băng tải và các bộ phận khác.

- Hệ thống truyền động: Bao gồm động cơ và hệ thống truyền lực, thường là pulley hoặc bánh răng, để dẫn động băng tải chuyển động liên tục.



Hình 1. Cấu tạo băng chuyền

Nguyên lý hoạt động của băng chuyền:

- Băng chuyền hoạt động dựa trên nguyên lý truyền động cơ học, nơi mà động cơ tạo ra lực kéo được truyền qua pulley hoặc bánh răng đến băng tải [12]. Điều này cho phép băng chuyền vận chuyển hàng hóa hoặc sản phẩm từ điểm này sang điểm khác trong một dây chuyền sản xuất.
- Các cảm biến được tích hợp dọc theo băng chuyền giúp phát hiện và phân loại sản phẩm dựa trên kích thước và các đặc điểm khác. Điều này đóng vai trò quan trọng trong việc tự động hóa các quy trình kiểm tra và đảm bảo chất lượng sản phẩm.



Hình 2. Hình ảnh thực tế băng chuyền

Input của bài toán bao gồm:

- Hình ảnh, video ghi hoặc chạy trực tiếp từ webcam ghi lại sự di chuyển của đối tượng trên băng chuyền và yêu cầu nhận diện nhãn cụ thể của từng đối tượng.
- Tỷ lệ của file pixel\_per\_metric được tính từ kích thước thực tế của vật thể tham chiếu.

Output của bài toán bao gồm:

- Danh sách các đối tượng được phát hiện trên băng chuyền, có nhãn và được phân loại cụ thể.
- Các thông số kích thước của mỗi đối tượng nhận diện được, bao gồm chiều rộng và chiều cao.

Ví dụ minh họa:

Giả sử băng chuyền vận chuyển các hộp sản phẩm có gắn nhãn cụ thể. Hệ thống sẽ phân tích hình ảnh từ camera để:

- Phát hiện sự hiện diện của các hộp dựa trên nhãn và phân loại và đếm số lượng chúng.
- Đo kích thước thực tế của mỗi hộp.

Dataset được sử dụng để giải quyết bài toán:

Tập dữ liệu 1: Đây là tập dữ liệu có sẵn được cung cấp bởi giảng viên hướng dẫn. Tập dữ liệu này khoảng một ngàn hình ảnh ghi lại từ camera quay băng chuyền, trong đó các đối tượng đa dạng về hình dạng, kích thước và màu sắc. Tập dữ liệu này sẽ được sử dụng trong giai đoạn đầu để thực nghiệm và so sánh các phương pháp khác nhau trong quá trình phát triển mô hình.

Tập dữ liệu 2: Đây là tập dữ liệu được nhóm tự thu thập, tập dữ liệu này sẽ bao gồm hàng ngàn hình ảnh từ webcam quay trực tiếp băng chuyền với các đối tượng.

Sẽ có sự đa dạng về hình dạng, kích thước và màu sắc và tập dữ liệu này phản ánh chính xác các điều kiện thực tế. Nó sẽ sử dụng với phương pháp tốt nhất được kết luận từ tập dữ liệu 1 và sẽ dùng để xây dựng hệ thống có khả năng phát hiện đối tượng, đếm số lượng và đo lường kích thước của đối tượng.

## 1.2 Tầm quan trọng

### 1.2.1 Tính mới

Đề tài này mang lại tiếp cận mới mẻ trong việc xử lý và phân tích tự động các đối tượng trên băng chuyền sản xuất với điểm nhấn là:

- Phương pháp học sâu chính xác: Sử dụng các mô hình học sâu tiên tiến ở đây nhóm sử dụng YOLOv8 với dữ liệu được gán nhãn rõ ràng, hệ thống có khả năng học và phân biệt đối tượng cùng với đếm đối tượng, tính kích thước với độ chính xác cao khá cao.
- Công nghệ xử lý ảnh tiên tiến: Áp dụng những phát triển mới nhất trong lĩnh vực thị giác máy tính và xử lý ảnh cho phép hệ thống xác định kích thước vật lý của đối tượng một cách chính xác trong môi trường sản xuất động.
- Ứng dụng đa ngành nghề: Cung cấp giải pháp có khả năng được tùy chỉnh cho nhiều ngành công nghiệp khác nhau.

Tính mới này không chỉ góp phần vào sự phát triển của thị giác máy tính và học sâu mà còn thúc đẩy tiến trình tự động hóa trong sản xuất công nghiệp.

### 1.2.2 Tính hữu dụng

Đề tài này giải quyết bài toán quan trọng trong sản xuất công nghiệp là cách thức tự động hóa việc theo dõi và đếm số lượng và tính kích thước sản phẩm mà không cần đến sự can thiệp của con người quá nhiều.

Ứng dụng thực tế của đề tài bao gồm:

- Tăng cường tự động hóa: Hệ thống giúp tự động hóa quy trình kiểm tra số lượng sản phẩm, giảm phụ thuộc vào kiểm tra thủ công, nâng cao năng suất và giảm chi phí lao động.
- Tính kích thước sản phẩm: Tính kích thước của sản phẩm trên băng chuyền cho phép phát hiện kích thước khi đi vào vùng quan tâm, tránh lãng phí thời gian.
- Cải thiện chất lượng: Đảm bảo rằng tất cả sản phẩm đều đáp ứng tiêu chuẩn chất lượng trước khi đến tay người tiêu dùng, tăng uy tín và hài lòng của khách hàng.
- Linh hoạt trong sản xuất: Hệ thống có thể được cài đặt và điều chỉnh dễ dàng để phù hợp với nhiều loại sản phẩm và kích thước khác nhau, giúp đáp ứng nhanh các yêu cầu.

Nhờ vào khả năng nhận diện và đo lường tự động. Hệ thống không chỉ cung cấp giải pháp mà còn tìm cách cải thiện chất lượng sản phẩm và còn góp phần vào việc thực thi các tiêu chuẩn chất lượng trong sản xuất công nghiệp.

### 1.3 Giới hạn nghiên cứu

Trong quá trình nghiên cứu và phát triển hệ thống nhận diện và đo lường kích thước đối tượng trên băng chuyền có một số ràng buộc và giả định cần được đưa ra để giới hạn phạm vi bài toán và tập trung vào các yếu tố quan trọng nhất.

- Loại đối tượng và băng chuyền: Hệ thống được thiết kế để nhận diện và đo lường các đối tượng. Băng chuyền sử dụng trong khóa luận là loại băng chuyền mini và được áp dụng trong học tập và nghiên cứu.
- Điều kiện ánh sáng: Nghiên cứu giả định rằng điều kiện ánh sáng trong môi trường sản xuất là ổn định và đủ sáng để camera có thể thu thập hình ảnh chất lượng cao. Các tình huống ánh sáng thay đổi mạnh hoặc ánh sáng kém sẽ có thể ảnh hưởng nhẹ.

- **Khả năng xử lý của hệ thống:** Hệ thống được thiết kế để hoạt động trên các thiết bị có khả năng xử lý mạnh mẽ như GPU hoặc các máy tính hiệu suất cao.
- **Phạm vi dữ liệu huấn luyện:** Dữ liệu huấn luyện cho mô hình YOLO và thuật toán phát hiện màu, trừ nền được thu thập từ các loại sản phẩm cụ thể cụ thể là đối với mô hình 1 cho tập dữ liệu 1 là gồm 3 vật thể và tập dữ liệu 2 là gồm 4 vật thể. Các loại sản phẩm không có trong dữ liệu huấn luyện có thể dẫn đến kết quả không chính xác.
- **Tính chính xác và tốc độ xử lý:** Khó khăn lớn nhất là cân bằng giữa tính chính xác và tốc độ xử lý với ưu tiên là đáp ứng yêu cầu thời gian thực. Các yếu tố khác như tính bền vững của hệ thống trong môi trường khắc nghiệt hoặc khả năng xử lý lượng lớn dữ liệu trong thời gian dài không được nghiên cứu chi tiết.

## 1.4 Bố cục khóa luận

Chương 1: Giới thiệu bài toán

Chương này trình bày câu hỏi nghiên cứu và tầm quan trọng của việc phát triển hệ thống nhận diện và đo lường kích thước đối tượng trong sản xuất công nghiệp. Nó cũng nêu rõ giới hạn nghiên cứu và mục tiêu cụ thể của khóa luận.

Chương 2: Lý thuyết nền tảng

Chương này cung cấp cái nhìn tổng quan về các lý thuyết và công nghệ cơ bản được sử dụng và tìm hiểu trong nghiên cứu bao gồm thị giác máy tính (Computer Vision), học sâu (Deep Learning) và các phương pháp đánh giá hiệu suất mô hình. Nó giới thiệu các khái niệm như Perceptron, Mạng neuron đa tầng (MLP), và Mạng neuron tích chập (CNN) cũng như các phương pháp đánh giá như ma trận confusion, accuracy, precision, recall, F1-Score, IoU, AP, và mAP.

### Chương 3: Các công trình liên quan

Chương này tổng hợp và đánh giá các công trình nghiên cứu liên quan đến phát hiện và theo dõi đối tượng. Nó trình bày các phương pháp truyền thống như phát hiện màu, trừ nền và nền đồng nhất cùng với các phương pháp hiện đại như YOLO và SORT. Chương này cũng xem xét sự tiến hóa của mô hình YOLO qua các phiên bản từ YOLOv1 đến YOLOv8 nó cũng thể hiện các cách vẽ ROI và cách tính kích thước của đối tượng.

### Chương 4: Giải pháp đề xuất

Chương này mô tả chi tiết giải pháp đề xuất của khóa luận bao gồm quá trình phân tích dữ liệu, phương pháp, mô hình phát hiện đối tượng và các thực nghiệm tiến hành. Nó trình bày thực nghiệm cả hai phương pháp Deep Learning và Machine Learning kết hợp với SORT để phát hiện và theo dõi đối tượng. Chương này cũng thể hiện đánh giá và so sánh kết quả của cả hai phương pháp Deep Learning và Machine Learning từ đó sẽ chọn ra được phương pháp tốt nhất để thực hiện cho việc triển khai thực tế và việc xây dựng ứng demo cũng được trình bày trong chương này.

### Kết luận và hướng phát triển

Phần cuối cùng của khóa luận tóm tắt những kết quả đạt được, nhấn mạnh những đóng góp chính của nghiên cứu và đề xuất các hướng phát triển tiếp theo để cải thiện hệ thống.

## Chương 2. Lý thuyết nền tảng

### 2.1 Thị giác máy tính

Thị giác máy tính (Computer Vision) [13] là một lĩnh vực của khoa học máy tính, nhằm mục đích mô phỏng khả năng nhìn và hiểu hình ảnh của con người bằng cách sử dụng máy tính. Thị giác máy tính bao gồm nhiều nhiệm vụ như nhận diện đối tượng, phân đoạn ảnh, phát hiện và theo dõi đối tượng, phân loại hình ảnh, tái tạo 3D và nhiều ứng dụng khác. Một số ứng dụng phổ biến của thị giác máy tính bao gồm nhận diện khuôn mặt, xe tự lái, phân tích y tế từ hình ảnh và còn rất nhiều ứng dụng.

#### 2.1.1 Quá trình hình thành ảnh

Quá trình hình thành ảnh [13] trong thị giác máy tính liên quan đến việc thu thập và xử lý ánh sáng từ cảnh vật xung quanh để tạo ra một hình ảnh số. Quá trình này bao gồm các bước sau:

- Thu thập ánh sáng:
  - Mô tả: Ánh sáng từ nguồn sáng chiếu tới các đối tượng trong cảnh và phản xạ lại.
  - Chi tiết: Các cảm biến ánh sáng trong máy ảnh (camera sensor) sẽ thu thập ánh sáng này và chuyển đổi nó thành tín hiệu điện.
- Chuyển đổi quang điện (Photoelectric Conversion):
  - Mô tả: Quá trình chuyển đổi ánh sáng thành tín hiệu điện.
  - Chi tiết: Các cảm biến CCD (Charge-Coupled Device) hoặc CMOS (Complementary Metal-Oxide-Semiconductor) trong máy ảnh sẽ thực hiện quá trình này.
- Xử lý tín hiệu (Signal Processing):
  - Mô tả: Tín hiệu điện từ cảm biến được chuyển đổi thành dữ liệu số.

- Chi tiết: Các tín hiệu này được khuếch đại, lọc nhiễu và chuyển đổi từ dạng tương tự (analog) sang dạng số (digital) để tạo ra hình ảnh số.
- Tạo hình ảnh số (Image Formation):
  - Mô tả: Tạo ra một ma trận các pixel từ tín hiệu số.
  - Chi tiết: Mỗi pixel đại diện cho một điểm màu duy nhất trong hình ảnh, được biểu diễn bằng các giá trị màu sắc như RGB.

### 2.1.2 Dữ liệu ảnh

Dữ liệu ảnh [13] là thông tin số đại diện cho một hình ảnh kỹ thuật số. Dữ liệu này có thể ở nhiều dạng khác nhau, tùy thuộc vào cách lưu trữ và xử lý hình ảnh.

- Pixel và màu sắc:
  - Mô tả: Độ phân giải của một hình ảnh là số lượng pixel được sử dụng để biểu diễn hình ảnh đó.
  - Chi tiết: Độ phân giải cao hơn có nghĩa là hình ảnh chi tiết hơn và sắc nét hơn, thường được biểu thị dưới dạng số lượng pixel theo chiều ngang và chiều dọc (ví dụ: 1920x1080).
- Định dạng ảnh:
  - Mô tả: Các định dạng lưu trữ hình ảnh khác nhau như JPEG, PNG, BMP, TIFF.
  - Chi tiết: Mỗi định dạng có các đặc điểm riêng về chất lượng, kích thước tệp và phương pháp nén dữ liệu.

### 2.1.3 Các thao tác trên dữ liệu ảnh

Các thao tác trên dữ liệu ảnh là những phương pháp và kỹ thuật được sử dụng để xử lý và biến đổi hình ảnh số nhằm phục vụ cho các nhiệm vụ thị giác máy tính khác nhau.

---

- Biến đổi không gian màu (Color Space Transformation):
  - Mô tả: Chuyển đổi giữa các không gian màu khác nhau như từ RGB sang HSV [4].
  - Chi tiết: Biến đổi này giúp xử lý các đặc trưng màu sắc của hình ảnh một cách hiệu quả hơn.
- Lọc ảnh (Image Filtering):
  - Mô tả: Sử dụng các bộ lọc để làm mịn hoặc làm sắc nét hình ảnh.
  - Chi tiết: Các bộ lọc thông dụng bao gồm lọc Gaussian để làm mờ ảnh và lọc Sobel để phát hiện biên cạnh [14].
- Phép toán hình thái học (Morphological Operations):
  - Mô tả: Thao tác với các cấu trúc hình dạng trong ảnh.
  - Chi tiết: Các phép toán như dilation (giãn nở), erosion (xói mòn), opening (mở), và closing (đóng) thường được sử dụng trong xử lý ảnh nhị phân [15].
- Biến đổi Fourier (Fourier Transform):
  - Mô tả: Phân tích tần số của hình ảnh.
  - Chi tiết: Biến đổi Fourier giúp chuyển hình ảnh từ miền không gian sang miền tần số để phân tích các thành phần tần số.

#### 2.1.4 Một số bài toán về thi giác máy tính

Thi giác máy tính giải quyết nhiều bài toán phức tạp liên quan đến việc hiểu và phân tích hình ảnh và video. Dưới đây là một số bài toán phổ biến:

- Nhận diện đối tượng (Object Recognition):
  - Mô tả: Xác định và phân loại các đối tượng trong hình ảnh hoặc video.

- Ứng dụng: Nhận diện khuôn mặt, biển số xe, sản phẩm trong cửa hàng.
- Phát hiện đối tượng (Object Detection):
  - Mô tả: Xác định vị trí của các đối tượng trong hình ảnh hoặc video và vẽ các hộp giới hạn (bounding box) xung quanh chúng.
  - Ứng dụng: Giám sát an ninh, xe tự lái, robot công nghiệp.
- Phân đoạn ảnh (Image Segmentation):
  - Mô tả: Chia hình ảnh thành các vùng có ý nghĩa, mỗi vùng đại diện cho một đối tượng hoặc một phần của đối tượng.
  - Ứng dụng: Y tế (phân đoạn các mô từ ảnh MRI/CT), nông nghiệp (phân đoạn cây trồng và cỏ dại).
- Theo dõi đối tượng (Object Tracking):
  - Mô tả: Theo dõi sự di chuyển của các đối tượng qua các khung hình trong một video.
  - Ứng dụng: Giám sát giao thông, thể thao, nghiên cứu hành vi động vật.
- Tái tạo 3D (3D Reconstruction):
  - Mô tả: Tái tạo mô hình 3D của các đối tượng hoặc cảnh từ các hình ảnh 2D.
  - Ứng dụng: Thiết kế công nghiệp, khảo cổ học, giải trí (game, phim ảnh).
- Nhận diện khuôn mặt (Face Recognition):
  - Mô tả: Nhận dạng và xác định danh tính của các khuôn mặt trong hình ảnh hoặc video.
  - Ứng dụng: Bảo mật, mở khóa thiết bị, nhận diện khách hàng trong bán lẻ.

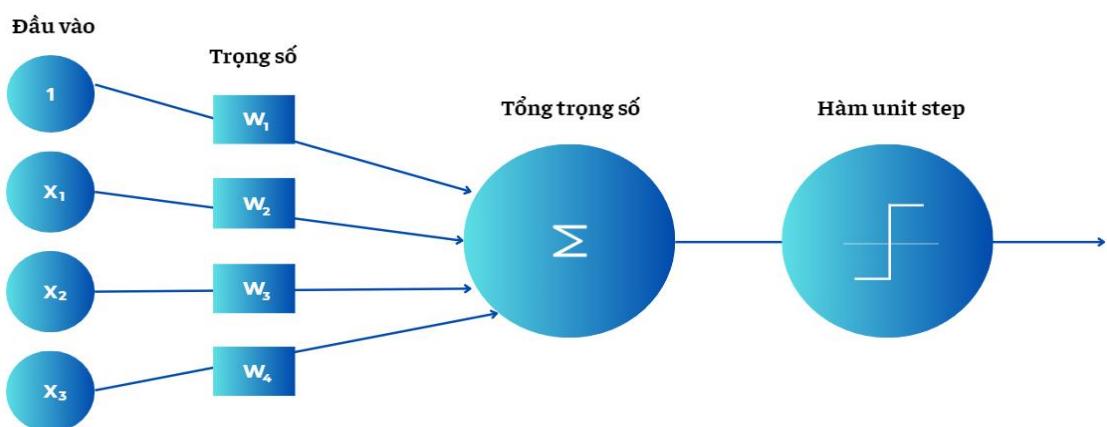
## 2.2 Deep learning

Deep Learning (Học sâu) [16] là một nhánh của học máy (Machine Learning) dựa trên các mạng neural (neural networks) với nhiều lớp (layers) để mô hình hóa các mẫu phức tạp trong dữ liệu. Các mạng này có khả năng học và trích xuất các đặc trưng từ dữ liệu một cách tự động đặc biệt là từ dữ liệu lớn như hình ảnh và video.

### 2.2.1 Perceptron

- Khái niệm:
  - Perceptron là đơn vị cơ bản nhất của một mạng neural, được giới thiệu bởi Frank Rosenblatt vào năm 1958. Đây là một mô hình đơn giản của một neuron sinh học, được sử dụng để phân loại các mẫu dữ liệu.
- Cấu trúc:
  - Perceptron bao gồm một tập hợp các đầu vào, các trọng số tương ứng, một bộ cộng, và một hàm kích hoạt

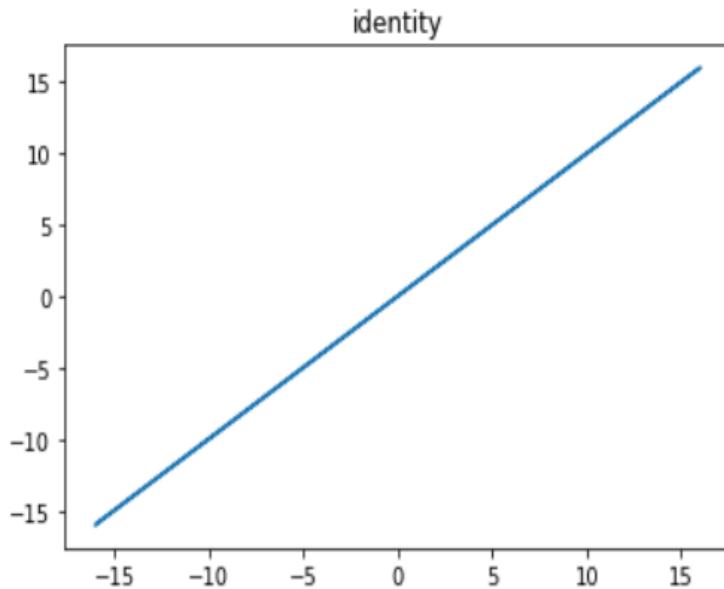
Perceptron hiểu theo nghĩa tiếng việt là một tế bào thần kinh đơn giản hơn nó là một hàm toán học nhận đầu vào từ một hoặc nhiều số thực hiện các phép tính và trả kết quả đầu ra. Các trọng số ẩn (weights) là thứ mà chúng ta đang tìm và sẽ được xác định trong quá trình huấn luyện dựa trên dữ liệu huấn luyện [17].



Hình 3. Quá trình hoạt động của perceptron

Trong hình ảnh trên ta có thể thấy dữ liệu đầu vào được biểu diễn bởi vector  $X = [x_0, x_1, \dots, x_n]$  và các trọng số cần học được biểu diễn bởi vector  $W = [w_0, w_1, \dots, w_n]$ , đi qua perceptron và kết quả đầu ra được trả thông qua một hàm gọi là unit step function hoặc có thể hiểu là hàm kích hoạt trong trường hợp trên là binary. Mục đích của hàm này sử dụng để cho perceptron có thể ra được quyết định. Ví dụ chúng ta đang cần phân biệt khuôn mặt là nam hay nữ thì sau khi đưa khuôn mặt qua perceptron chúng ta sẽ tính toán và thu được một số thực (giả sử là 0.75) và tác dụng của unit step function là đặt ra một ngưỡng cho chúng ta lựa chọn. Chẳng hạn như lớn hơn 0.75 thì là nam và ngược lại là nữ [17].

- Các hàm kích hoạt (Activation):
  - Hàm Identity: Tổ hợp các hàm tuyến tính là hàm tuyến tính.
    - $\sigma(x) = x$

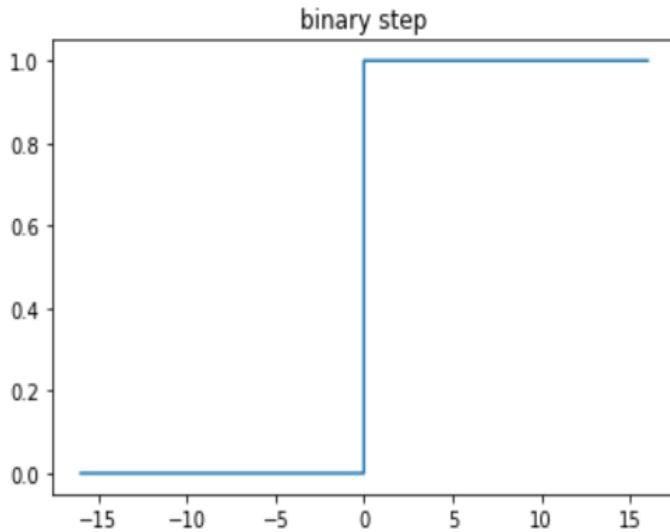


Hình 4. Hàm Identify

- Khuyết điểm: Không thể dùng để giải quyết các bài toán phi tuyến.

o Hàm Step

- $\sigma(x) = \begin{cases} 0 & \text{for } x \leq 0 \\ 1 & \text{for } x > 0 \end{cases}$  [17]

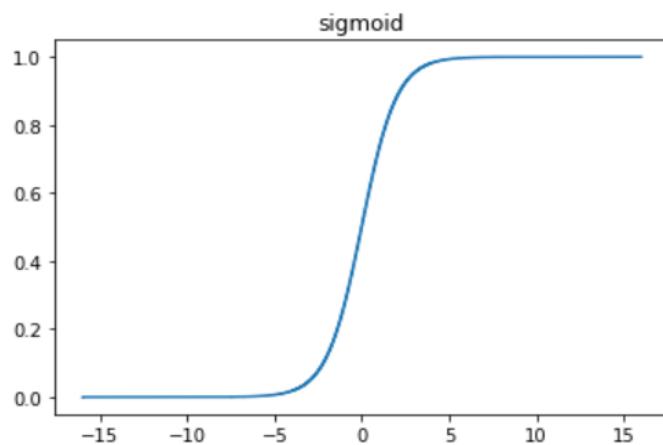


Hình 5. Hàm Step

- Khuyết điểm: Đạo hàm hàm = 0 → có vấn đề khi áp dụng phương pháp gradient descent để huấn luyện mạng.

o Hàm Sigmoid

- $\sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1}$  [17]

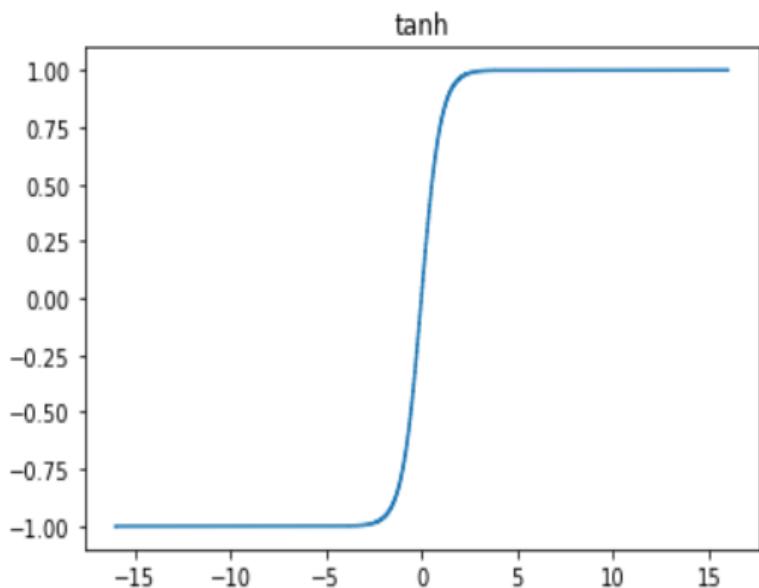


Hình 6. Hàm Sigmoid

- Ưu điểm: Liên tục và có đạo hàm mọi nơi, đối xứng qua trục y và tiệm cận các giá trị bảo hòa.
- Khuyết điểm: Các giá trị đầu ra không tập trung tại 0, các neuron bảo hòa bị triệt tiêu gradient (vì các đạo hàm cực nhỏ).

○ Hàm tanh

- $\sigma(x) = \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} = \frac{e^{2x} - 1}{e^{2x} + 1}$  [17]

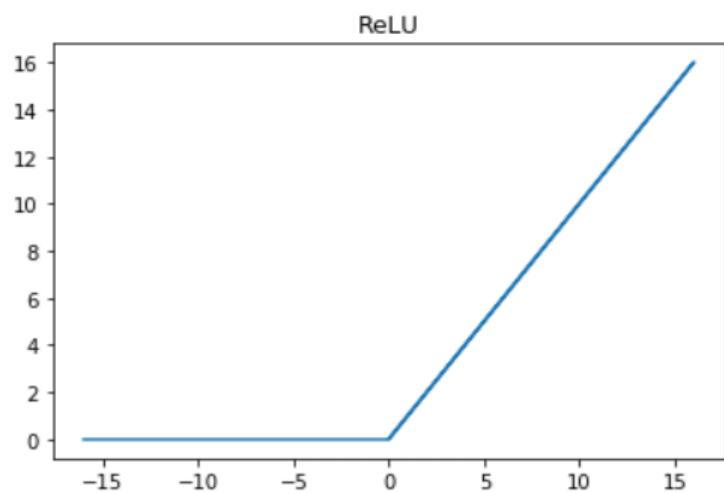


Hình 7. Hàm tanh

- Các giá trị đầu ra tập trung tại 0
- Các neuron bảo hòa vẫn bị triệt tiêu gradient (vì các đạo hàm cực nhỏ)

○ Hàm ReLU

- Ra đời năm 2003
- Phổ biến năm 2015
- $\sigma(x) = \begin{cases} x & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$  [17]  
 $= \max(0, x)$



Hình 8. Hàm ReLU

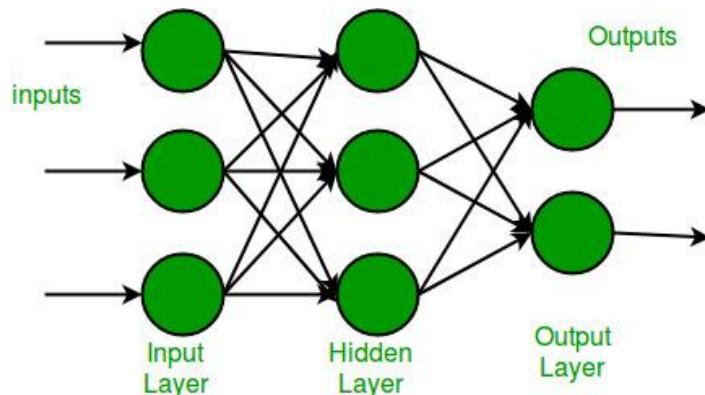
- **Ưu điểm:** Giải quyết các vấn đề trước đây
- **Khuyết điểm:** Không có đạo hàm tại 0 và loại bỏ giá trị âm

### 2.2.2 Mạng neuron đa tầng (Multilayer Perceptron – MLP)

- Khái niệm:

- Multilayer Perceptron (MLP) là một loại mạng neural feedforward với ít nhất ba lớp: một lớp đầu vào (input layer), một hoặc nhiều lớp ẩn (hidden layers), và một lớp đầu ra (output layer). Đây là một dạng mở rộng của mô hình perceptron đơn giản [18, 19].

- Cấu trúc:



Hình 9. Cấu trúc MLP

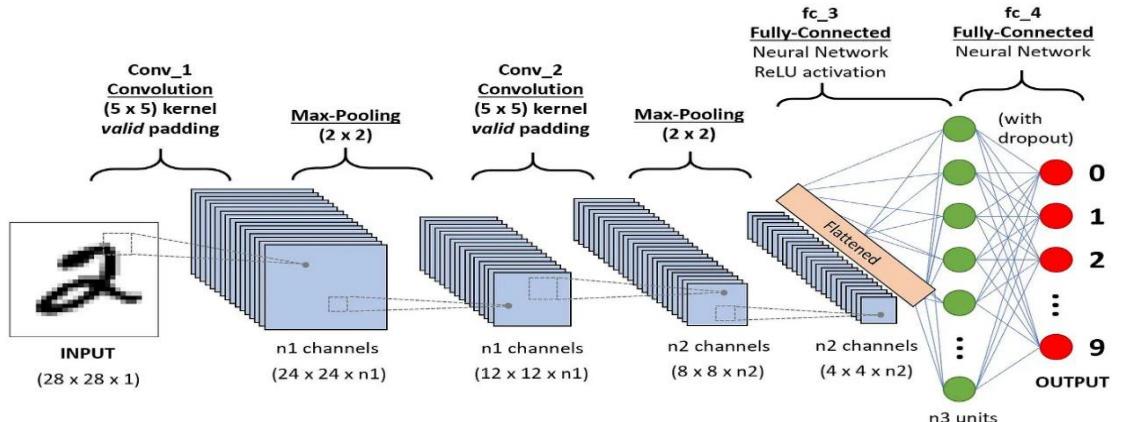
- Input Layer (Lớp đầu vào):
  - Nhận dữ liệu đầu vào dưới dạng các vector.
  - Mỗi nút trong lớp đầu vào đại diện cho một đặc trưng của dữ liệu đầu vào.
- Hidden Layers (Các lớp ẩn):
  - Một hoặc nhiều lớp neuron nằm giữa lớp đầu vào và lớp đầu ra.
  - Mỗi lớp ẩn bao gồm nhiều neuron, và mỗi neuron nhận tín hiệu từ tất cả các neuron của lớp trước đó.
  - Sử dụng các hàm kích hoạt phi tuyến (non-linear activation function) như ReLU (Rectified Linear Unit), Sigmoid hoặc Tanh để học các mối quan hệ phi tuyến giữa các đặc trưng.

- MLP có hai hay nhiều lớp ẩn thì gọi là deep neural network (DNN)
- Output Layer (Lớp đầu ra):
  - Sản sinh đầu ra của mạng neural.
  - Số lượng neuron trong lớp đầu ra phụ thuộc vào nhiệm vụ cụ thể ví dụ: với bài toán phân loại nhị phân thì có một neuron đầu ra với hàm kích hoạt Sigmoid, còn với bài toán phân loại đa lớp thì có một neuron cho mỗi lớp với hàm kích hoạt Softmax.
- Hoạt động của MLP:
  - Lan truyền tiến (Forward Propagation):
    - Dữ liệu đầu vào được truyền từ lớp đầu vào qua các lớp ẩn đến lớp đầu ra.
    - Tại mỗi neuron, tín hiệu đầu vào được nhân với trọng số tương ứng, cộng với hệ số bù (bias), và sau đó được chuyển qua hàm kích hoạt để tạo ra đầu ra của neuron.
  - Lan truyền ngược (Backpropagation):
    - Quá trình cập nhật các trọng số của mạng neural dựa trên lỗi giữa đầu ra thực tế và đầu ra dự đoán.
    - Sử dụng thuật toán Gradient Descent để giảm thiểu hàm mất mát (loss function).
    - Gradient của hàm mất mát đối với các trọng số được tính toán thông qua phương pháp lan truyền ngược, và các trọng số được cập nhật theo hướng giảm dần của gradient.
- Kết luận:
  - MLP là một mô hình mạng neural cơ bản nhưng mạnh mẽ, có khả năng học và mô hình hóa các mối quan hệ phi tuyến trong dữ liệu. MLP thường được sử dụng trong các bài toán phân loại và hồi quy và là nền tảng của nhiều mô hình học sâu phức tạp hơn.

### 2.2.3 Mạng neuron tích chập (Convolutional Neural Network - CNN)

- Khái niệm:
  - CNN [20, 21] là một loại mạng neural được thiết kế đặc biệt để xử lý dữ liệu có cấu trúc lưới, chẳng hạn như hình ảnh. CNN sử dụng các phép toán convolution để tự động trích xuất các đặc trưng từ hình ảnh, giúp mạng học cách nhận diện các đặc trưng quan trọng như cạnh, góc và các mẫu phức tạp hơn.
- Cấu trúc:
  - Lớp tích chập (Convolutional Layer):
    - Đây là lớp chính trong CNN, chịu trách nhiệm trích xuất các đặc trưng từ hình ảnh đầu vào.
    - Lớp tích chập sử dụng các bộ lọc để thực hiện phép toán convolution trên hình ảnh đầu vào.
  - Lớp phi tuyến (Non-linear Layer):
    - Sau mỗi lớp tích chập một hàm kích hoạt phi tuyến như ReLU áp dụng để tăng tính phi tuyến vào hệ thống.
    - Hàm kích hoạt ReLU chuyển tất cả các giá trị âm thành 0 và giữ nguyên các giá trị dương, giúp mạng neural học các mối quan hệ phi tuyến.
  - Lớp gộp (Pooling Layer):
    - Lớp gộp giảm kích thước của biểu diễn không gian (spatial representation) của hình ảnh, giúp giảm số lượng tham số và tính toán trong mạng.

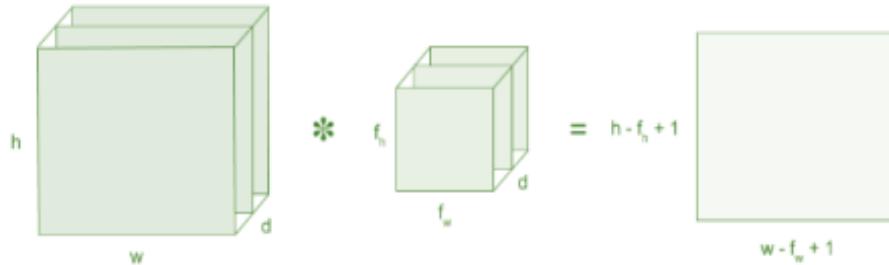
- Các phép toán gộp phổ biến bao gồm max pooling (chọn giá trị lớn nhất trong mỗi vùng) và average pooling (tính trung bình các giá trị trong mỗi vùng).
- Lớp fully connected (Fully Connected Layer):
  - Sau các lớp tích chập và gộp, các lớp fully connected (kết nối đầy đủ) được sử dụng để thực hiện phân loại dựa trên các đặc trưng đã trích xuất.
  - Mỗi neuron trong lớp fully connected kết nối với tất cả các neuron trong lớp trước đó.



Hình 10. Cấu trúc của CNN cho việc nhận dạng chữ số

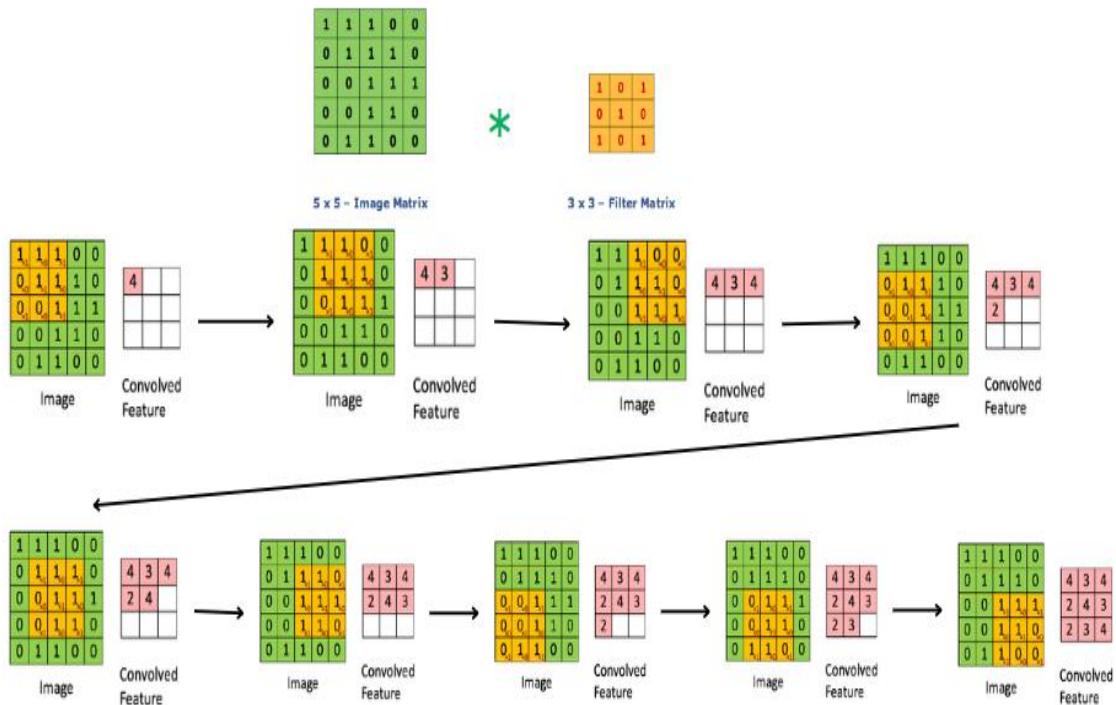
- Chi tiết hơn về CNN:
  - Lớp tích chập: Nó sẽ là lớp đầu tiên để trích xuất các tính năng từ hình ảnh đầu vào. Tích chập duy trì mối quan hệ giữa các pixel bằng cách tìm hiểu các tính năng hình ảnh bằng cách sử dụng các ô vuông nhỏ của dữ liệu đầu vào. Nó là 1 phép toán có 2 đầu vào như ma trận hình ảnh và 1 bộ lọc [20, 22].

- An image matrix (volume) of dimension ( $h \times w \times d$ )
- A filter ( $f_h \times f_w \times d$ )
- Outputs a volume dimension ( $h - f_h + 1 \times (w - f_w + 1) \times 1$ )



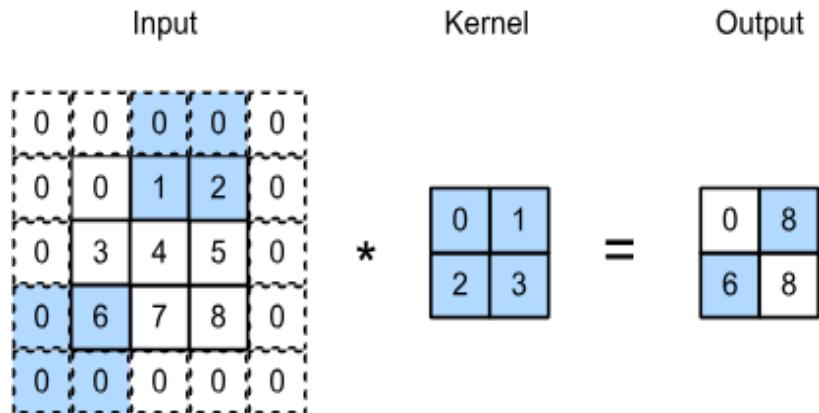
Hình 11. Mô tả cách một phép toán convolution được thực hiện

- Xem xét 1 ma trận  $5 \times 5$  có giá trị pixel là 0 và 1 và bộ lọc  $3 \times 3$  như hình dưới đây:



Hình 12. Ví dụ về hoạt động của phép toán convolution

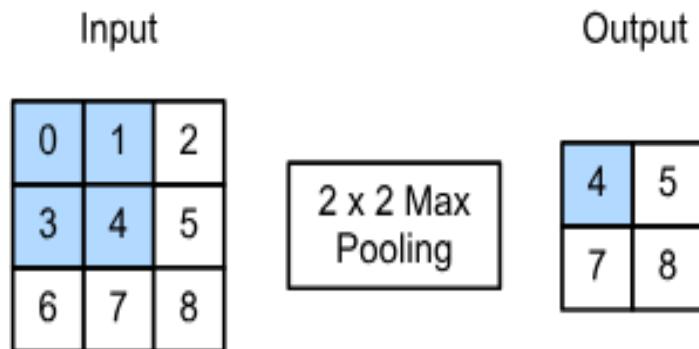
- Ta sẽ thấy lớp tích chập của ma trận hình ảnh  $5 \times 5$  nhân với ma trận bộ lọc  $3 \times 3$  gọi là 'Feature Map'.
- Sự kết hợp của 1 hình ảnh với các bộ lọc khác nhau có thể thực hiện các hoạt động như phát hiện cạnh, làm mờ và làm sắc nét bằng cách áp dụng các bộ lọc.
- Bước nhảy (Stride)
  - Stride là số pixel thay đổi trên ma trận đầu vào. Khi stride là 1 thì ta di chuyển các kernel 1 pixel. Khi stride là 2 thì ta di chuyển các kernel đi 2 pixel và tiếp tục như vậy.



Hình 13. Ví dụ về Stride

- Công dụng: Thay đổi stride ảnh hưởng đến kích thước của đầu ra và tốc độ tính toán của mạng. Stride lớn hơn làm giảm kích thước của đầu ra và tăng tốc độ tính toán, nhưng có thể mất thông tin chi tiết.
- Đệm (Padding)
  - Padding là quá trình thêm các hàng và cột xung quanh biên của hình ảnh để giữ nguyên kích thước của hình ảnh sau khi áp dụng phép toán convolution.

- Các loại padding: Zero Padding: Thêm các giá trị 0 xung quanh biên của hình ảnh và Valid Padding: Không thêm bất kỳ giá trị nào (kích thước hình ảnh giảm sau mỗi lần convolution).
- Công dụng: Padding giúp giữ nguyên kích thước của hình ảnh, tránh mất mát thông tin biên và cải thiện hiệu suất của mạng.
- Lớp gộp: Lớp pooling sẽ giảm bớt số lượng tham số khi hình ảnh quá lớn. Không gian pooling còn được gọi là lấy mẫu con hoặc lấy mẫu xuống làm giảm kích thước của mỗi map nhưng vẫn giữ lại thông tin quan trọng. Các pooling có thể có nhiều loại khác nhau: Max Pooling, Average Pooling, Sum Pooling.
- Max pooling lấy phần tử lớn nhất từ ma trận đối tượng hoặc lấy tổng trung bình. Tổng tất cả các phần tử trong map gọi là sum pooling.



Hình 14. Ví dụ về phép tính max pooling

- Kết luận: CNN có rất nhiều ứng dụng như nhận diện và phát hiện đối tượng bao gồm cả phân đoạn và phân lớp các đối tượng. CNN đã chứng minh được hiệu quả vượt trội trong nhiều ứng dụng thị giác máy tính như nhận diện đối tượng, phân đoạn ảnh và phát hiện đối tượng, và tiếp tục là một công cụ quan trọng trong nghiên cứu và ứng dụng AI.

## 2.3 Các phương pháp đánh giá bài toán

### 2.3.1 Ma trận confusion

- Khái niệm
  - Ma trận nhầm lẫn là một bảng biểu diễn số lượng dự đoán đúng và sai của mô hình phân loại, được sắp xếp theo các lớp thực và lớp dự đoán [19].
- Cấu trúc:
  - True Positive (TP): Số lượng mẫu được dự đoán đúng là lớp positive.
  - True Negative (TN): Số lượng mẫu được dự đoán đúng là lớp negative.
  - False Positive (FP): Số lượng mẫu được dự đoán sai là lớp positive (nhưng thực tế là negative).
  - False Negative (FN): Số lượng mẫu được dự đoán sai là lớp negative (nhưng thực tế là positive)

	Predicted Positive	Predicted Negative
Actual Positive	TP (True Positive)	FN (False Negative)
Actual Negative	FP (False Positive)	TN (True Negative)

Bảng 1. Ví dụ về ma trận nhầm lẫn

### 2.3.2 Accuracy

- Khái niệm:
  - Accuracy (độ chính xác) là tỷ lệ phần trăm các dự đoán đúng trên tổng số dự đoán.
- Công thức:
  - $$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} [19]$$

- Ví dụ: Giả sử chúng ta có 100 mẫu, trong đó có 50 mẫu positive và 50 mẫu negative. Nếu mô hình dự đoán đúng 90 mẫu, thì accuracy là 90%.

### 2.3.3 Precision

- Khái niệm:
  - Precision (độ chính xác của positive) là tỷ lệ các dự đoán positive đúng trên tổng số dự đoán positive.
- Công thức:
  - $\text{Precision} = \frac{TP}{TP+FP}$  [19]
- Ví dụ: Nếu mô hình dự đoán có 30 mẫu positive, trong đó có 25 mẫu đúng và 5 mẫu sai thì precision là  $\frac{25}{30} = 0.83$ .

### 2.3.4 Độ nhạy (Recall)

- Khái niệm:
  - Recall (độ nhạy) là tỷ lệ các dự đoán positive đúng trên tổng số đối tượng thực sự positive.
- Công thức:
  - $\text{Recall} = \frac{TP}{TP+FN}$  [19]
- Ví dụ: Nếu trong số 50 mẫu positive thực tế, mô hình dự đoán đúng 40 mẫu thì recall là  $\frac{40}{50} = 0.8$ .

### 2.3.5 F1-Score

- Khái niệm:
  - F1-Score là trung bình điều hòa của Precision và Recall, cung cấp một thước đo cân bằng giữa Precision và Recall.

- Công thức:

- $F1\text{-Score} = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$  [19]

- Ví dụ: Với Precision là 0.83 và Recall là 0.8 thì F1-Score sẽ là:

- $2 \cdot \frac{0.83 \cdot 0.8}{0.83 + 0.8} \approx 0.815.$

### 2.3.6 IoU

- Khái niệm:

- IoU là một thước đo được sử dụng trong các bài toán phát hiện đối tượng để đánh giá mức độ trùng khớp giữa hộp giới hạn dự đoán và hộp giới hạn thực tế.

- Công thức:

- $\text{IoU} = \frac{\text{Diện tích phần giao}}{\text{Diện tích phần hợp}}$  [19]

- Ví dụ : Giả sử hộp giới hạn dự đoán và hộp giới hạn thực tế có diện tích phần giao là 50 và diện tích phần hợp là 150, thì IoU là  $\frac{50}{150} = 0.33.$

### 2.3.7 AP

- Khái niệm:

- AP là một thước đo hiệu suất tổng quát của mô hình phát hiện đối tượng, tính toán diện tích dưới đường cong Precision-Recall cho từng lớp.

- Công thức: AP được tính bằng cách vẽ đường cong Precision-Recall và tính diện tích dưới đường cong này (AUC-PR).
- Ví dụ : Nếu đường cong Precision-Recall của một lớp cho kết quả diện tích dưới đường cong là 0.75, thì AP của lớp đó là 0.75.

### 2.3.8 mAP

- Khái niệm:
  - mAP là trung bình của AP qua tất cả các lớp trong bài toán phát hiện đối tượng.
- Công thức:  $mAP = \frac{1}{N} \sum_{i=1}^N AP_i$  [19]
  - Trong đó  $N$  là số lượng lớp và  $AP_i$  là giá trị AP của lớp  $i$ .
- Ví dụ: Nếu bài toán có 3 lớp với AP lần lượt là 0.8, 0.7, và 0.9 thì mAP là
  - $\frac{0.8+0.7+0.9}{3} = 0.8$

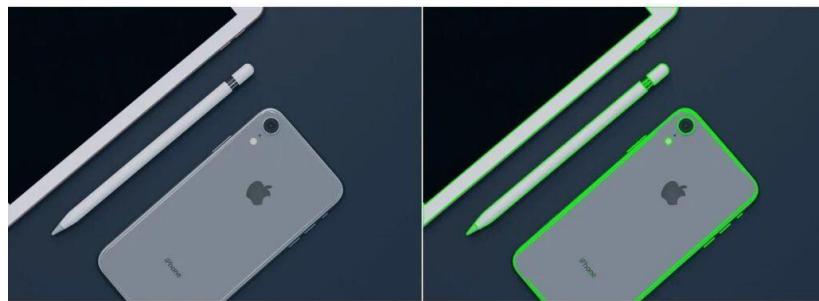
## Chương 3. Các công trình liên quan

### 3.1 Object Detection (Phát hiện đối tượng)

#### 3.1.1 Một số phương pháp Machine Learning

##### 3.1.1.1 Biên của đối tượng

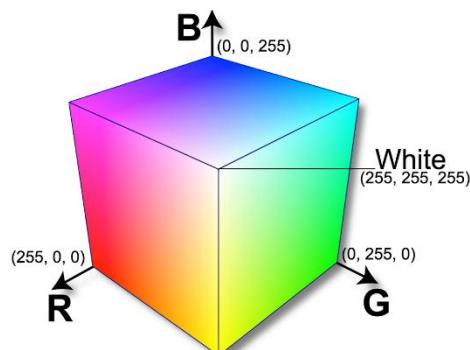
- Khái niệm:
  - Biên của đối tượng (Contours) có thể được giải thích đơn giản là một đường cong nối tất cả các điểm liên tục (đọc theo đường biên), có cùng màu sắc hoặc cường độ. Các đường biên (Contours) là một công cụ hữu ích để phân tích hình dạng và phát hiện và nhận dạng đối tượng [1, 23].
- Tìm biên của đối tượng:
  - Trong bài hướng dẫn của OpenCV phương pháp phổ biến nhất để tìm biên của đối tượng (Contours) đó là sử dụng hàm ‘findContours()’ nó hoạt động tốt nhất trên ảnh nhị phân vì vậy trước tiên sử dụng chúng ta nên chuyển ảnh sang ảnh xám và áp dụng các kỹ thuật ngưỡng, cạnh Sobel, v.v [1, 23].
- Vẽ biên của đối tượng:
  - Để vẽ ta sẽ sử dụng hàm ‘drawContours’. Hàm này cũng có thể được dùng để vẽ bất kỳ hình dạng nào nếu ta có các điểm ranh giới của nó. Tham số đầu tiên là ảnh gốc, tham số thứ hai là các đường biên được truyền dưới dạng Python list, tham số thứ ba là chỉ số của đường biên (sẽ hữu ích khi vẽ từng đường biên riêng lẻ. Để vẽ tất cả các đường viền, ta truyền giá trị là -1) và các tham số còn lại là màu sắc, độ dày, v.v [23].



Hình 15. Hình ban đầu và hình ảnh với các đường biên được vẽ lên

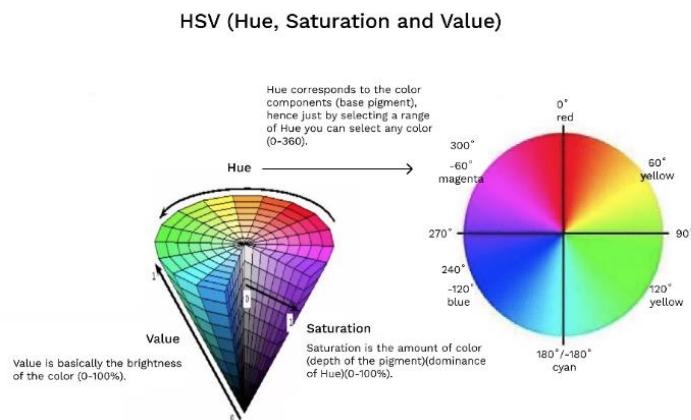
### 3.1.1.2 Không gian màu RGB, HSV

- Khái niệm chung: Không gian màu là một biểu diễn toán học của một tập hợp các màu. Có nhiều không gian màu khác nhau nhưng trong thị giác máy tính những không gian màu được sử dụng phổ biến nhất đó là RGB, HSV và Grayscale nhưng ở đây ta sẽ chỉ nói rõ về RGB và HSV [4].
- RGB: Là viết tắt của Red, Green, Blue nghĩa là đỏ, xanh lục, xanh lam. Trong không gian màu RGB, mỗi màu được thể hiện bằng sự kết hợp của ba màu cơ bản đỏ, lục và lam. Các giá trị cho mỗi kênh màu nằm trong khoảng từ 0 đến 255. Ví dụ như màu đỏ được biểu thị bằng các giá trị (255, 0, 0) trong khi màu xanh lục sẽ được biểu thị bằng các giá trị (0, 255, 0) [2].
- Với OpenCV thì mặc định của ảnh màu trong OpenCV là BGR không phải RGB, do đó khi ta đọc ảnh bằng hàm cv2.imread(), màu sắc được lưu trữ theo thứ tự Blue, Green, và Red.



Hình 16. Không gian màu RGB

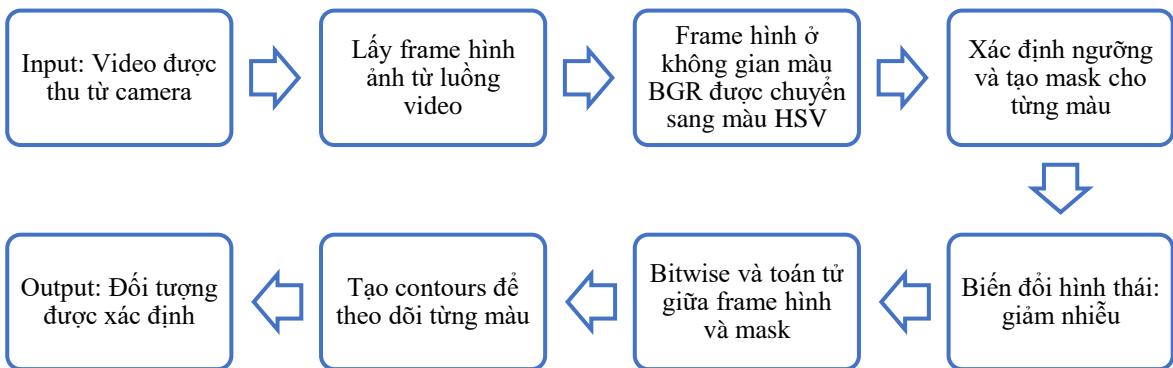
- HSV: Là viết tắt của các từ Hue, Saturation, Value và nó là một cách thể hiện thay thế của không gian màu RGB nó được tạo ra vào năm 1970. Không gian màu HSV tách thông tin màu thành ba thành phần: màu sắc, độ bão hòa và giá trị. Thành phần màu sắc đại diện cho chính màu đó, thành phần bão hòa đại diện cho độ tinh khiết của màu và thành phần giá trị đại diện cho độ sáng của màu [2].
- Với OpenCV việc chuyển đổi Chuyển đổi từ BGR sang HSV sẽ sử dụng câu lệnh ‘cv2.cvtColor(image, cv2.COLOR\_BGR2HSV)’.



Hình 17. Không gian màu HSV

#### 3.1.1.3 Thuật toán phát hiện đối tượng dựa trên màu

Thuật toán Phát hiện đối tượng dựa trên màu sắc [2, 3] là một phương pháp hiệu quả và thường được sử dụng trong các tình huống mà màu sắc của đối tượng có thể được dự đoán trước hoặc có đặc tính đặc trưng rõ ràng. Phương pháp này tận dụng lợi thế của không gian màu HSV (Hue, Saturation, Value) trong đó màu sắc được biểu thị qua thành phần Hue, cho phép chúng ta dễ dàng phân biệt và lọc ra các đối tượng dựa trên màu sắc trong một loạt điều kiện ánh sáng và môi trường.



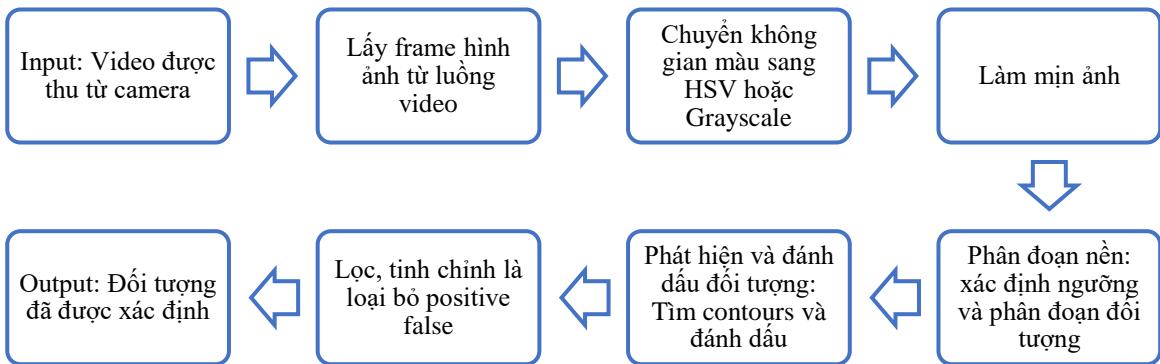
Biểu đồ 1. Workflow của thuật toán phát hiện màu

Mô tả: Biểu đồ 1 trên cho thấy các bước thực hiện thuật toán phát hiện đối tượng dựa trên màu.

### 3.1.1.4 Thuật toán phát hiện đối tượng dựa trên nền đồng nhất

Thuật toán phát hiện đối tượng trên nền đồng nhất (homogeneous background) là một phương pháp trong xử lý ảnh và thị giác máy tính, dùng để nhận diện các đối tượng nổi bật so với nền không phức tạp hoặc đơn sắc của ảnh. Đây là kỹ thuật phổ biến khi ta có một nền tương đối đơn giản và cần tách biệt các đối tượng cụ thể ra khỏi nền này.

Thuật toán này hoạt động dựa trên nguyên tắc phát hiện sự khác biệt giữa đối tượng và nền, thường qua các thuộc tính như màu sắc, độ sáng hoặc kết cấu. Bởi vì nền đồng nhất, bất kỳ sự khác biệt nào từ mức trung bình chung có thể được coi là một phần của đối tượng.



Biểu đồ 2. Workflow của thuật toán phát hiện đối tượng dựa trên nền đồng nhất

Mô tả: Biểu đồ 2 trên cho thấy các bước thực hiện thuật toán phát hiện đối tượng dựa trên nền đồng nhất.

### 3.1.1.5 Thuật toán phát hiện đối tượng dựa trên việc trừ nền

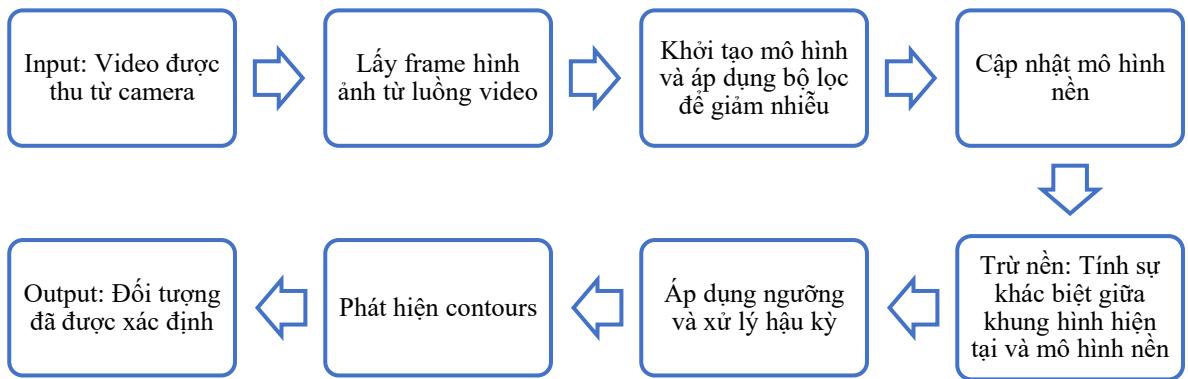
Giải thuật Background Subtraction [5, 6] là một kỹ thuật phổ biến trong xử lý ảnh và thị giác máy tính, đặc biệt là trong các ứng dụng giám sát video, để phát hiện các đối tượng động trong một chuỗi video mà nền của nó tương đối bất biến. Phương pháp này dựa trên việc loại bỏ nền tĩnh từ các khung hình hiện tại để phát hiện sự thay đổi hay chuyển động.

Thuật toán này so sánh từng khung hình của video với một mô hình nền đã biết hoặc được cập nhật liên tục để phát hiện các khu vực có sự thay đổi. Điều này cho phép phát hiện các đối tượng di chuyển như người, xe cộ hoặc bất kỳ vật thể nào khác mà xuất hiện hoặc di chuyển trong khung hình. Điểm yếu của thuật toán này là nếu đối tượng ở trạng thái tĩnh lâu thì có thể bị nhầm là nền và phát hiện sai [5, 6].

Trong OpenCV có 3 thuật toán để thực hiện việc trừ nền đó là:

- **BackgroundSubtractorMOG:** Đây là Thuật toán phân đoạn nền/tiền cảnh dựa trên hỗn hợp Gaussian.

- BackgroundSubtractorMOG2 - Đây cũng là Thuật toán phân đoạn nền/tiền cảnh dựa trên hỗn hợp Gaussian. Nó cung cấp khả năng thích ứng tốt hơn với các cảnh khác nhau do thay đổi ánh sáng, v.v.
- BackgroundSubtractorGMG – Thuật toán này kết hợp ước tính hình ảnh nền thông kê và phân đoạn Bayesian trên mỗi pixel.



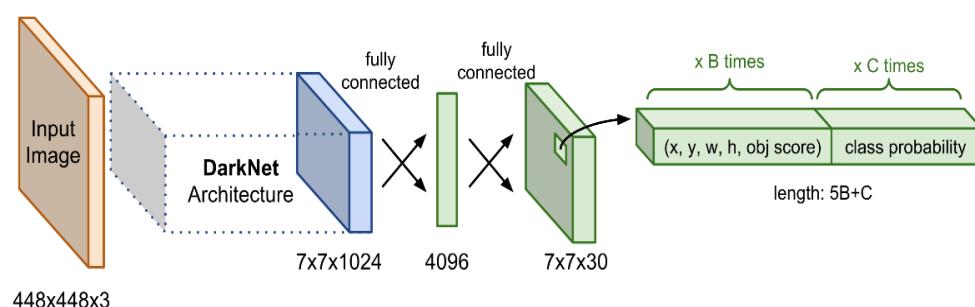
### 3.1.2 Phương pháp YOLO (Deep Learning)

#### 3.1.2.1 Tổng quan về mô hình YOLO

YOLO (You Only Look Once) là một trong những mô hình phát hiện đối tượng theo thời gian thực nổi bật nhất. YOLO khác với các mô hình phát hiện đối tượng trước đó như R-CNN hay Fast R-CNN ở chỗ nó xử lý toàn bộ ảnh chỉ trong một lần duy nhất (single forward pass) thay vì phải qua nhiều bước hoặc nhiều mô hình khác nhau [8, 10, 9].

Kiến trúc tổng quát của YOLO [8, 10, 9] có thể chia thành ba phần chính:

- Backbone: Đây là phần mạng nơ-ron sâu chịu trách nhiệm trích xuất các đặc trưng từ ảnh đầu vào. Backbone của YOLO ban đầu sử dụng các mạng CNN đơn giản, nhưng qua các phiên bản sau, nó được cải tiến với các kiến trúc như Darknet-19, Darknet-53.
- Neck: Phần này bao gồm các lớp nối tiếp nhau, chịu trách nhiệm tổng hợp các đặc trưng trích xuất được từ các mức khác nhau của backbone. Các kỹ thuật phổ biến bao gồm Feature Pyramid Network (FPN), Path Aggregation Network (PANet).
- Head: Đây là phần cuối cùng của mô hình, nơi dự đoán các bounding boxes và xác suất cho các lớp đối tượng. Đầu ra của YOLO là một tensor chứa các thông tin về vị trí ( $x, y, w, h$ ) và xác suất.



Hình 18.Kiến trúc YOLOv1

Quy trình hoạt động của YOLO [8, 10, 9]:

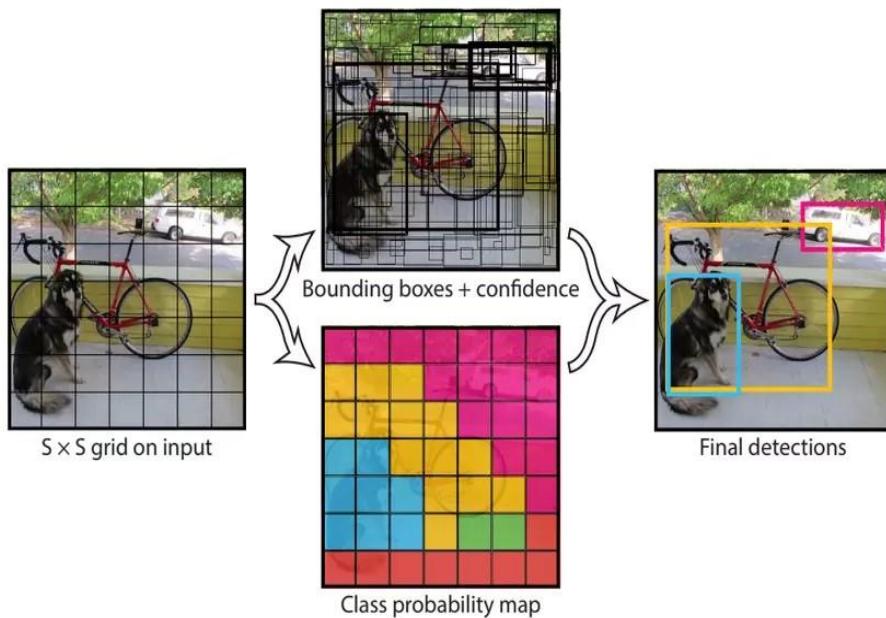
- Chia ảnh thành lưới (Grid Division): YOLO chia ảnh đầu vào thành một lưới  $S \times S$ . Mỗi ô lưới sẽ chịu trách nhiệm phát hiện các đối tượng nằm bên trong nó. Kích thước của lưới  $S \times S$  thường là một giá trị cố định 7x7 hoặc 13x13 tùy thuộc vào kích thước ảnh đầu vào và phiên bản YOLO.
  - Ví dụ: Đối với ảnh có kích thước 448x448, nếu lưới có kích thước 7x7 thì mỗi ô lưới sẽ có kích thước khoảng 64x64 pixels.
- Dự đoán Bounding Boxes và Confidence Scores: Mỗi ô lưới dự đoán  $B$  bounding boxes, mỗi bounding box sẽ được xác định bởi 5 tham số:
  - $x$  và  $y$ : Tọa độ tâm của bounding box so với ô lưới hiện tại.
  - $w$  và  $h$ : Chiều rộng và chiều cao của bounding box được chuẩn hóa theo chiều rộng và chiều cao của toàn bộ ảnh.
  - $c$ : Độ tin cậy (confident score) của bounding box cho biết xác suất bounding box có chứa một đối tượng và mức độ chính xác của bounding box.
  - Ví dụ: Với  $B = 2$ , mỗi ô lưới sẽ dự đoán 2 bounding boxes, và mỗi bounding box sẽ có 5 giá trị ( $x, y, w, h, c$ ).
- Dự đoán các lớp đối tượng (Class Predictions): Mỗi ô lưới cũng dự đoán xác suất cho  $C$  lớp đối tượng khác nhau ví dụ: (chó, mèo, xe hơi, v.v.). Các xác suất này cho biết đối tượng trong bounding box thuộc về lớp nào.
  - Ví dụ: Nếu có 20 lớp đối tượng, mỗi ô lưới sẽ dự đoán 20 giá trị xác suất tương ứng với 20 lớp.
- Tính toán Output Tensor: Đầu ra của mô hình YOLO là một tensor có kích thước  $S \times S \times (B \times 5 + C)$ . Mỗi ô lưới sẽ chứa thông tin về  $B$  bounding boxes và  $C$  lớp đối tượng.
  - Ví dụ: Với  $S=7, B=2, C=20$  thì đầu ra sẽ có kích thước là  $7 \times 7 \times (2 \times 5 + 20) = 7 \times 7 \times 30$ .

- Loss Function: YOLO sử dụng một hàm mất mát (loss function) để huấn luyện mô hình bao gồm ba thành phần chính:
  - Localization Loss: Đo lường độ lệch giữa bounding box dự đoán và bounding box thật.
  - Confidence Loss: Đo lường độ tin cậy của bounding box dự đoán.
  - Classification Loss: Đo lường độ chính xác của dự đoán lớp đối tượng.
- Post-processing: Non-Maximum Suppression (NMS) [8, 10, 9]: Sau khi có các dự đoán từ mô hình, YOLO sử dụng một kỹ thuật gọi là Non-Maximum Suppression (NMS) để lọc ra các bounding boxes không cần thiết và giữ lại những bounding boxes có độ tin cậy cao nhất.
  - Bước 1: Sắp xếp các bounding boxes theo thứ tự giảm dần của độ tin cậy.
  - Bước 2: Chọn bounding box có độ tin cậy cao nhất và so sánh với các bounding boxes khác. Nếu tỷ lệ giao nhau trên hợp (IoU) của bounding box này với các bounding boxes khác lớn hơn một ngưỡng (thường là 0.5), loại bỏ các bounding boxes đó.

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


Hình 19. Hàm tính IoU

- Bước 3: Lặp lại quá trình cho đến khi không còn bounding boxes nào cần kiểm tra.



Hình 20. Quy trình hoạt động của YOLO

- Kết luận: YOLO là một mô hình phát hiện đối tượng mạnh mẽ với nhiều cải tiến qua các phiên bản. Nó đã và đang là nền tảng cho nhiều ứng dụng thực tế trong các lĩnh vực khác nhau, từ xe tự lái, robot đến giám sát video và y tế.

### 3.1.2.2 Sự tiến hóa của YOLO từ v1 đến v7

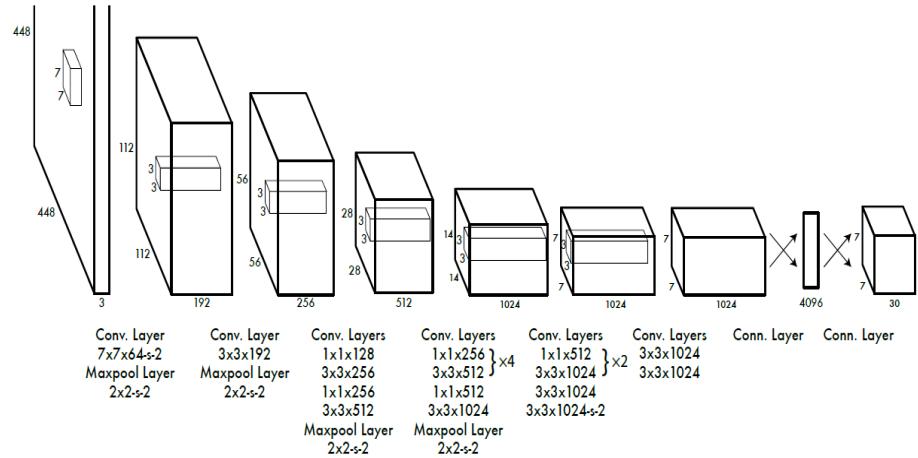


Hình 21. Quá trình phát triển của YOLO từ v1 đến v7

YOLO đã trải qua nhiều phiên bản cải tiến từ khi được giới thiệu lần đầu tiên vào năm 2016. Mỗi phiên bản mới mang lại những cải tiến về độ chính xác, tốc độ, và khả năng phát hiện đối tượng [8]. Dưới đây là quá trình tiến hóa từ YOLOv1 đến YOLOv7:

- YOLOv1 (2016)
  - Giới thiệu: YOLOv1 [8, 10] được giới thiệu bởi Joseph Redmon vào năm 2016.
  - Kiến trúc:
    - YOLOv1 sử dụng một kiến trúc CNN đơn giản với 24 lớp convolution và 2 lớp fully connected. Nó chia ảnh đầu vào thành lưới  $S \times S$  và dự đoán bounding boxes và lớp đối tượng cho mỗi ô lưới.
    - Kích thước ảnh đầu vào: 448x448x3 (RGB image).

- Các lớp convolution và maxpooling để trích xuất đặc trưng với kích thước đặc trưng giảm dần từ  $112 \times 112 \times 64$  đến  $7 \times 7 \times 1024$ .
- Flatten đầu ra từ convolution thành một vector và qua hai lớp fully connected để dự đoán bounding boxes và xác suất lớp.
- Output: Tensor  $7 \times 7 \times 30$ , mỗi ô lưới dự đoán 2 bounding boxes và xác suất của 20 lớp đối tượng



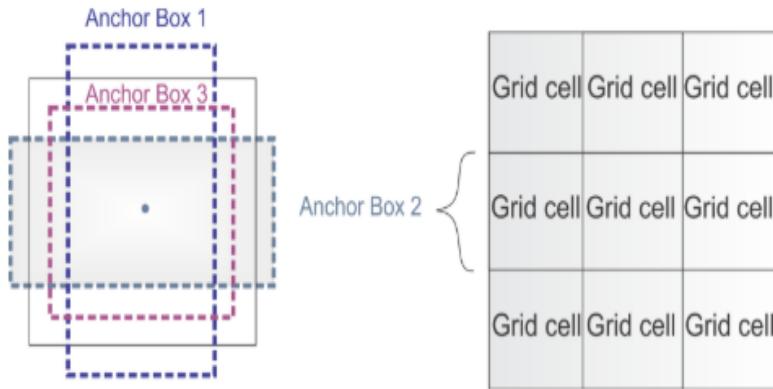
Hình 22. Kiến trúc YOLOv1

○ Đặc điểm:

- Phát hiện đối tượng trong 1 lần duy nhất (single pass).
- Tốc độ nhanh nhưng độ chính xác chưa tốt thấp hơn so với các mô hình như Fast R-CNN.
- Hạn chế: Khó khăn trong việc phát hiện các đối tượng nhỏ và đối tượng gần nhau.
- Hiệu suất: YOLOv1 đạt được độ chính xác trung bình (AP) là 63.4% trên tập dữ liệu PASCAL VOC2007.

- YOLOv2 (YOLO9000, 2017)

- Giới thiệu: YOLOv2 [8, 10] hay còn được gọi là YOLO9000 được giới thiệu vào năm 2017.
- Kiến trúc: YOLOv2 sử dụng kiến trúc Darknet-19 làm backbone để trích xuất các đặc trưng từ ảnh đầu vào. Darknet-19 bao gồm 19 lớp convolution và 5 lớp max-pooling.
- Kích thước ảnh đầu vào: 416x416x3 (RGB image).
- Fine-Grained Features: Kết hợp đặc trưng từ các lớp nông và sâu để cải thiện khả năng phát hiện đối tượng nhỏ.
- Bounding Boxes và Lớp Đôi Tượng: Sử dụng anchor boxes và các đặc trưng từ backbone để dự đoán bounding boxes và xác suất lớp.
- Loss Function: Bao gồm localization loss, confidence loss và class probability loss.
- Cải tiến:
  - Batch Normalization: Giảm overfitting và cải thiện tốc độ huấn luyện.
  - High-Resolution Classifier: Sử dụng các ảnh có độ phân giải cao hơn (448x448) để cải thiện độ chính xác.
  - Anchor Boxes: Sẽ sử dụng các anchor boxes để dự đoán ra bounding boxes. Giúp cải thiện khả năng phát hiện đối tượng.

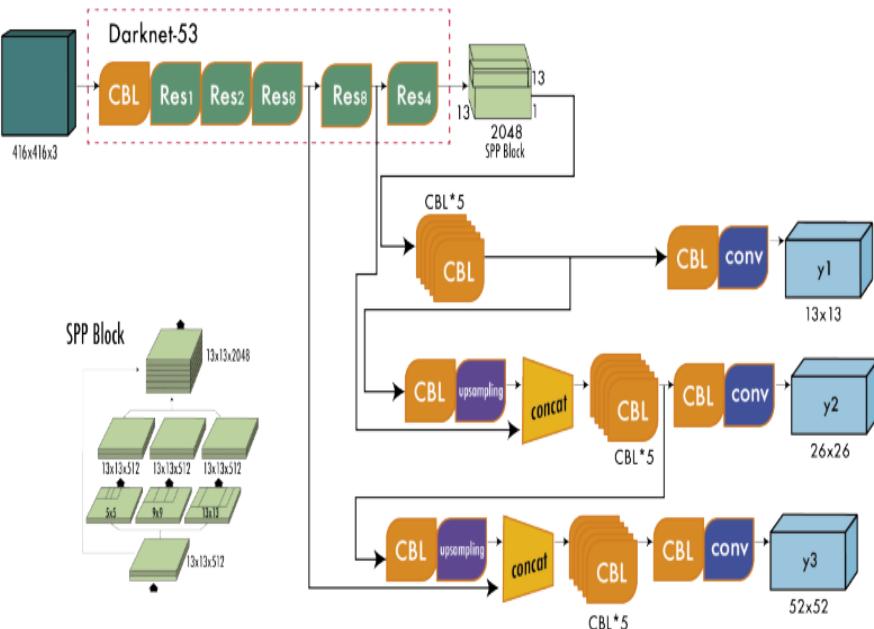


Hình 23. YOLOv2 xác định nhiều anchor boxes cho mỗi lưới

- Dimension Clusters: Sử dụng k-means clustering để tìm ra các kích thước anchor tốt nhất.
- Direct location prediction: Dùng để dự đoán trực tiếp tọa độ bounding box.
- Finer-grained Features: Loại bỏ một lớp pooling để có được bản đồ đặc trưng chi tiết hơn.
- Multi-Scale Training: Huấn luyện mô hình với các kích thước đầu vào khác nhau để cải thiện khả năng phát hiện đối tượng.
- Đặc điểm:
  - Cải thiện đáng kể độ chính xác và khả năng phát hiện đối tượng nhỏ.
  - Có thể phát hiện hơn 9000 lớp đối tượng bằng cách huấn luyện kết hợp với dữ liệu từ ImageNet và COCO.
- Hiệu suất: YOLOv2 đạt được AP là 78.6% trên tập dữ liệu PASCAL VOC2007.

- YOLOv3 (2018)

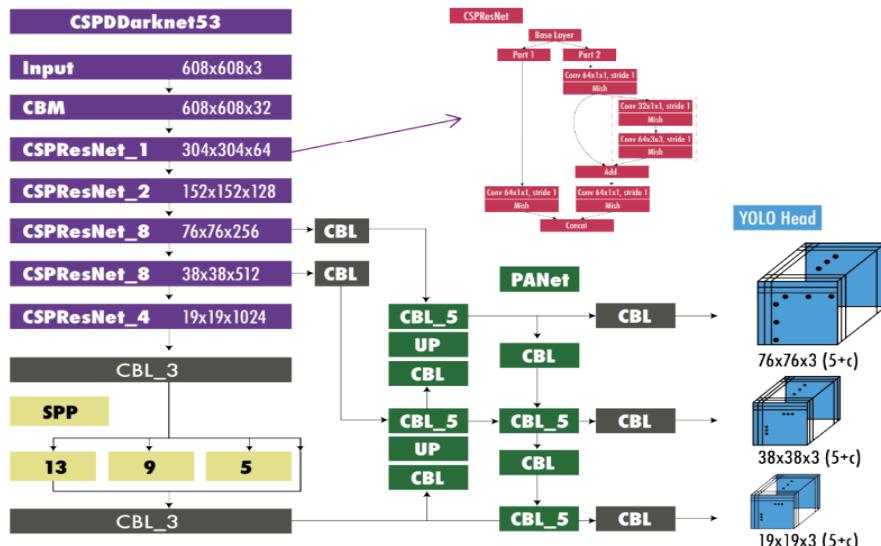
- Giới thiệu: YOLOv3 [8, 10] được giới thiệu vào năm 2018 bởi Joseph Redmon và Ali Farhadi.
- Kiến trúc: Sử dụng kiến trúc Darknet-53 để làm backbone với 53 lớp convolution và sử dụng residual connections tương tự như ResNet.
- Kích thước ảnh đầu vào: 416x416x3 (RGB image).
- Neck: YOLOv3 sử dụng các lớp convolution và upsampling để kết hợp các đặc trưng từ nhiều mức khác nhau của backbone (Darknet-53), thực hiện dự đoán ở ba mức độ phân giải khác nhau (13x13, 26x26, 52x52).
- Head: Mỗi Head thực hiện dự đoán bounding boxes và xác suất lớp đối tượng. Sử dụng logistic regression cho tọa độ và objectness score, binary cross-entropy loss cho xác suất lớp.
- Loss Function: Bao gồm localization loss, confidence loss và class probability loss.



Hình 24. Kiến trúc YOLOv3

- Cải tiến:
  - Sử dụng logistic regression để dự đoán objectness score cho mỗi bounding box, giúp đơn giản hóa việc dự đoán.
  - Multi-scale predictions: Dự đoán bounding boxes ở ba mức lưới khác nhau (13x13, 26x26, và 52x52.) cải thiện khả năng phát hiện đối tượng nhỏ.
  - Class Prediction: Sử dụng binary cross-entropy cho việc phân loại lớp, cho phép mô hình phân loại nhiều lớp cho cùng một bounding box.
  - YOLOv3 sử dụng các anchor boxes tương tự như YOLOv2, nhưng với các kích thước khác nhau để phù hợp với ba mức độ phân giải khác nhau.
- Đặc điểm: Tốc độ nhanh và độ chính xác cao phù hợp với những ứng dụng thời gian thật.
- Hiệu suất: YOLOv3 đạt được AP là 36.2% và AP50 là 60.6% trên tập dữ liệu COCO với tốc độ 20 FPS
- YOLOv4 (2020)
  - Giới thiệu: YOLOv4 [8, 10] được giới thiệu vào năm 2020 bởi Alexey Bochkovskiy, Chien-Yao Wang, và Hong-Yuan Mark Liao.
  - Kiến trúc:
    - Backbone: Sử dụng kiến trúc CSPDarknet53 làm backbone để trích xuất các đặc trưng từ ảnh đầu vào. CSPDarknet53 kết hợp các lớp convolution với CSPNet để tối ưu hóa việc trích xuất đặc trưng và đồng thời để giảm số lượng tính toán mà không làm giảm độ chính xác.

- Neck: YOLOv4 sử dụng PAN để tổng hợp các đặc trưng từ nhiều mức khác nhau của backbone, giúp cải thiện khả năng phát hiện đối tượng.
  - Feature Pyramid Network (FPN): Tổng hợp các đặc trưng từ nhiều mức khác nhau của backbone.
  - Path Aggregation Network (PAN): Kết hợp các đặc trưng từ nhiều mức khác nhau để tăng cường khả năng tổng hợp thông tin.
- Head: Thực hiện dự đoán ở ba mức độ phân giải khác nhau để cải thiện khả năng phát hiện đối tượng ở các kích thước khác nhau ((13x13, 26x26, và 52x52)).
- Kích thước ảnh đầu vào: 416x416x3 (RGB image).
- Bounding Boxes và Lớp Đối Tượng: Sử dụng anchor boxes và các đặc trưng từ backbone để dự đoán bounding boxes và xác suất lớp.
- Loss Function: Bao gồm CIoU loss, confidence loss và class probability loss.

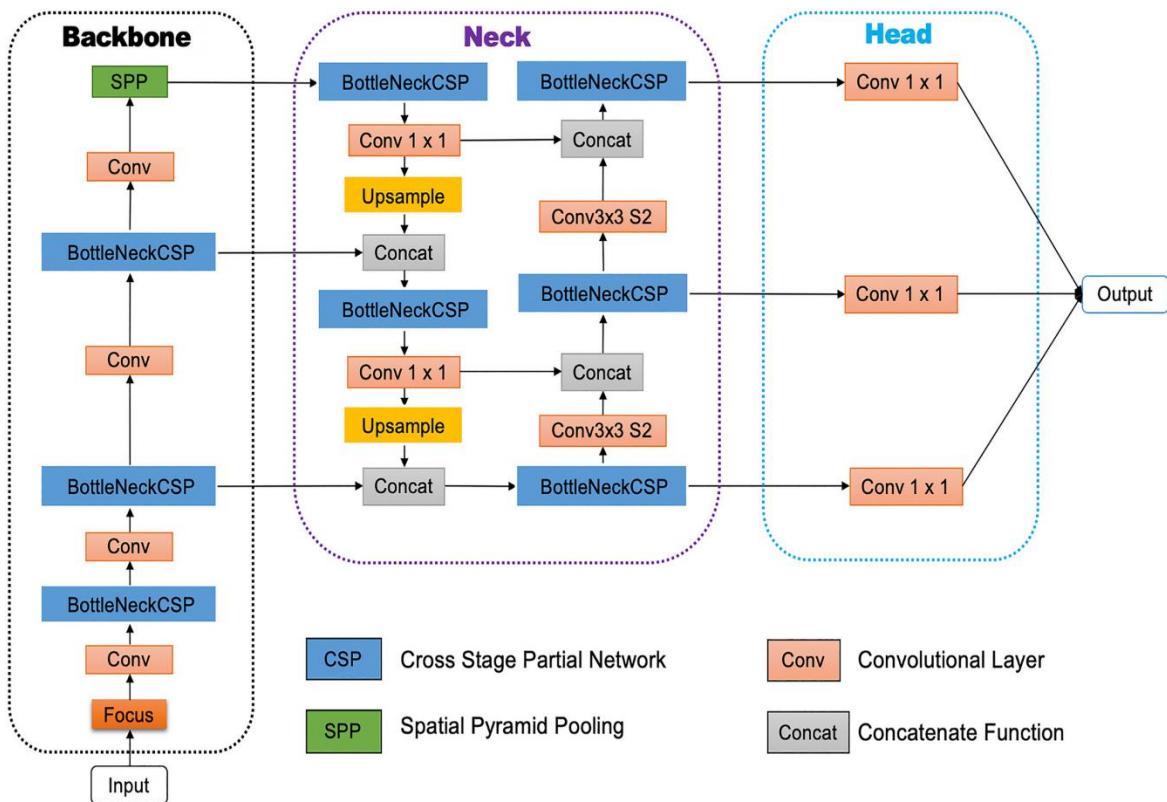


Hình 25. Kiến trúc YOLOv4

- Cải tiến:
  - Backbone: Dùng CSPDarknet53 (Cross Stage Partial Networks) để giảm số lượng tính toán mà không làm giảm độ chính xác.
  - Bag of Freebies (BoF): Các kỹ thuật cải tiến huấn luyện mà không ảnh hưởng đến tốc độ suy luận, như Mosaic data augmentation, DropBlock regularization, và CIoU loss.
  - Bag of Specials (BoS): Các kỹ thuật cải thiện độ chính xác mà chỉ ảnh hưởng nhẹ đến tốc độ suy luận, như Mish activation, PANet path aggregation, và SPP (Spatial Pyramid Pooling).
  - Anchor Boxes: Sử dụng k-means clustering để chọn các anchor boxes tốt nhất.
  - CIoU Loss: Sử dụng Complete Intersection over Union (CIoU) loss để cải thiện dự đoán bounding boxes.
- Đặc điểm: Đạt hiệu suất cao trên COCO dataset với độ chính xác và tốc độ cải thiện.
- Hiệu suất: YOLOv4 đạt được AP là 43.5% và AP50 là 65.7% trên tập dữ liệu COCO với tốc độ hơn 50 FPS trên NVIDIA V100.
- YOLOv5 (2020)
  - Giới thiệu: YOLOv5 [8, 10] được phát triển và phát hành bởi Ultralytics vào năm 2020
  - Kiến trúc:
    - YOLOv5 được thiết kế để nhẹ hơn và dễ triển khai hơn so với các phiên bản trước. Nó có thể chạy trên nhiều loại phần cứng khác nhau từ GPU cao cấp đến CPU và thậm chí trên các thiết bị di động.

- Kích thước ảnh đầu vào: 640x640x3 (RGB image).
- Backbone: Sử dụng một kiến trúc CSP-DarkNet53 để trích xuất các đặc trưng từ ảnh đầu vào. Backbone của YOLOv5 có thể thay đổi tùy theo phiên bản (s, m, l, x) với số lượng lớp và độ phức tạp khác nhau. Các phiên bản nhỏ hơn (s) sử dụng ít lớp hơn và nhẹ hơn, trong khi các phiên bản lớn hơn (x) sử dụng nhiều lớp hơn và mạnh mẽ hơn. Sử dụng Focus layer và các lớp convolution với CSPNet để trích xuất đặc trưng.
- Neck: Sử dụng sử dụng hai thành phần chính là SPPF và CSP-PANet để tổng hợp các đặc trưng từ nhiều mức khác nhau của backbone, giúp cải thiện khả năng phát hiện đối tượng.
  - Path Aggregation Network (PAN): Kết hợp các đặc trưng từ nhiều mức khác nhau để tăng cường khả năng tổng hợp thông tin. Kích thước của output lần lượt là (80x80x256, 40x40x512, 20x20x1024.)
  - Feature Pyramid Network (FPN): Tổng hợp các đặc trưng từ nhiều mức khác nhau của backbone kích thước output lần lượt là (80x80x256, 40x40x512, 20x20x1024.)
  - SPPF (Spatial Pyramid Pooling-Fast): là một lớp pooling đặc biệt giúp tổng hợp thông tin không gian từ nhiều vùng khác nhau của ảnh. Nó giúp tăng cường khả năng tổng hợp thông tin không gian và tăng tốc độ tính toán. Sử dụng các kernel pooling với kích thước khác nhau (ví dụ: 1x1, 3x3, 5x5, 7x7) để trích xuất thông tin từ các vùng khác nhau của ảnh, sau đó kết hợp các đặc trưng này lại.

- **CSP-PANet:** CSP-PANet kết hợp các đặc trưng từ nhiều mức khác nhau của backbone, giúp cải thiện khả năng tổng hợp thông tin và dự đoán. Sử dụng các lớp convolution và upsample để kết hợp các đặc trưng từ nhiều mức độ phân giải khác nhau. Kết hợp thông tin từ các mức độ phân giải cao (fine-grained features) và mức độ phân giải thấp (high-level features).
- **Head:** Head của YOLOv5 thực hiện dự đoán ở ba mức độ phân giải khác nhau để cải thiện khả năng phát hiện đối tượng ở các kích thước khác nhau ((20x20, 40x40, và 80x80)).



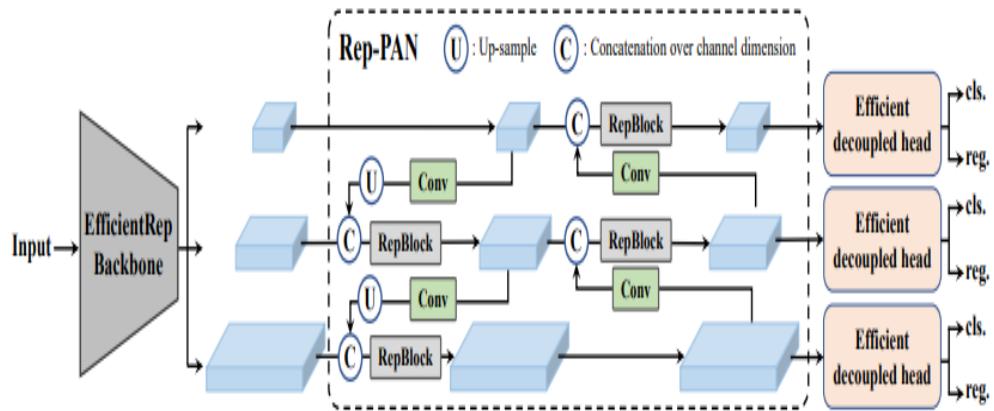
Hình 26. Kiến trúc YOLOv5

- Cải tiến:
  - PyTorch Implementation: Sử dụng framework PyTorch thay vì Darknet, giúp dễ dàng hơn trong việc phát triển và triển khai.
  - AutoAnchor: Tự động điều chỉnh anchor boxes để phù hợp với dữ liệu huấn luyện.
  - Loss Function: YOLOv5 sử dụng CIoU (Complete Intersection over Union) loss và BCE (Binary Cross-Entropy) cho các dự đoán bounding boxes và lớp đối tượng.
  - Scaled Models: Cung cấp nhiều phiên bản mô hình với kích thước khác nhau (nano, small, medium, large, extra-large) để phù hợp với các yêu cầu tài nguyên khác nhau.
  - Anchor Boxes và Grid Cell: Sử dụng các anchor boxes tương tự như các phiên bản trước nhưng cải tiến để tối ưu hóa việc dự đoán.
  - Augmentations: Sử dụng các kỹ thuật tăng cường dữ liệu tiên tiến như Mosaic, Copy-Paste, và MixUp.
- Đặc điểm: Dễ sử dụng, hiệu xuất cao và tích hợp tốt với các công cụ thư viện hiện có.
- Hiệu suất: YOLOv5 đạt được AP là 50.7% với kích thước ảnh 640 pixels trên tập dữ liệu COCO với tốc độ 200 FPS trên NVIDIA V100.
- YOLOR (2021)
  - Giới thiệu: YOLOR [8, 10] (You Only Learn One Representation) được phát triển bởi các tác giả của YOLOv4 và giới thiệu vào năm 2021.

- Kiến trúc: YOLOR kết hợp các khái niệm từ YOLO với các phương pháp học tập tiên tiến, như học tập implicit và explicit.
- Cải tiến:
  - Implicit Knowledge: YOLOR kết hợp cả implicit knowledge (kiến thức ngầm) và explicit knowledge (kiến thức rõ ràng) trong quá trình huấn luyện.
  - Unified Network: Mạng nơ-ron của YOLOR có thể xử lý cả các nhiệm vụ phát hiện đối tượng và phân loại trong cùng một mô hình duy nhất.
  - Efficient Representation: Sử dụng các biểu diễn hiệu quả để giảm số lượng tham số và tăng tốc độ suy luận.
- Đặc điểm:
  - Khả năng phát hiện đối tượng với độ chính xác cao và tốc độ nhanh.
  - Tích hợp khả năng học tập implicit và explicit giúp cải thiện hiệu suất tổng thể của mô hình.
- Hiệu suất: YOLOR đạt được AP là 55.6% trên tập dữ liệu COCO với tốc độ 42 FPS trên NVIDIA Tesla V100.
- YOLOX (2021)
  - Giới thiệu: YOLOX [8, 10] được phát triển bởi Megvii Research và giới thiệu vào năm 2021.
  - Kiến trúc: YOLOX giới thiệu một số cải tiến về thiết kế mạng và chiến lược huấn luyện.
  - Cải tiến:

- Anchor-free design sử dụng thiết kế không cần anchor để đơn giản hóa quá trình huấn luyện.
  - Decoupled Head: Phân tách các nhánh dự đoán cho localization và classification, giúp cải thiện hiệu suất.
  - Advanced Data Augmentation: Sử dụng các kỹ thuật tăng cường dữ liệu tiên tiến như Mosaic, MixUp.
  - IoU-aware Loss: Sử dụng hàm mất mát dựa trên IoU để cải thiện độ chính xác của dự đoán bounding box.
- Đặc điểm:
    - Cải thiện độ chính xác và tốc độ so với các phiên bản YOLO trước đó.
    - Thiết kế không cần anchor giúp đơn giản hóa quá trình huấn luyện và tăng cường khả năng tổng quát hóa của mô hình.
  - Hiệu suất: YOLOX đạt được AP là 51.5% trên tập dữ liệu COCO với tốc độ 68 FPS trên NVIDIA Tesla V100.
- YOLOv6 (2022)
    - Giới thiệu: YOLOv6 [8, 10] được phát triển bởi Meituan Vision AI và ra mắt vào năm 2022 và ra đời sau YOLOv7 hai tháng.
    - Kiến trúc:
      - Backbone: Sử dụng EfficientRep làm backbone, bao gồm các khối RepVGG và RepConv.
        - RepVGG: Được sử dụng trong quá trình huấn luyện, có các kết nối dư (skip connections).
        - RepConv: Được sử dụng trong quá trình suy luận, chỉ bao gồm các lớp convolution 3x3 đơn giản.

- Đối với các mô hình lớn hơn, YOLOv6 sử dụng CSPStackRep, một phiên bản cải tiến của CSPNet.
- Neck - Rep-PAN:
  - Neck của YOLOv6 sử dụng Path Aggregation Network (PAN) với các khối reparameterized (Rep-PAN).
  - Rep-PAN kết hợp các đặc trưng từ nhiều mức độ phân giải khác nhau, tương tự như PAN trong YOLOv4 và YOLOv5, nhưng sử dụng các khối RepBlock hoặc CSPStackRep.
- Head - Efficient Decoupled Head:
  - YOLOv6 sử dụng Efficient Decoupled Head, tách biệt các nhánh dự đoán objectness, classification và bounding box prediction, giúp giảm thiểu tính toán và tăng độ chính xác.
  - Điều này khác với các phiên bản trước đó như YOLOv4 và YOLOv5, nơi các nhánh này chia sẻ tham số.
- Anchor-Free:
  - Khác với các phiên bản YOLO trước đó, YOLOv6 không sử dụng anchor-based methods, mà chuyển sang anchor-free, giúp tăng tốc độ và giảm độ phức tạp.



Hình 27. Kiến trúc YOLOv6

- Cải tiến:

- Backbone: Sử dụng EfficientRep (dựa trên RepVGG) với độ song song cao hơn.
- Neck: Sử dụng PAN (Path Aggregation Network) được cải tiến với các khối Rep.
- Head: Sử dụng đầu decoupled head hiệu quả.
- Task-aligned Learning: Sử dụng phương pháp học tập điều chỉnh nhiệm vụ để giảm sự sai lệch giữa phân loại và định vị.
- Loss Functions: Sử dụng VariFocal loss cho phân loại và SIoU/GIoU loss cho định vị.
- Self-Distillation: Sử dụng kỹ thuật knowledge distillation để cải thiện độ chính xác mà không tăng chi phí tính toán.
- Longer Training Epochs: Một số mô hình YOLOv6 được huấn luyện trong 400 epochs thay vì 300 epochs, giúp mô hình hội tụ tốt hơn.

- Đặc điểm:
  - Hiệu suất cao hơn với tốc độ nhanh hơn và độ chính xác cao hơn so với các phiên bản trước đó.
- Hiệu suất: YOLOv6 đạt được AP là 57.2% với tốc độ 29 FPS trên NVIDIA Tesla T4.
- YOLOv7 (2022)
  - Giới thiệu: YOLOv7 [8, 10] mang lại nhiều cải tiến quan trọng trong việc phát hiện đối tượng theo thời gian thực. nó được phát triển bởi cùng nhóm tác giả của YOLOv4, YOLOR và nó được ra mắt vào năm 2022.
  - Kiến trúc:
    - Backbone: E-ELAN
      - E-ELAN: Sử dụng các khối OSA để tối ưu hóa việc tổng hợp đặc trưng.
      - CSPNet: Kết hợp CSPNet để cải thiện việc học đặc trưng.
    - Neck: PAN với RepConv
      - Path Aggregation Network (PAN): Tổng hợp các đặc trưng từ nhiều mức độ phân giải khác nhau.
      - RepConv: Dùng các khối re-parameterized convolution để tối ưu hóa hiệu suất.
    - Head: Decoupled Head
      - Decoupled Head: Tách biệt các nhánh dự đoán objectness, classification và bounding box prediction, giúp giảm thiểu tính toán và tăng độ chính xác.

- Loss Functions

- Varifocal Loss (VFL): Sử dụng cho classification loss, giúp cân bằng các mẫu dễ và khó trong quá trình huấn luyện.
- Distribution Focal Loss (DFL): Sử dụng cho box regression loss, giúp xử lý tốt hơn khi các biên của ground truth bị mờ.

- Cải tiến:

- Extended Efficient Layer Aggregation Network (E-ELAN): Giúp mô hình học và hội tụ hiệu quả hơn bằng cách kiểm soát gradient dài nhất.
  - Model Scaling: Chiến lược mở rộng mô hình để tạo ra các mô hình có kích thước khác nhau bằng cách điều chỉnh các thuộc tính của mô hình.
  - RepConvN: Sử dụng các lớp convolution được tái cấu trúc mà không có kết nối identity.
  - Batch Normalization Integration: Tích hợp batch normalization vào các lớp convolution để cải thiện hiệu suất.
  - Implicit Knowledge: Áp dụng kiến thức ngầm vào mô hình.
- Đặc điểm: Hiệu suất vượt trội về cả tốc độ và độ chính xác, phù hợp cho nhiều ứng dụng khác nhau.
  - Hiệu suất: YOLOv7 đạt được AP là 55.9% và AP50 là 73.5% với kích thước ảnh 1280 pixels trên tập dữ liệu COCO với tốc độ 50 FPS trên NVIDIA V100.

- Tổng kết

- Quá trình tiến hóa của YOLO từ YOLOv1 đến YOLOv7 đã chứng kiến nhiều cải tiến quan trọng, bao gồm các kỹ thuật mới trong thiết kế kiến trúc, các phương pháp tăng cường dữ liệu, và các chiến lược huấn luyện tiên tiến. Những cải tiến này đã giúp YOLO trở thành một trong những mô hình phát hiện đối tượng phổ biến và hiệu quả nhất trong lĩnh vực thị giác máy tính.

#### 3.1.2.3 Tổng quan về mô hình YOLOv8

- Giới thiệu về YOLOv8

- YOLOv8 là một trong những mô hình mới nhất của dòng mô hình YOLO được phát triển bởi Ultralytics và giới thiệu vào năm 2023. Nó được phát triển dựa trên kiến trúc của Yolov5 và mục tiêu của YOLOv8 là cải thiện cả về độ chính xác và tốc độ, đồng thời duy trì tính dễ sử dụng và khả năng triển khai thực tế và nó mang lại nhiều cải tiến trong kiến trúc và phương pháp huấn luyện. Các cải tiến này nhằm tăng hiệu suất tổng thể của mô hình, làm cho YOLOv8 trở thành một lựa chọn mạnh mẽ cho các bài toán phát hiện đối tượng [8, 10, 11].

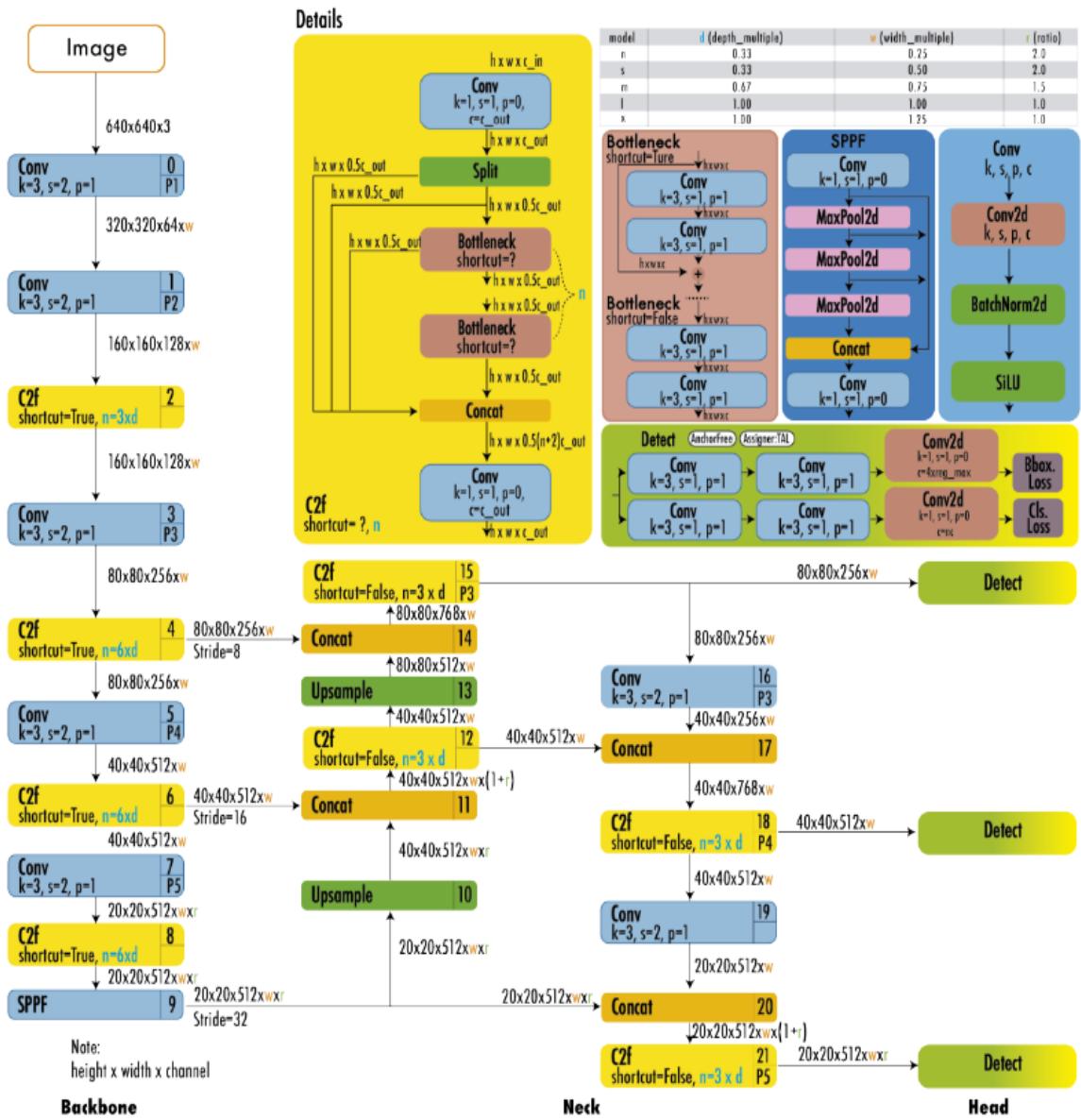
- Kiến trúc của YOLOv8

- Backbone: Là phần đầu tiên của mô hình và nó chịu trách nhiệm trích xuất đặc trưng từ ảnh đầu vào nó bao gồm các lớp convolution, C2f và SPPF.
- Cải tiến của backbone:
  - C2f (Cross Stage Partial with Two Convolutions): YOLOv8 sử dụng một biến thể khác của CSPNet nó bao gồm CSPNet và hai lớp convolution để giảm số lượng tính toán mà không làm giảm độ chính xác. C2f giúp mô hình xử lý thông tin hiệu quả hơn bằng cách chia và hợp nhất các đặc trưng qua nhiều giai đoạn.

- Anchor-free Design: Thiết kế không cần anchor giúp đơn giản hóa quá trình huấn luyện và cải thiện độ chính xác. Mô hình dự đoán trực tiếp vị trí của các bounding boxes mà không cần phải chọn trước các anchor boxes. Tránh các vấn đề liên quan đến lựa chọn anchor boxes và giúp mô hình dễ dàng tổng quát hóa hơn.
- Cấu trúc của backbone:
  - Conv Layers:
    - P1: 640x640 -> 320x320 (Conv, k=3, s=2, p=1)
    - P2: 320x320 -> 160x160 (Conv, k=3, s=2, p=1)
    - P3: 160x160 -> 80x80 (Conv, k=3, s=2, p=1)
    - P4: 80x80 -> 40x40 (Conv, k=3, s=2, p=1)
    - P5: 40x40 -> 20x20 (Conv, k=3, s=2, p=1)
  - C2f Layers:
    - P1, P2, P3, P4, P5: Các lớp C2f (shortcut=True, số lượng bottleneck khác nhau) giúp chia và hợp nhất các đặc trưng qua nhiều giai đoạn.
  - SPPF (Spatial Pyramid Pooling-Fast):
    - P5: Sử dụng SPPF để tổng hợp thông tin không gian từ nhiều vùng khác nhau của ảnh, kích thước đặc trưng từ 20x20 -> 20x20.
- Neck: là phần trung gian giữa backbone và head, giúp tổng hợp các đặc trưng từ nhiều mức khác nhau của backbone để chuẩn bị cho phần head xử lý.
- Cải tiến của neck:
  - PAN (Path Aggregation Network): PAN sẽ giúp tổng hợp các đặc trưng từ các mức khác nhau của backbone và cải thiện khả năng tổng hợp thông tin. Tăng cường khả năng phát hiện đối tượng ở các kích thước khác nhau và tăng cường độ chính xác.

- Upsample: Là quá trình tăng kích thước của đặc trưng để phù hợp với kích thước của các đặc trưng từ các mức khác nhau trong backbone. Giúp kết hợp thông tin từ các mức độ phân giải khác nhau, tăng cường khả năng nhận diện đối tượng.
  - Concatenate (Concat): Là quá trình kết hợp các đặc trưng từ nhiều nguồn khác nhau để tạo ra một đặc trưng tổng hợp. Sẽ tổng hợp thông tin từ nhiều nguồn khác nhau, giúp mô hình hiểu rõ hơn về các đối tượng trong ảnh.
- Cấu trúc của neck:
    - Từ P5 -> P4: Upsample đặc trưng từ P5 (20x20 -> 40x40) và Concat với đặc trưng từ P4.
    - Từ P4 -> P3: Upsample đặc trưng từ P4 (40x40 -> 80x80) và Concat với đặc trưng từ P3.
  - Head: Là phần cuối cùng của mô hình, chịu trách nhiệm dự đoán bounding boxes và xác suất cho các lớp đối tượng.
  - Cải tiến của head:
    - Decoupled Head: YOLOv8 sử dụng decoupled head, tách biệt các nhánh dự đoán cho localization và classification. Điều này giúp cải thiện hiệu xuất và độ chính xác. Nó cũng sẽ giảm đi sự xung đột giữa các nhiệm vụ dự đoán và tăng cường khả năng tối ưu hóa mô hình.
    - Decoupled Head: YOLOv8 sử dụng decoupled head, tách biệt các nhánh dự đoán cho localization và classification. Điều này giúp cải thiện hiệu xuất và độ chính xác. Nó cũng sẽ giảm đi sự xung đột giữa các nhiệm vụ dự đoán và tăng cường khả năng tối ưu hóa mô hình.
  - Cấu trúc của head:
    - Detect Layers:

- Detect (P3): Dự đoán bounding boxes và lớp đối tượng tại P3.
- Detect (P4): Dự đoán bounding boxes và lớp đối tượng tại P4.
- Detect (P5): Dự đoán bounding boxes và lớp đối tượng tại P5.



Hình 28. Kiến trúc YOLOv8

- Các điểm mới trong YOLOv8:
  - Anchor-free Design: Khác với các phiên bản trước YOLOv8 sử dụng mô hình không cần anchor (anchor-free), giúp đơn giản hóa quá trình huấn luyện và cải thiện độ chính xác. Anchor-free giúp mô hình dự đoán trực tiếp vị trí của các bounding boxes mà không cần phải chọn trước các anchor boxes [11].
  - Decoupled Head: Decoupled head tách biệt dự đoán localization và classification thành hai nhánh riêng biệt, giúp cải thiện khả năng tối ưu hóa và độ chính xác của mô hình. Localization head dự đoán tọa độ bounding box, trong khi classification head dự đoán lớp của đối tượng.
  - Advanced Data Augmentation: YOLOv8 sử dụng các kỹ thuật tăng cường dữ liệu tiên tiến như Mosaic, MixUp, và Copy-Paste để cải thiện khả năng tổng quát hóa của mô hình. Các kỹ thuật này tạo ra các biến thể khác nhau của ảnh huấn luyện, giúp mô hình học được nhiều đặc trưng hơn từ dữ liệu.
  - Loss Functions: YOLOv8 sử dụng các hàm mất mát tiên tiến như CIoU (Complete IoU), DIoU (Distance IoU), và SIoU (Scale IoU) để cải thiện độ chính xác trong việc dự đoán bounding boxes. Các hàm mất mát này không chỉ xem xét diện tích giao nhau giữa các bounding boxes mà còn tính đến khoảng cách giữa các tâm và tỷ lệ kích thước.
  - Model Scaling: YOLOv8 cung cấp nhiều phiên bản mô hình với kích thước khác nhau (nano, small, medium, large, extra-large) để phù hợp với các yêu cầu tài nguyên khác nhau. Điều này giúp người dùng lựa chọn phiên bản phù hợp với nhu cầu về hiệu suất và tài nguyên tính toán.

- Task Alignment Learning: Sử dụng các kỹ thuật học tập điều chỉnh nhiệm vụ để giảm sự sai lệch giữa phân loại và định vị. Điều này giúp cải thiện hiệu suất của mô hình trong các tình huống phức tạp.
  - YOLOv8 cũng cung cấp một mô hình phân đoạn ngữ nghĩa gọi là YOLOv8-Seg, với cấu trúc backbone là CSPDarknet53 và các module C2f. Mô hình này đạt được kết quả tốt nhất trên các bộ dữ liệu phân đoạn ngữ nghĩa và phát hiện đối tượng, duy trì tốc độ và hiệu suất cao.
- Hiệu suất của YOLOv8
    - Độ chính xác: YOLOv8 cải thiện độ chính xác so với các phiên bản trước, với AP (Average Precision) cao hơn trên các bộ dữ liệu phổ biến như COCO. Điều này giúp YOLOv8 trở thành một lựa chọn tốt cho các bài toán yêu cầu độ chính xác cao.
    - Tốc độ: YOLOv8 được thiết kế để duy trì tốc độ cao, cho phép xử lý theo thời gian thực trên các thiết bị phần cứng hiện đại. Điều này rất quan trọng cho các ứng dụng yêu cầu phản hồi nhanh như giám sát an ninh, theo dõi sản phẩm ở băng chuyền, lái xe tự động và robot.

## 3.2 Object Tracking (Truy vết đối tượng)

### 3.2.1 Kalman Filter

Kalman Filter [24, 25] là một thuật toán mạnh mẽ được sử dụng để ước lượng trạng thái của một hệ thống động lực từ các phép đo có nhiễu. Thuật toán này được đặt theo tên của Rudolf E. Kalman, người đã phát triển nó vào năm 1960. Bộ lọc Kalman có ứng dụng rộng rãi trong nhiều lĩnh vực như điều khiển và dẫn đường, kỹ thuật y sinh và đặc biệt là theo dõi đối tượng.

Trong việc theo dõi đối tượng Kalman Filter được biết đến nhiều nhất với vai trò dự đoán các trạng thái của đối tượng hiện tại dựa vào các track trong quá khứ và cập nhật lại các detection sau khi đã được liên kết với các track trước đó.

Nguyên lý cơ bản của Kalman Filter bao gồm việc dự đoán và cập nhật:

- Dự đoán: Sử dụng mô hình toán học để dự đoán trạng thái tương lai của hệ thống.
- Cập nhật: Sử dụng các phép đo mới để điều chỉnh dự đoán, tối ưu hóa ước lượng trạng thái.

Dưới đây sẽ là các bước để thực hiện Kalman Filter [24, 25, 26].

Bước 1: Xây dựng mô hình đây là bước quan trọng nhất và chúng ta phải chắc chắn rằng các điều kiện của Kalman Filter phù hợp với vấn đề của chúng ta điều này bao gồm việc xác định các phương trình trạng thái và đo lường của hệ thống, cũng như các tham số liên quan như nhiễu quá trình và nhiễu đo lường và ta sẽ nói về loại Kalman Filter đơn giản nhất Linear Kalman Filter để dễ hiểu (ngoài ra còn có Extended Kalman filter, Unscented Kalman filter, ...). Dưới đây là hai phương trình của Kalman Filter.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (1) \quad [24]$$

Trong đó ở phương trình 1 là phương trình trạng thái:

- $x_k$ : Là trạng thái hệ thống đại diện cho các biến cần ước lượng tại thời điểm  $k$ .
- $A$ : Là ma trận chuyển đổi trạng thái.
- $B$ : Là ma trận chuyển đổi điều khiển (nếu có) mô tả ảnh hưởng của tín hiệu điều khiển  $u_k$  đến trạng thái.
- $u_k$ : Là tín hiệu điều khiển (nếu có).
- $w_k$ : Là nhiễu quá trình (Sự biến động ngẫu nhiên ảnh hưởng đến trạng thái của hệ thống. Giả định là phân phối Gaussian với trung bình bằng 0 và hiệp phương sai  $Q$  ).

$$z_k = Hx_k + v_k \quad (2) [24]$$

Trong đó ở phương trình 2 là phương trình đo lường:

- $z_k$ : Là giá trị đo lường từ hệ thống tại thời điểm  $k$ .
- $H$ : Là ma trận đo lường (Ma trận này mô tả cách trạng thái của hệ thống liên quan đến các giá trị đo lường).
- $v_k$ : Là nhiễu đo lường (Sự biến động ngẫu nhiên ảnh hưởng đến các phép đo của hệ thống. Giả định là phân phối Gaussian với trung bình bằng 0 và hiệp phương sai  $R$  ).

Điều này có nghĩa là  $x_k$  (giá trị tín hiệu) có thể được đánh giá bằng cách sử dụng phương trình ngẫu nhiên tuyến tính (phương trình thứ nhất). Bất kỳ  $x_k$  nào cũng là sự kết hợp tuyến tính của giá trị trước đó của nó cộng với tín hiệu điều khiển  $u_k$  và nhiễu quá trình. Hãy nhớ rằng hầu hết thời gian không có tín hiệu điều khiển  $u_k$ .

Phương trình thứ hai cho biết bất kỳ giá trị đo lường nào (mà chúng ta không chắc chắn về độ chính xác) là sự kết hợp tuyến tính giữa giá trị tín hiệu và nhiễu đo lường và cả hai đều được xem là Gaussian. Các thành phần **A**, **B** và **H** thường là ma trận nhưng trong hầu hết các vấn đề xử lý tín hiệu của chúng ta. Chúng ta sử dụng các mô hình sao cho các thành phần này chỉ là giá trị số. Ngoài ra để dễ dàng hơn mặc dù các giá trị này có thể thay đổi giữa các trạng thái nhưng trong hầu hết thời gian chúng ta có thể giả định rằng chúng không thay đổi.

Nếu chúng ta chắc được hệ thống chúng ta phù hợp thì điều duy nhất còn lại là ước tính giá trị trung bình và độ lệch chuẩn của các hàm nhiễu  $w_{k-1}$  và  $v_k$ . Chúng ta biết rằng ngoài đời thực không có tín hiệu nào là Gaussian nào là thuần túy nhưng chúng ta cũng có thể giả định nó với một số gần đúng.

Đây không phải là vấn đề lớn vì chúng ta sẽ thấy được rằng Kalman Filter cố gắng hội tụ thành các ước lượng chính xác ngay cả khi tham số nhiễu Gaussian được ước lượng kém.

---

Điều duy nhất ta cần ghi nhớ là “Ta ước tính các tham số nhiều càng chính xác thì ta càng nhận được ước tính chính xác hơn”

Bước 2: Bắt đầu quá trình sau khi xây dựng mô hình ở bước 1. Bước tiếp theo là khởi động quá trình sử dụng Kalman Filter. Điều này bao gồm việc xác định các tham số cần thiết và giá trị ban đầu, cũng như áp dụng các phương trình cập nhật thời gian và đo lường.

Chúng ta có 2 bộ phương trình riêng biệt: Phương trình cập nhật thời gian (dự đoán) và phương trình cập nhật đo lường (chỉnh sửa). Cả hai bộ phương trình đều được áp dụng ở mỗi trạng thái  $\text{thứ } k$ .

Phương trình cập nhật thời gian:

$$\begin{aligned}\hat{x}_k^- &= A\hat{x}_{k-1} + Bu_k \quad (1) \\ P_k^- &= AP_{k-1}A^T + Q \quad (2)\end{aligned}[24]$$

Trong đó

- Phương trình 1 dùng để dự đoán trạng thái ( $\hat{x}_k^-$ ):
  - $\hat{x}_k^-$ : Là ước lượng trạng thái tại thời điểm  $k$  trước khi cập nhật đo lường.
  - $A$ : Là ma trận chuyển đổi trạng thái
  - $\hat{x}_{k-1}$ : Là ước lượng trạng thái tại thời điểm  $k - 1$  sau khi cập nhật đo lường.
  - $B$ : Là ma trận điều khiển.
  - $u_k$ : Là tín hiệu tại thời điểm  $k$ .
- Phương trình 2 dùng để dự đoán phương sai của trạng thái ( $P_k^-$ ):
  - $P_k^-$ : Là hiệp phương sai của ước lượng trạng thái tại thời điểm  $k$  trước khi cập nhật đo lường.
  - $P_{k-1}$ : Là hiệp phương sai của ước lượng trạng thái tại thời điểm  $k - 1$  sau khi cập nhật đo lường.
  - $Q$ : Là hiệp phương sai của nhiều quá trình.

Phương trình cập nhật đo lường:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \quad (3)$$

$$\hat{x}_k = \hat{x}_k^- + K_k (z_k - H \hat{x}_k^-) \quad [24]$$

$$P_k = (I - K_k H) P_k^- \quad (5)$$

Trong đó

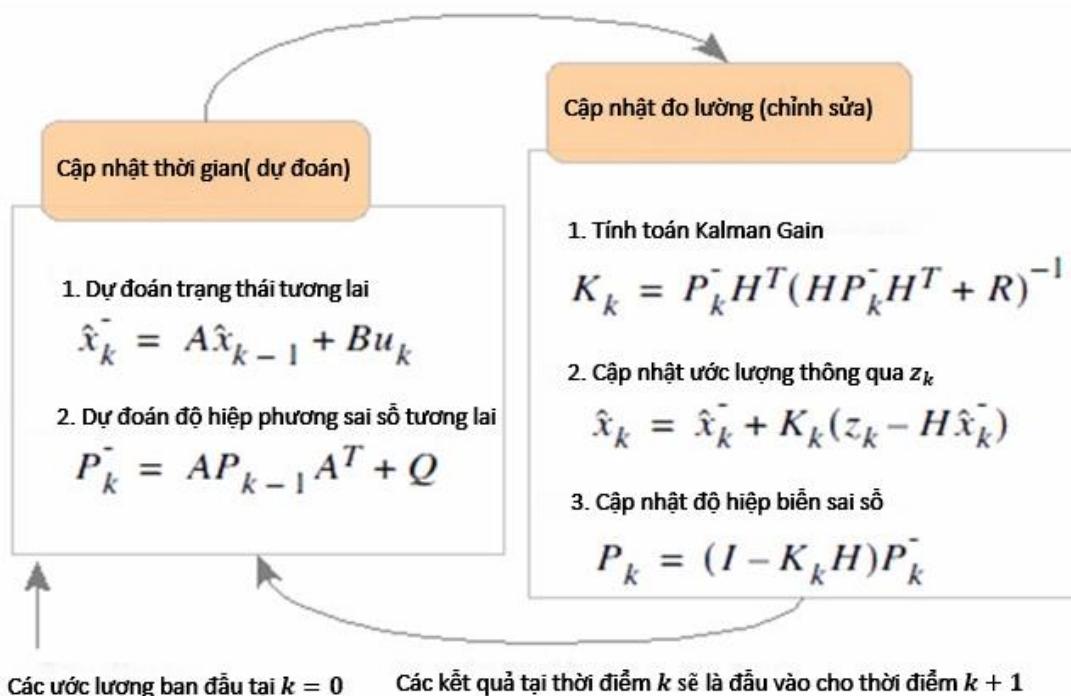
- Phương trình 3 dùng để tính toán hệ số Kalman ( $K_k$ ):
  - $K_k$ : Là hệ số Kalman tại thời điểm  $k$ .
  - $P_k^-$ : Là hiệp phương sai của ước lượng trạng thái trước khi cập nhật đo lường.
  - $H$ : Là ma trận đo lường.
  - $R$ : Hiệp phương sai của nhiễu đo lường.
- Phương trình 4 dùng để cập nhật trạng thái ( $\hat{x}_k$ ):
  - $\hat{x}_k$ : Là ước lượng trạng thái tại thời điểm  $k$  sau khi cập nhật đo lường.
  - $\hat{x}_k^-$ : Là ước lượng trạng thái trước khi cập nhật đo lường.
  - $z_k$ : Là giá trị đo lường tại thời điểm  $k$ .
  - $K_k$ : Là hệ số Kalman tại thời điểm  $k$ .
  - $H$ : Là ma trận đo lường.
- Phương trình 5 dùng để cập nhật hiệp phương sai của trạng thái ( $P_k$ ):
  - $P_k$ : Là hiệp phương sai của ước lượng trạng thái tại thời điểm  $k$  sau khi cập nhật đo lường.
  - $I$ : Là ma trận đơn vị.
  - $K_k$ : Là hệ số Kalman tại thời điểm  $k$
  - $H$ : Là ma trận đo lường.
  - $P_k^-$ : Là hiệp phương sai của ước lượng trạng thái trước khi cập nhật đo lường.

Chúng ta đã tạo mô hình ở bước 1 vì vậy chúng ta đã biết các ma trận **A**, **B** và **H** rất có thể chúng sẽ là hằng số và thậm chí chúng có thể bằng 1. Việc khó khăn nhất còn

---

lại là việc xác định  $\mathbf{R}$  và  $\mathbf{Q}$ . Với  $\mathbf{R}$  thì khá đơn giản để tìm ra bởi vì nói chung thì chúng ta sẽ khá chắc chắn được về nhiễu trong môi trường. Nhưng việc tìm ra  $\mathbf{Q}$  thì không dễ dàng và ở giải đoạn này thì chưa thể có một giải pháp cụ thể và để bắt đầu quá trình thì ta cần biết ước lượng của  $\mathbf{x}_0$  và  $\mathbf{P}_0$ .

Bước 3: Lặp lại quá trình sau khi đã xác định được các tham số cần thiết và khởi động quá trình bằng các phương trình cập nhật thời gian và đo lường, bước tiếp theo là lặp lại quá trình này qua từng bước thời gian. Đây là quá trình liên tục để cải thiện các ước lượng trạng thái của hệ thống theo thời gian.



Hình 29. Quá trình lặp lại của Kalman Filter

Ở đây  $\hat{\mathbf{x}}_k^-$  là ước lượng trước theo một cách nào đó, Có nghĩa là một ước lượng sơ bộ trước khi điều chỉnh và cập nhật đo lường.  $\mathbf{P}_k^-$  cũng như vậy nó hiệp phương sai của ước lượng trạng thái trước khi cập nhật đo lường. Chúng ta sử dụng các giá trị trước này trong các phương trình cập nhật đo lường của mình. Trong các phương trình cập nhật đo lường ta sẽ tìm thấy  $\hat{\mathbf{x}}_k$  là ước lượng trạng thái tại thời điểm  $k$  sau khi cập nhật đo lường đây chính là điều chúng ta muốn tìm nó sẽ là ước lượng tốt nhất dựa trên các thông tin đo lường và hệ thống. Quá trình lặp lại dự đoán và cập nhật qua

từng bước thời gian để liên tục cải thiện ước lượng trạng thái của hệ thống. Kalman Filter sử dụng dữ liệu đo lường mới nhất và mô hình hệ thống để tối ưu hóa ước lượng, giúp đạt được độ chính xác cao trong việc ước lượng trạng thái của hệ thống. Kết luận : Mục tiêu chính của việc sử dụng Kalman Filter là để ước lượng trạng thái thực của hệ thống từ các phép đo có nhiễu và giảm thiểu sai số ước lượng. Quá trình lặp lại của Kalman Filter sẽ tiếp tục cho đến khi đạt đến một trong các điều kiện dừng như số bước lặp cố định, sai số ước lượng đạt ngưỡng, không còn dữ liệu mới hoặc ước lượng trạng thái hội tụ.

### 3.2.2 Thuật toán Hungarian

Thuật toán Hungarian [27] còn được biết đến với tên gọi thuật toán Kuhn-Munkres, là một phương pháp được sử dụng để giải quyết bài toán phân công (Assignment Problem) trong đồ thị hai phía. Bài toán phân công yêu cầu tìm cách gán các đối tượng từ một tập hợp này sang một tập hợp khác sao cho tổng chi phí là nhỏ nhất hoặc tổng trọng số là lớn nhất. Thuật toán này được áp dụng rộng rãi trong nhiều lĩnh vực bao gồm tối ưu hóa, vận tải và theo dõi đối tượng trong thời gian thực.

Lịch sử phát triển:

- Harold Kuhn phát triển và công bố thuật toán này vào năm 1955, dựa trên công trình của các nhà toán học Hungary là Dénes König và Jenő Egerváry.
- James Munkres sau đó chứng minh rằng thuật toán này có thể chạy trong thời gian đa thức vào năm 1957, do đó thuật toán cũng được gọi là thuật toán Kuhn-Munkres.
- Một khám phá sau này cho thấy Carl Gustav Jacobi đã phát minh ra một thuật toán tương tự từ một thế kỷ trước, nhưng công trình này đã bị lãng quên.

Bài toán phân công có thể được mô tả dưới nhiều dạng khác nhau nhưng bản chất vẫn giống nhau [28]:

- Có  $n$  công việc và  $n$  người lao động. Mỗi người lao động yêu cầu một khoản tiền nhất định để thực hiện một công việc cụ thể. Mục tiêu là gán các công việc cho người lao động sao cho tổng chi phí là nhỏ nhất.
- Cho một ma trận chi phí  $n \times n$ , chọn một số từ mỗi hàng sao cho mỗi số được chọn từ một cột khác nhau, và tổng các số được chọn là nhỏ nhất.
- Cho một đồ thị hai phía hoàn chỉnh với  $n$  đỉnh mỗi phía, mỗi cạnh có một trọng số. Mục tiêu là tìm một bộ ghép hoàn hảo với tổng trọng số là nhỏ nhất.

	W <sub>1</sub>	W <sub>2</sub>	W <sub>3</sub>	W <sub>4</sub>	W <sub>5</sub>	W <sub>6</sub>
V <sub>1</sub>	1	1	0	0	0	0
V <sub>2</sub>	1	1	1	1	0	0
V <sub>3</sub>	0	1	1	0	0	0
V <sub>4</sub>	0	0	1	1	1	0
V <sub>5</sub>	0	0	0	1	1	1
V <sub>6</sub>	0	0	0	1	1	0

Hình 30. Minh họa về thuật toán Hungarian

Đầu tiên ta sẽ bắt đầu với việc khởi tạo ma trận chi phí ban đầu: Cho một ma trận chi phí  $C$  có kích thước  $n \times n$  trong đó  $C[i][j]$  là chi phí gán công việc  $i$  cho người lao động  $j$  hoặc có thể là chi phí cho một việc gì đó mà ta cần phân công tối ưu.

Các bước thực hiện thuật toán Hungarian:

- Subtract Row Minimums (Trừ giá trị nhỏ nhất của hàng): Trừ giá trị nhỏ nhất trong mỗi hàng từ tất cả các phần tử trong hàng đó.
- Subtract Column Minimums (Trừ giá trị nhỏ nhất của cột): Trừ giá trị nhỏ nhất trong mỗi cột từ tất cả các phần tử trong cột đó.
- Cover All Zeros (Che phủ tất cả số 0): Vẽ các đường ngang và dọc qua các số 0 sao cho số đường kẻ là ít nhất.
  - Nếu số đường kẻ ít hơn  $n$ , chuyển sang bước 4.
  - Nếu số đường kẻ bằng  $n$  gán các công việc theo các số 0 đã chọn và dừng lại.

- Adjust Matrix (Điều chỉnh ma trận):
  - Tìm giá trị nhỏ nhất không bị che phủ bởi bất kỳ đường nào.
  - Trừ giá trị này từ tất cả các phần tử không bị che phủ và cộng giá trị này vào tất cả các phần tử bị che phủ bởi hai đường.
- Lặp lại các bước trên cho đến khi có đủ số đường che phủ bằng số hàng (hoặc cột).
- Sử dụng các đường che phủ để xác định các gán ghép tối ưu giữa các đối tượng.

Ứng dụng trong theo dõi đối tượng (Object Tracking):

Trong việc theo dõi đối tượng, thuật toán Hungarian được sử dụng để gán các đối tượng phát hiện trong khung hình hiện tại với các đối tượng đã được theo dõi từ các khung hình trước đó. Điều này giúp xác định và theo dõi các đối tượng qua nhiều khung hình trong video và chúng ta sẽ đi rõ vào nó ở phần tiếp theo là thuật toán SORT (Simple Online and Realtime Tracking).

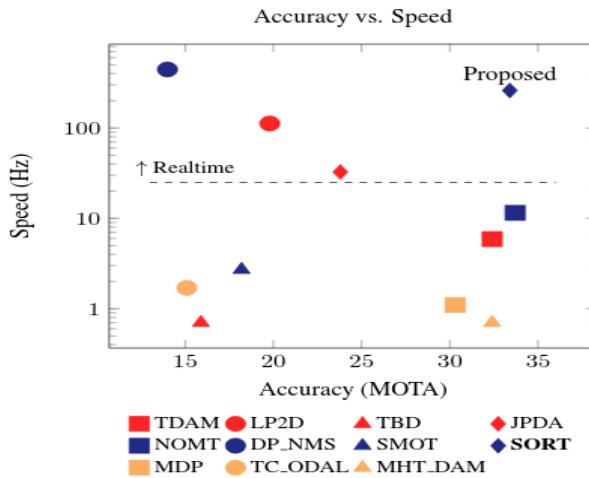
Kết luận: Thuật toán Hungarian là một công cụ mạnh mẽ để giải quyết các bài toán phân công tối ưu và có nhiều ứng dụng thực tế, từ quản lý công việc đến theo dõi đối tượng trong video.

### 3.2.3 Thuật toán SORT (Simple Online and Realtime Tracking):

Giới thiệu về SORT:

SORT (Simple Online and Realtime Tracking) [29] được giới thiệu lần đầu năm 2016, chính sửa bổ sung v2 vào năm 2017 do Alex Bewley và những tác giả khác phát triển. Là một thuật toán theo dõi đối tượng thời gian thực đơn giản nhưng hiệu quả. Được giới thiệu nhằm mục đích cung cấp một giải pháp theo dõi nhanh chóng và dễ triển khai cho các ứng dụng yêu cầu xử lý thời gian thực. Đề xuất giải pháp cho object tracking, đồng thời giải quyết cả 2 vấn đề multiple object tracking và realtime object tracking. SORT kết hợp các kỹ thuật đơn giản và hiệu quả như Kalman Filter và Hungarian algorithm để theo dõi đối tượng qua các khung hình video.

---



Hình 31. Biểu đồ độ chính xác và tốc độ giữa những phương pháp

Các phương pháp theo dõi đối tượng phổ biến trước SORT:

- Multiple Hypothesis Tracking (MHT): Tạo ra nhiều giả thuyết về quỹ đạo của đối tượng và chọn lựa giả thuyết tốt nhất dựa trên các quan sát.
- Joint Probabilistic Data Association (JPDA): Tính toán xác suất chung cho tất cả các kết hợp có thể có giữa phát hiện và đối tượng.
- Particle Filters: Sử dụng các hạt để biểu diễn phân phối xác suất của trạng thái đối tượng và cập nhật chúng dựa trên các quan sát mới.
- Tracklet-based Methods: Chia nhỏ các quỹ đạo dài thành các đoạn ngắn hơn (tracklets) và sau đó liên kết chúng lại với nhau.
- Global Optimization Methods: Tối ưu hóa toàn cục cho tất cả các khung hình cùng một lúc, chẳng hạn như sử dụng Linear Programming hoặc Network Flow.

Những phương pháp này thường phức tạp và yêu cầu tài nguyên tính toán lớn, do đó không phù hợp cho các ứng dụng thời gian thực. MHT và JPDA là hai phương pháp có mối liên hệ với SORT trong bối cảnh theo dõi nhiều đối tượng.

Tuy nhiên, điểm yếu của hai phương pháp này là độ phức tạp tính toán cao và trì hoãn việc đưa ra quyết định gán. Do đó, SORT đã ra đời và cải thiện được hầu hết các yếu điểm về tốc độ xử lý, độ phức tạp tính toán, và hiệu suất, tính chính xác so với các phương pháp cũ.

Phương pháp luận của SORT:

SORT bao gồm bốn thành phần chính: Phát hiện đối tượng, Kalman Filter (có 3 loại Linear – Extended – Unscented trong nội dung dưới đây sử dụng chính là Linear Kalman Filter), Hungarian Algorithm và quản lý danh tính đối tượng.

Đầu tiên ta sẽ nói về phát hiện đối tượng: Trong mỗi khung hình video, các đối tượng được phát hiện bằng các mô hình Deep Learning như YOLO hoặc Faster R-CNN và có thể là bất kỳ mô hình detector nào. Kết quả của bước này là một loạt các hộp giới hạn (bounding boxes) chứa các đối tượng được phát hiện [25].

Ví dụ như nếu có một khung hình video chứa những người đi bộ. YOLO sẽ phát hiện và vẽ các hộp giới hạn xung quanh từng người.



Hình 32. Ví dụ về phát hiện đối tượng

Sau đó sẽ nói đến với Kalman Filter thì công việc của Kalman Filter ở thuật SORT sẽ là một thuật toán sử dụng để dự đoán vị trí tương lai của các đối tượng. Mô hình này sử dụng trạng thái hiện tại của đối tượng (bao gồm vị trí và vận tốc) để dự đoán trạng thái tiếp theo. Khi có thông tin mới từ các phát hiện đối tượng, Kalman Filter sẽ cập nhật dự đoán của mình để cải thiện độ chính xác [25].

---

Công thức của Kalman Filter (Linear Kalman Filter):

$$x = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}]^T [25]$$

Với

- $x$  : Có ma trận hiệp phương sai ban đầu được khởi tạo với giá trị lớn để thể hiện sự không chắc chắn của trạng thái.
- $u, v$  : Lần lượt là tọa độ của tâm đối tượng (ở đây là tâm bounding box).
- $s$  : Là diện tích của bounding box.
- $r$  : Là tỉ lệ aspect ratio của bounding box.
- $\dot{x}, \dot{y}, \dot{s}$  : Lần lượt là giá trị vận tốc tương ứng của  $x, y, s$ .

Ví dụ như một người đi bộ được phát hiện tại tọa độ (100, 200) với vận tốc (5, 0). Kalman Filter sẽ dự đoán vị trí của người đi bộ trong khung hình tiếp theo là (105, 200) [25].

Tiếp đến là Hungarian Algorithm thuật toán này sẽ giải quyết bài toán liên kết các phát hiện mới với các đối tượng đã theo dõi, dựa trên chỉ số IoU (Intersection-over-Union). Quy trình này bao gồm các bước sau:

- Tính toán IoU: Đo lường mức độ trùng lặp giữa các hộp giới hạn dự đoán và hộp giới hạn mới.
- Tạo ma trận chi phí: Sử dụng IoU để tạo ra ma trận chi phí, trong đó giá trị của mỗi ô là  $1 - \text{IoU}$

Gán tối ưu: Hungarian Algorithm tìm cách gán tối ưu giữa các phát hiện và các đối tượng để tổng chi phí là nhỏ nhất. Ví dụ chúng ta có 3 đối tượng được theo dõi từ khung hình trước (A, B, C) và 3 phát hiện mới (1, 2, 3). Ma trận chi phí có thể như sau:

Đối tượng	1	2	3
A	0.75	0.50	0.90
B	0.60	0.30	0.80
C	0.85	0.70	0.40

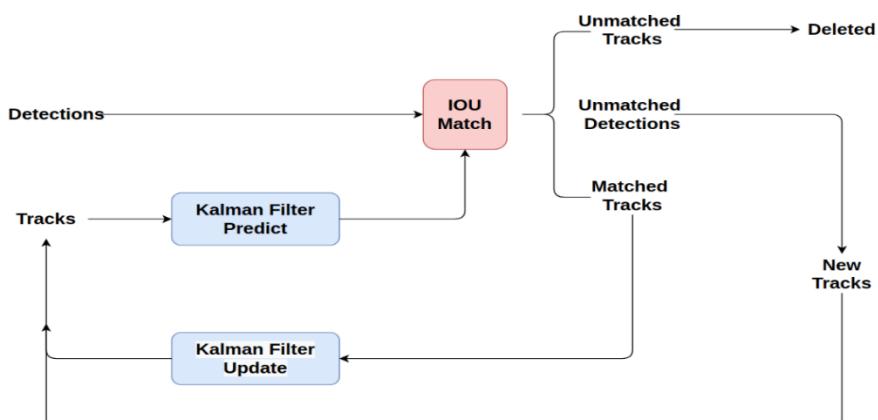
Bảng 2. Ví dụ ma trận chi phí

Hungarian Algorithm sẽ gán: A-2, B-1, C-3 với tổng chi phí nhỏ nhất.

Sau đó sẽ đến với quản lý danh tính và đối tượng. SORT quản lý danh tính đối tượng bằng cách tạo danh tính mới khi phát hiện đối tượng mới và xóa danh tính khi đối tượng không được phát hiện trong một số khung hình liên tiếp. Các bước cụ thể bao gồm:

- Tạo danh tính mới: Khi đối tượng mới xuất hiện và không trùng với bất kỳ đối tượng nào đã theo dõi, hệ thống tạo một danh tính mới.
- Xóa danh tính: Nếu đối tượng không được phát hiện trong hai khung hình liên tiếp, danh tính của nó sẽ bị xóa để tránh theo dõi sai.

Ví dụ: Nếu một người đi bộ mới xuất hiện, hệ thống sẽ tạo danh tính mới cho người đó. Nếu người đi bộ không được phát hiện trong vài khung hình tiếp theo, danh tính sẽ bị xóa.



Hình 33. Quy trình hoạt động của thuật toán SORT

---

SORT sẽ hoạt động theo các bước chính sau:

- Detections (Phát hiện): Các đối tượng được phát hiện trong mỗi khung hình.
- Kalman Filter Predict (Dự đoán bằng Kalman Filter): Kalman Filter sử dụng trạng thái hiện tại của các đối tượng để dự đoán vị trí tương lai của chúng.
- IOU Match (Khớp theo IOU): Tính toán IoU giữa các hộp giới hạn dự đoán và hộp giới hạn mới, sau đó sử dụng Hungarian Algorithm để tìm cách gán tối ưu.
- Matched Tracks (Các đối tượng khớp): Các đối tượng được khớp sẽ tiếp tục được theo dõi và cập nhật.
- Unmatched Tracks (Các đối tượng không khớp): Các đối tượng không khớp với bất kỳ phát hiện mới nào sẽ bị xóa sau một số khung hình nhất định.
- Unmatched Detections (Các phát hiện không khớp): Các phát hiện mới không khớp với bất kỳ đối tượng dự đoán nào sẽ được tạo thành các track mới.
- New Tracks (Các đối tượng mới): Các đối tượng mới sẽ được theo dõi như là các track mới.
- Kalman Filter Update (Cập nhật Kalman Filter): Kalman Filter cập nhật trạng thái dự đoán của mình dựa trên các phát hiện mới đã gán để cải thiện độ chính xác.
- Tracks (Các đối tượng): Quay lại vòng lặp dự đoán và cập nhật với các track hiện có.

Những quy trình này tiếp tục lặp lại cho mỗi khung hình trong video, giúp theo dõi các đối tượng một cách hiệu quả và chính xác trong thời gian thực [25].

Kết quả và hiệu suất của SORT:

SORT đã được chứng minh là một giải pháp theo dõi đối tượng hiệu quả với tốc độ cao, đặc biệt phù hợp cho các ứng dụng thời gian thực như giám sát an ninh và hệ thống xe tự hành. SORT không chỉ dễ triển khai mà còn đạt được độ chính xác cao nhờ sự kết hợp của các thuật toán Kalman Filter và Hungarian.

---

Tuy nhiên nhược điểm của SORT là:

- Không xử lý tốt khi đối tượng bị che khuất lâu: SORT không hiệu quả khi đối tượng bị che khuất trong một thời gian dài vì nó không có cơ chế nhận dạng lại đối tượng. Điều này là do ID Switches đây là vấn đề lớn nhất của SORT hiện tại. Do việc liên kết giữa detection và track trong SORT chỉ đơn giản dựa trên độ đo IOU (tức SORT chỉ quan tâm đến hình dạng của đối tượng) điều này gây ra hiện tượng số lượng ID Switches của 1 đối tượng là vô cùng lớn khi đối tượng bị che khuất, khi quay đao trùng lặp.
- Độ chính xác phụ thuộc vào chất lượng phát hiện: Nếu chất lượng phát hiện đối tượng không cao, hiệu suất của SORT cũng bị ảnh hưởng.
- Không phù hợp cho các môi trường phức tạp: Trong các môi trường phức tạp với nhiều đối tượng tương tự, SORT có thể gặp khó khăn trong việc gán đúng các đối tượng.

Thực nghiệm:

SORT được đánh giá trên bộ dữ liệu MOT (Multiple Object Tracking) với các chuỗi video chứa cả camera động và tĩnh. Hiệu suất của SORT được đo bằng các chỉ số như MOTA và MOTP [29].

Method	Type	MOTA↑	MOTP↑	FAF↓	MT↑	ML↓	FP↓	FN↓	ID sw↓	Frag↓
TBD [20]	Batch	15.9	70.9	2.6%	6.4%	47.9%	14943	34777	1939	1963
ALEXTRAC [5]	Batch	17.0	71.2	1.6%	3.9%	52.4%	9233	39933	1859	1872
DP_NMS [23]	Batch	14.5	70.8	2.3%	6.0%	40.8%	13171	34814	4537	3090
SMOT [1]	Batch	18.2	71.2	1.5%	2.8%	54.8%	8780	40310	1148	2132
NOMT [11]	Batch	<b>33.7</b>	71.9	<b>1.3%</b>	12.2%	44.0%	7762	32547	<b>442</b>	<b>823</b>
RMOT [4]	Online	18.6	69.6	2.2%	5.3%	53.3%	12473	36835	684	1282
TC_ODAL [17]	Online	15.1	70.5	2.2%	3.2%	55.8%	12970	38538	637	1716
TDAM [18]	Online	33.0	<b>72.8</b>	1.7%	<b>13.3%</b>	39.1%	10064	<b>30617</b>	464	1506
MDP [12]	Online	30.3	71.3	1.7%	13.0%	38.4%	9717	32422	680	1500
SORT (Proposed)	Online	<b>33.4</b>	72.1	<b>1.3%</b>	11.7%	<b>30.9%</b>	<b>7318</b>	32615	1001	1764

Hình 34. Bảng thông số đánh giá các mô hình tracking

Trong đó:

- MOTA (↑): Độ chính xác theo dõi đối tượng.

- MOTP ( $\uparrow$ ): Độ chính xác định vị đối tượng.
- FAF ( $\downarrow$ ): Số lần báo động sai mỗi khung hình.
- MT ( $\uparrow$ ): Số lượng đối tượng được theo dõi chủ yếu.
- ML ( $\downarrow$ ): Số lượng đối tượng bị mất theo dõi.
- FP ( $\downarrow$ ): Số lần phát hiện sai.
- FN ( $\downarrow$ ): Số lần bỏ sót đối tượng.
- ID sw ( $\downarrow$ ): Số lần thay đổi nhận diện.
- Frag ( $\downarrow$ ): Số lần phân đoạn.

Kết luận:

SORT (Simple Online and Realtime Tracking) là một giải pháp đơn giản nhưng hiệu quả cho bài toán theo dõi nhiều đối tượng thời gian thực. Bằng cách kết hợp Kalman Filter và Hungarian Algorithm, SORT đạt được độ chính xác cao và dễ triển khai. Mặc dù có nhược điểm như không xử lý tốt các trường hợp che khuất lâu dài và phụ thuộc vào chất lượng phát hiện, SORT vẫn là lựa chọn tuyệt vời cho các ứng dụng như giám sát an ninh và xe tự hành nhờ tính đơn giản và hiệu suất cao.

### 3.3 Tạo ROI (Region Of Interest)

- ROI là hình chữ nhật
  - Cách tạo ROI: ROI hình chữ nhật [30] được xác định bằng cách chọn hai điểm: góc trên bên trái và góc dưới bên phải. Tọa độ của ROI hình chữ nhật có thể được biểu diễn bằng bốn giá trị ( $x_1, y_1, x_2, y_2$ ) trong đó ( $x_1, y_1$ ): tọa độ của góc trên bên trái và ( $x_2, y_2$ ): tọa độ của góc dưới bên phải.
- Thuật toán xác định đối tượng trong ROI là hình chữ nhật
  - Kiểm tra xem toàn bộ bounding box (BB) có nằm hoàn toàn trong ROI hay không bằng cách so sánh các tọa độ.

- Ưu và Khuyết Điểm của ROI là hình chữ nhật:
  - Ưu Điểm: Dễ dàng thiết lập và tính toán, thực hiện kiểm tra nhanh chóng, hiệu quả.
  - Khuyết điểm: Chỉ phù hợp cho các ứng dụng với hình dạng ROI đơn giản. Không thể mô tả chính xác các vùng ROI phức tạp hoặc không phải hình chữ nhật.
- ROI là hình đa giác
  - Cách tạo ROI: ROI đa giác [30] được xác định bởi một danh sách các điểm (tọa độ x, y) tạo thành các đỉnh của đa giác.
    - Xác định các đỉnh của đa giác: Mỗi đỉnh được biểu diễn bằng một cặp tọa độ (x, y).
    - Tạo danh sách các đỉnh: Danh sách chứa các cặp tọa độ theo thứ tự liên tiếp tạo thành các cạnh của đa giác.
- Thuật Toán xác định đối tượng trong ROI là hình đa giác.
  - Để xác định một điểm có nằm trong đa giác lồi hay không, chúng ta có thể sử dụng hàm cv2.pointPolygonTest trong OpenCV.
  - Xác Định Trọng Tâm Bounding Box Nằm Trong ROI Đa Giác: Sử dụng trọng tâm của BB và hàm cv2.pointPolygonTest để kiểm tra xem điểm đó có nằm trong ROI đa giác hay không.
- Ưu và Khuyết Điểm:
  - Ưu điểm: Mô tả chính xác các vùng ROI phức tạp. Linh hoạt hơn trong việc xác định các khu vực không phải hình chữ nhật.
  - Khuyết điểm: Tính toán phức tạp hơn và có thể chậm hơn so với ROI hình chữ nhật. Cần thuật toán phức tạp hơn để kiểm tra.

### 3.4 Tính kích thước đối tượng

Việc tính kích thước đối tượng [31] để phục vụ cho ứng dụng của bài toán sẽ được thực hiện theo các bước dưới đây:

- Bước 1: Tính tỷ lệ pixel trên đơn vị đo lường (pixels per metric)
  - Chọn đối tượng tham chiếu: Lựa chọn một đối tượng có kích thước đã biết (metric) đơn vị cm hoặc mm.
  - Đặt đối tượng lên băng chuyền: Đặt đối tượng tham chiếu lên băng chuyền và chụp ảnh với camera.
  - Xác định bbox: Sử dụng công cụ để xác định bounding box của đối tượng tham chiếu trong ảnh ở đây nhóm sử dụng là YOLOv8.
  - Tính kích thước bbox: Đo độ rộng của bbox (pixels\_size).
  - Tính tỷ lệ pixels per metric

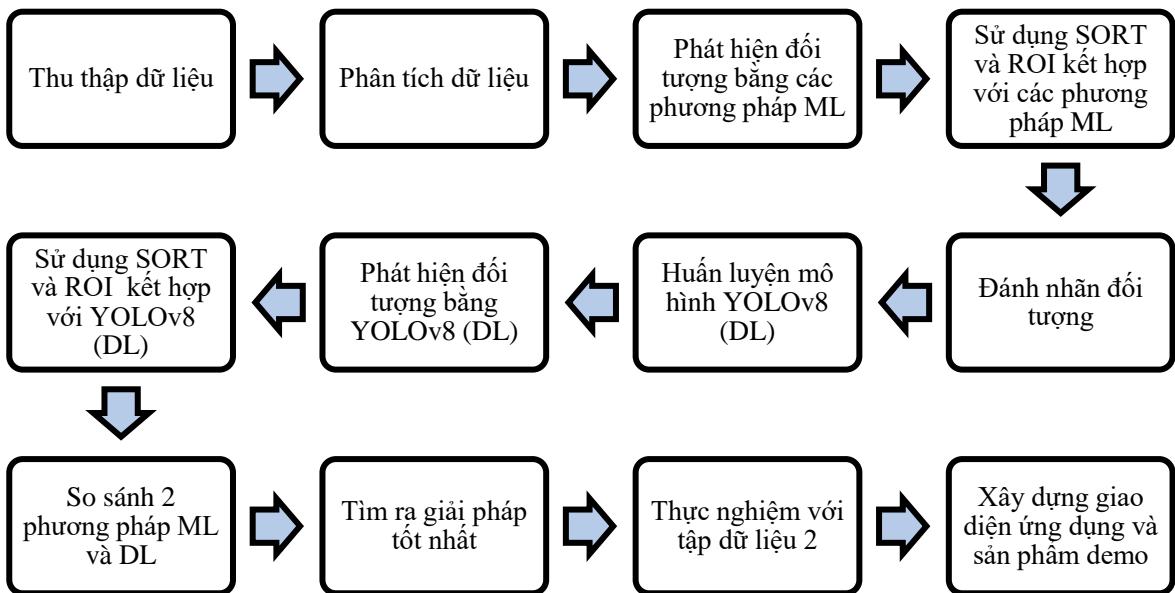
$$\text{Pixels per metric} = \frac{\text{pixels\_size}}{\text{metric}} \quad [31]$$

- Bước 2: Tính thước các đối tượng khác
  - Xác định bounding box: Sử dụng công cụ để xác định bounding box của các đối tượng cần đo kích thước ở đây nhóm sử dụng là YOLOv8.
  - Tính độ rộng, cao của bounding box: Đo các giá trị d1, d2 tương ứng với chiều rộng và chiều cao của bounding box.
  - Tính kích thước thực tế của đối tượng:

$$d1size = \frac{d1}{\text{pixels per metric}} \quad [31]$$

$$d2size = \frac{d2}{\text{pixels per metric}} \quad [31]$$

## Chương 4. Giải pháp đề xuất



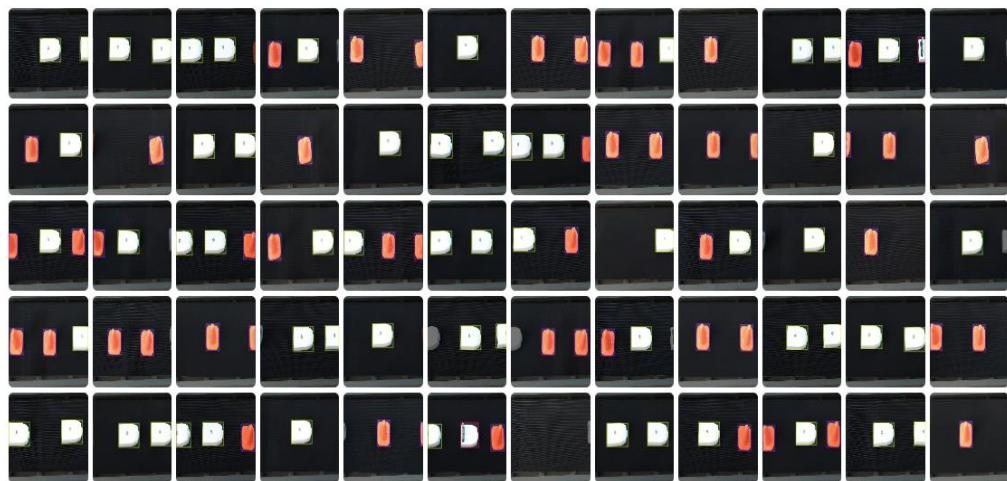
Mô tả: Giải pháp đề xuất chi tiết của bài toán bao gồm 12 bước như mô tả của biểu đồ 4 trên.

### 4.1 Phân tích dữ liệu

#### 4.1.1 Tập dữ liệu 1

- Số lượng mẫu và kích thước của tập dữ liệu 1:
  1. Tổng số lượng ảnh: 593 ảnh và sau tăng cường là 1008 ảnh
  2. Nguồn dữ liệu: Giảng viên viên hướng dẫn cung cấp
  3. Số lượng ảnh trong tập huấn luyện (train): 840 ảnh.
  4. Số lượng ảnh trong tập kiểm tra (validation): 119 ảnh.
  5. Số lượng ảnh trong tập xác nhận (test): 59 ảnh.
  6. Kích thước dữ liệu: 88,4MB.
  7. Số lượng đối tượng: 3 đối tượng.

Hình ảnh của tập dữ liệu 1 (tập dữ liệu có sẵn) được chia theo tỷ lệ 70% cho train và 20% cho valid, 10% cho test. Tập dữ liệu được hiển thị ở hình 35 phía dưới.



Hình 35. Hình ảnh bộ dữ liệu 1

- Mô tả các đối tượng ở tập dữ liệu 1

1. Object 1: Là một chiếc thùng màu trắng có một dấu mộc màu xám xanh ở giữa hoặc gần giữa thùng và được hiển thị ở hình 36 phía dưới.
  - Xuất hiện nhiều nhất trong tập dữ liệu.
  - Đặc điểm: Khi che 1 nửa, object 1 và object 3 giống nhau.



Hình 36. Hình ảnh Object 1

2. Object 2: Là một chiếc cốc tiêu màu cam và được hiển thị ở hình 37 phía dưới.
  - Đặc điểm riêng biệt không giống với các đối tượng khác khi bị che khuất.



Hình 37. Hình ảnh Object 2

3. Object 3: Là một chiếc thùng màu trắng khá giống đối tượng thứ nhất nhưng ở cuối thùng có ba chiếc cột tròn màu xám ánh kim loại dưới ba chiếc cột tròn đó có một nền bao quanh có màu sắc là màu xanh lá đậm và ở đáy thùng có một nền có màu tương tự là màu xanh lá đậm và được hiển thị ở hình 38 phía dưới.

- Chỉ xuất hiện 1 lần trong toàn bộ tập dữ liệu.
- Đặc điểm: Khi che 1 nửa object 3 và object 1 giống nhau.



Hình 38. Hình ảnh Object 3

- Nhận xét về tập dữ liệu 1

1. Mục đích sử dụng của tập dữ liệu 1:

- Sử dụng để tìm hiểu các phương pháp phát hiện đối tượng kết hợp với thuật toán truy vết đối tượng nhằm tìm ra phương án tốt nhất để thực hiện trên tập dữ liệu thực tế là tập dữ liệu 2.

2. Sự phân bố của tập dữ liệu 1:

- Tập dữ liệu có sự chênh lệch về số lượng xuất hiện của các đối tượng, đặc biệt là object 3 xuất hiện rất ít (chỉ 1 lần), điều này có thể gây ra khó khăn trong việc huấn luyện mô hình nhận diện object 3 với độ chính xác cao.
- Object 1 và object 3 có đặc điểm giống nhau khi bị che khuất một phần, điều này có thể dẫn đến việc nhầm lẫn giữa hai đối tượng trong quá trình nhận diện.

3. Kích thước của tập dữ liệu 1:

- Kích thước 88,4MB cho 1008 ảnh là một kích thước không quá lớn nhưng đảm bảo đủ dữ liệu để huấn luyện và kiểm tra mô

hình mà không quá lớn để gây khó khăn cho quá trình xử lý và lưu trữ.

#### 4. Chất lượng của tập dữ liệu 1:

- Ở mức ổn do cắt mỗi 5 frame 1 hình trên video nên dữ liệu thiếu đa dạng và video quay có những lúc mất nét và nhiễu nên bộ dữ liệu có chất liệu ổn và có thể train và nhận diện được vật thể

##### 4.1.2 Tập dữ liệu 2

- Số lượng mẫu và kích thước của tập dữ liệu 2:

1. Tổng số lượng ảnh: 3234 ảnh và sau tăng cường là 12290 ảnh.
2. Nguồn gốc: Tập dữ liệu tự thu thập
3. Số lượng ảnh trong tập huấn luyện (train): 11320 ảnh.
4. Số lượng ảnh trong tập kiểm tra (test): 647 ảnh.
5. Số lượng ảnh trong tập xác nhận (validation): 323 ảnh.
6. Kích thước dữ liệu: 742MB.
7. Số lượng đối tượng: 4 đối tượng.

Hình ảnh của tập dữ liệu 2 (tự thu thập) được chia theo tỷ lệ 70% cho train, 20% cho valid và 10% cho test. Tập dữ liệu được hiển thị ở hình 39 phía dưới.



Hình 39. Hình ảnh bộ dữ liệu 2

- Mô tả các đối tượng ở tập dữ liệu 1

1. HopKeo1: Là một chiếc hộp kẹo màu hồng có hình chữ nhật hơi bầu và được hiển thị ở hình 40 phía dưới.

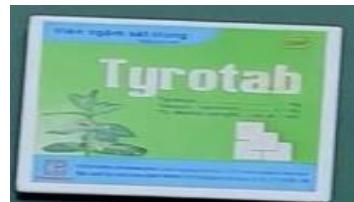
- Xuất hiện nhiều nhất trong tập dữ liệu.
- Kích thước: Chiều cao 4.9cm và chiều rộng 6cm.
- Đặc điểm: Có màu khá giống phần đầu của MayNgheNhac.



Hình 40. Hình ảnh HopKeo1

2. HopKeo2: Là một chiếc hộp kẹo màu xanh trắng có màu xanh lá có hình chữ nhật và được hiển thị ở hình 41 phía dưới.

- Xuất hiện nhiều thứ ba trong tập dữ liệu.
- Kích thước: Chiều cao 4.8cm và chiều rộng 5.6cm.
- Đặc điểm: Hình dạng giống HopKeo1 nhưng về màu sắc thì khá đặc biệt do không giống các đối tượng khác.



Hình 41. Hình ảnh HopKeo2

3. MayNgheNhac: Là một chiếc máy nghe nhạc màu hồng và đen có kích thước dài và thon và được hiển thị ở hình 42 phía dưới.

- Xuất hiện nhiều thứ hai trong tập dữ liệu.
- Kích thước: Chiều cao 9cm và chiều rộng 2.35cm.

- Đặc điểm: Khác biệt với các đối tượng khác nhưng có phần đầu có màu khá giống HopKeo 1.



Hình 42. Hình ảnh MayNgheNhac

4. XitThomMieng: Là một cái Xịt thơm miệng màu hồng và trắng cũng có hình dạng dài và thon và được hiển thị ở hình 43 phía dưới.

- Xuất hiện ít hơn các đối tượng khác trong tập dữ liệu.
- Kích thước: Chiều cao 11.9cm và chiều rộng 4.35 cm.
- Đặc điểm: Khác biệt với các đối tượng khác và có kích thước lớn nhất.



Hình 43. Hình ảnh XitThomMieng

- Nhận xét về tập dữ liệu 2

1. Mục đích sử dụng của tập dữ liệu 2 : Dựa trên những tìm hiểu ở tập dữ liệu 1 tìm ra được phương pháp tối ưu từ đó áp dụng phương pháp ấy vào thực tế là tập dữ liệu 2.

2. Sự phân bố của tập dữ liệu 2:

- Tập dữ liệu có sự chênh lệch về số lượng xuất hiện của các đối tượng nhưng không đáng kể do vật thể ít xuất hiện nhất nhưng có những đặc trưng riêng biệt nên vẫn ổn khi chạy thử và sự

phân bố của nó cũng cho ra kết quả khá khả quan khi test thực tế.

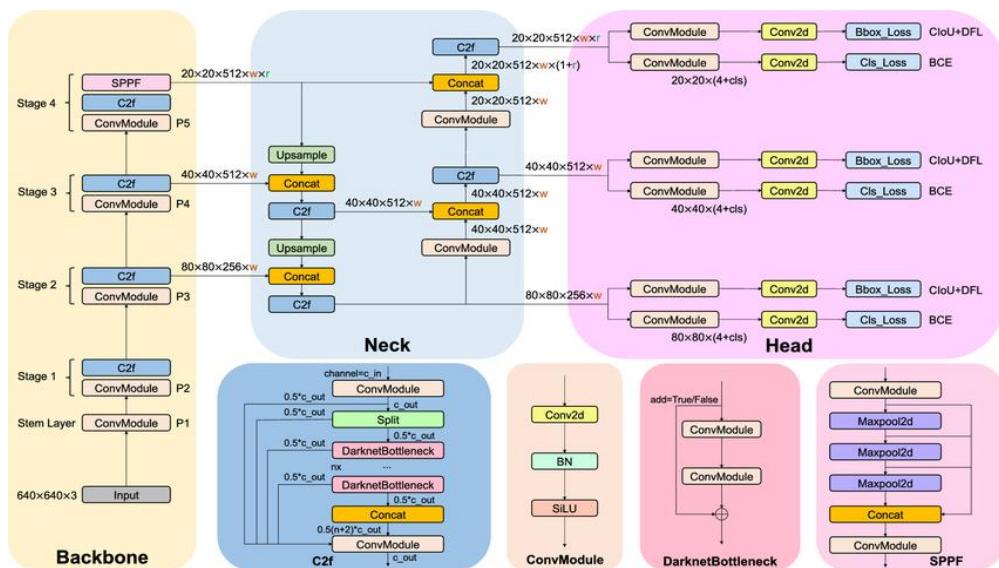
### 3. Kích thước của tập dữ liệu 2:

- Kích thước 762MB cho 12290 ảnh là một kích thước ổn và đảm bảo đủ dữ liệu để huấn luyện và kiểm tra mô hình mà không quá lớn để gây khó khăn cho quá trình xử lý và lưu trữ.

### 4. Chất lượng của tập dữ liệu 2:

- Ở mức tốt và được lấy 5 frame 1 hình trên video và tự quay nên có nhiều gốc và đã được tăng cường dữ liệu khá nhiều ở tập train nhằm đảm bảo phát hiện đối tượng tốt nhất.

## 4.2 Mô hình



Hình 44. Kiến trúc YOLOv8

Cấu trúc bao gồm:

Backbone:

- Input: Đầu vào là hình ảnh với kích thước  $640 \times 640 \times 3$ .
- Stem Layer: Gồm một khối ConvModule đầu tiên để giảm kích thước từ  $640 \times 640 \times 3$  xuống  $320 \times 320 \times 64$ .

- Stage 1 đến Stage 4:
  1. Stage 1: ConvModule và C2f giảm kích thước xuống 160x160x128.
  2. Stage 2: ConvModule và C2f giảm kích thước xuống 80x80x256.
  3. Stage 3: ConvModule và C2f giảm kích thước xuống 40x40x512.
  4. Stage 4: ConvModule và C2f giảm kích thước xuống 20x20x1024.
- Sppf (Spatial Pyramid Pooling – Fast): Kết thúc backbone với kích thước 20x20x512xW.

Neck của YOLOv8 kết nối backbone và head, thực hiện các tác vụ sau:

- Upsample: Tăng kích thước hình ảnh từ các giai đoạn trước.
- Concat: Nối các đặc trưng từ các giai đoạn trước đó.
- ConvModule: Kết hợp các lớp Conv để tinh chỉnh đặc trưng.

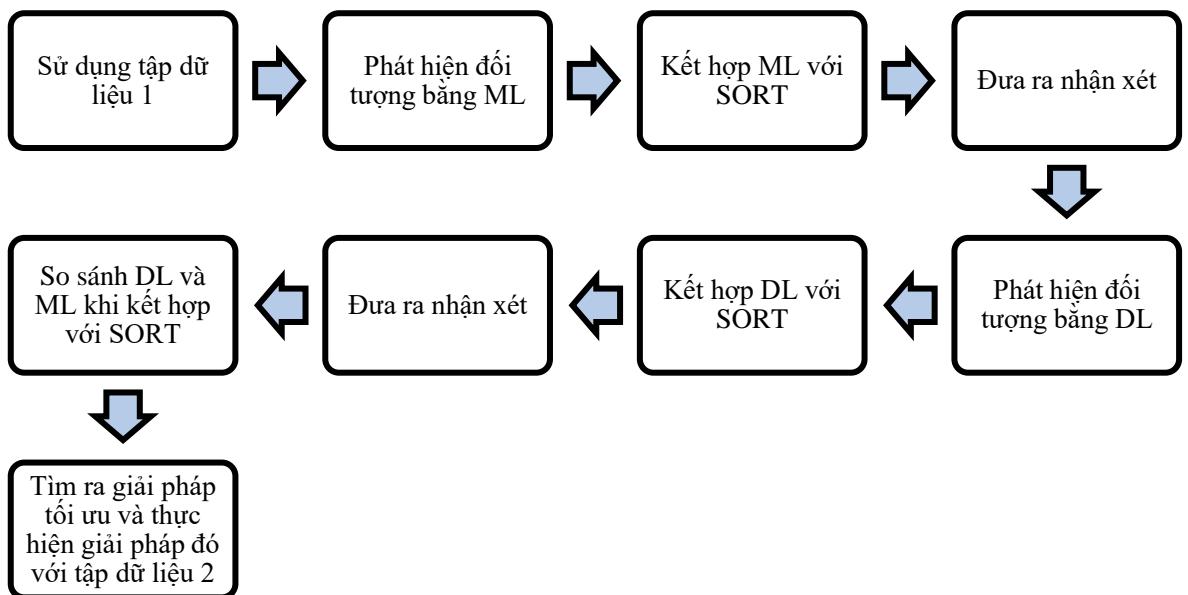
Head của YOLOv8 chịu trách nhiệm cho việc phát hiện đối tượng và phân loại:

- ConvModule: Chuyển đổi các đặc trưng từ Neck.
- BBox và Cls Loss:
  - Bbox Loss: Tính toán lỗi cho bounding box.
  - Cls Loss: Tính toán lỗi cho phân loại.
  - BCE: Binary Cross-Entropy Loss.
  - DFL: Distribution Focal Loss.

#### Các Module Phụ

- C2f (CSP Bottleneck with 2 Convolutional Layers): Sử dụng cấu trúc của DarknetBottleneck và ConvModule.
- ConvModule: Kết hợp Conv2d, BatchNorm (BN) và SiLU.
- DarknetBottleneck: Sử dụng thêm các ConvModule.
- SPPF (Spatial Pyramid Pooling – Fast): Bao gồm nhiều lớp MaxPool và Concat.

### 4.3 Thực nghiệm



Biểu đồ 5. Quy trình thực nghiệm

Mô tả: Quy trình thực nghiệm được biểu hiện bao gồm 8 bước như mô tả của biểu đồ 5 phía trên.

Cấu hình của máy thực nghiệm

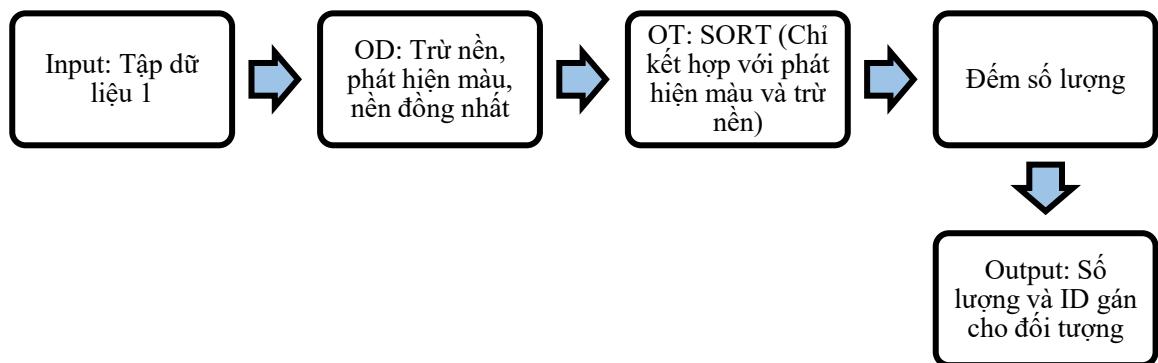
Máy 1:

- CPU: I5-12500H
- GPU: RTX 3050
- RAM: 24GB

Máy 2:

- CPU: I5-11400H
- GPU: RTX 3050
- RAM 16GB

### 4.3.1 Phát hiện đối tượng bằng Machine Learning



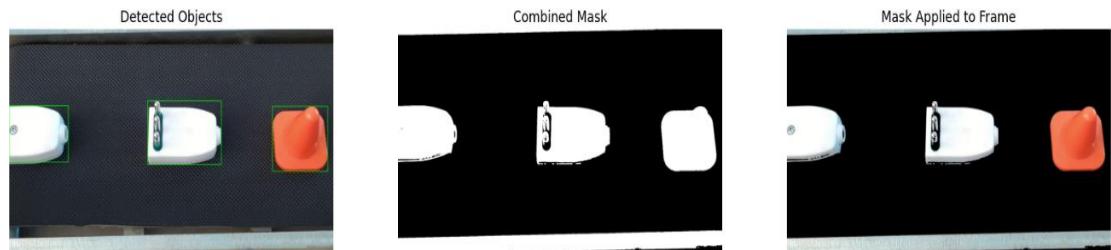
Biểu đồ 6. Quy trình thực hiện phát hiện đối tượng bằng ML

Mô tả: Quy trình thực hiện phát hiện bằng ML gồm 5 bước như biểu đồ 6 phía trên.

#### 4.3.1.1 Thuật toán phát hiện đối tượng dựa trên màu

- Các bước thực hiện:
  - Chuyển đổi ảnh BGR sang HSV.
  - Lựa chọn sắc độ màu trong khoảng có khả năng sẽ khớp với object (cam và trắng) sau đó sử dụng để tìm mask của object.
  - Để tìm contour cần dùng trên ảnh có ngưỡng nhị phân nén mask của ảnh là phù hợp.
  - Tìm contour + lọc dựa trên diện tích contour và tỷ lệ của đa số các vật thể là Width/Height < 3-4.
  - Thực hiện vẽ bbox từ các contour khớp với điều kiện.

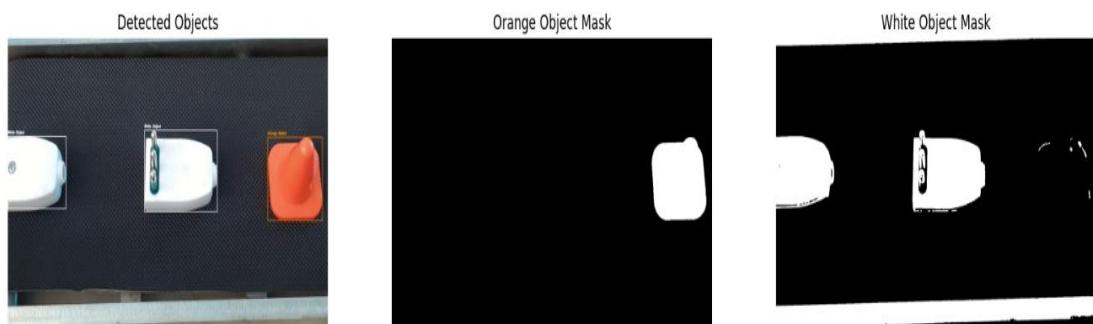
- Cách 1: Vẽ contours chung và dùng bitwise\_or



Hình 45. Vẽ contours chung cho tất cả các màu

Nhận xét: Contour được vẽ khi dùng bitwise\_or sẽ nhanh hơn, giảm phép tính vì khi bitwise\_or sẽ tạo ra 1 mask mới bao gồm cả ngưỡng lower và upper của cả 2 màu. Do cả ngưỡng của cả 2 mask được gộp thành 1 mask mới nên chỉ có thể xác định được object nằm trong khoảng này mà không thể xác định được màu sắc của object, kèm theo đó nếu gặp phải nhiễu hoặc 1 số những điểm không phải là object bị detect nhầm sẽ khó phân biệt được cần phải điều chỉnh ngưỡng nào hoặc chú ý vào những gam màu nào để xử lý.

- Cách 2: Vẽ contours riêng cho từng màu và không dùng bitwise\_or:



Hình 46. Vẽ contours riêng cho từng màu

Nhận xét: Thực hiện trên từng mask màu mà không dùng bitwise\_or gộp lại cũng cho kết quả cũng giống nhau. Điều khác ở đây là biết được chính xác màu sắc của từng vật thể, tốc độ xử lý sẽ chậm hơn đổi lại có thể làm được tốt những điều mà khi dùng bitwise\_or không làm được.

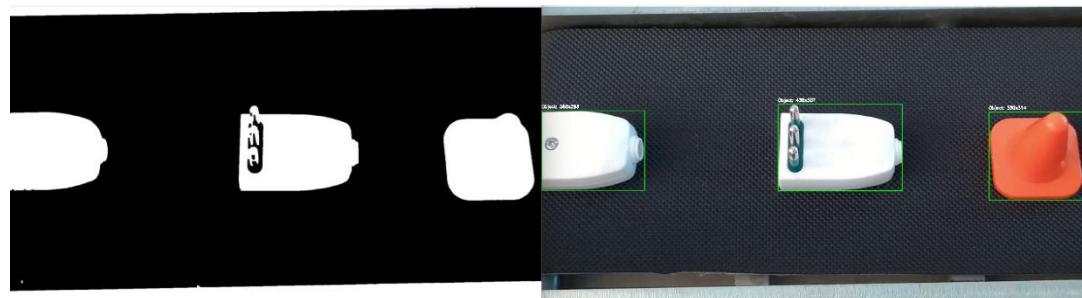
Kết luận:

Các object có màu trắng, cam đều phù hợp cho cả HSV và RGB nhưng HSV không bị ảnh hưởng nhiều như RGB về yếu tố ánh sáng do khả năng tách biệt sắc độ (H) ra khỏi độ sáng (V) và độ bão hòa (S) giúp giữ nguyên được giá trị của màu sắc từ đó phân biệt dễ dàng hơn. Có thể thấy ở các object đều có những phần sáng không đồng đều do ánh sáng không chiếu được toàn bộ object. Ở object màu cam có những phần bị sáng chói điều này sẽ ảnh hưởng nếu dùng RGB khả năng sẽ không phân biệt được màu sắc ở vùng bị chói sáng.

#### 4.3.1.2 Thuật toán phát hiện đối tượng dựa trên nền đồng nhất

- Các bước thực hiện:

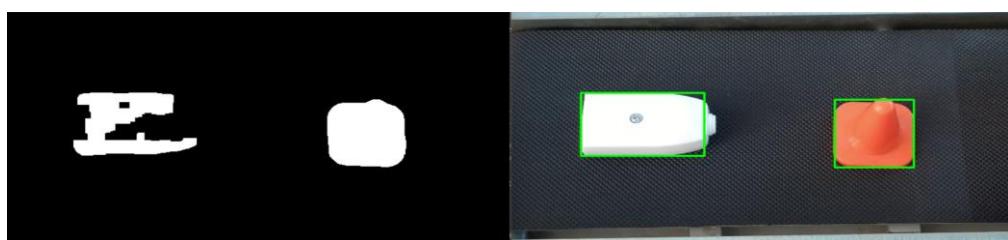
- Khởi tạo: Tạo các cửa sổ hiển thị kết quả và mở video từ tệp.
- Xử lý từng khung hình trong video.
- Chuyển đổi khung hình sang không gian màu HSV
- Phân ngưỡng động dựa trên độ sáng: Áp dụng phân ngưỡng Otsu để tạo mặt nạ nhị phân.
- Cải thiện mặt nạ bằng các phép biến đổi hình thái:
  - Tạo kernel hình thái học.
  - Áp dụng phép biến đổi đóng (closing).
  - Áp dụng phép biến đổi mở (opening).
- Làm mịn mặt nạ bằng Gaussian Blur
- Tìm và vẽ đường viền: Tìm các đường viền trong mặt nạ nhị phân.
  - Lặp qua từng đường viền: Tính diện tích của đường viền nếu diện tích lớn hơn ngưỡng thì tính hình chữ nhật bao quanh (bounding box) và nếu tỷ lệ khung hình hợp lý vẽ hình chữ nhật bao quanh và thêm nhãn đối tượng.
- Hiển thị kết quả.



Hình 47. Thực nghiệm thuật toán phát hiện đối tượng dựa trên nền đồng nhất

#### 4.3.1.3 Thuật toán phát hiện đối tượng dựa trên việc trừ nền

- Các bước thực hiện:
  - Khởi tạo: Mở video và khởi tạo bộ trừ nền.
  - Xử lý từng khung hình trong video.
  - Tiên xử lý khung hình:
    - Thay đổi kích thước khung hình cho phù hợp.
    - Áp dụng bộ lọc bilateral để giảm nhiễu muối tiêu.
    - Áp dụng Gaussian Blur để làm mịn khung hình.
  - Áp dụng phương pháp trừ nền:
    - Áp dụng bộ trừ nền để tạo mặt nạ nền trừ.
    - Phân ngưỡng để tạo mặt nạ nhị phân.
  - Cải thiện mặt nạ bằng các phép biến đổi hình thái:
    - Tạo kernel hình thái học.
    - Áp dụng phép biến đổi đóng (closing).
    - Áp dụng phép biến đổi mở (opening).
    - Thực hiện dilation và erosion để hậu xử lý mặt nạ.
  - Tìm và vẽ đường viền: Tìm các đường viền trong mặt nạ nhị phân.
    - Lặp qua từng đường viền: Tính diện tích của đường viền nếu diện tích lớn hơn ngưỡng thì tính hình chữ nhật bao quanh (bounding box) và nếu tỷ lệ khung hình hợp lý vẽ hình chữ nhật bao quanh và thêm nhãn đối tượng.
  - Hiển thị kết quả.



Hình 48.Thực nghiệm thuật toán phát hiện đối tượng dựa trên việc trừ nền

---

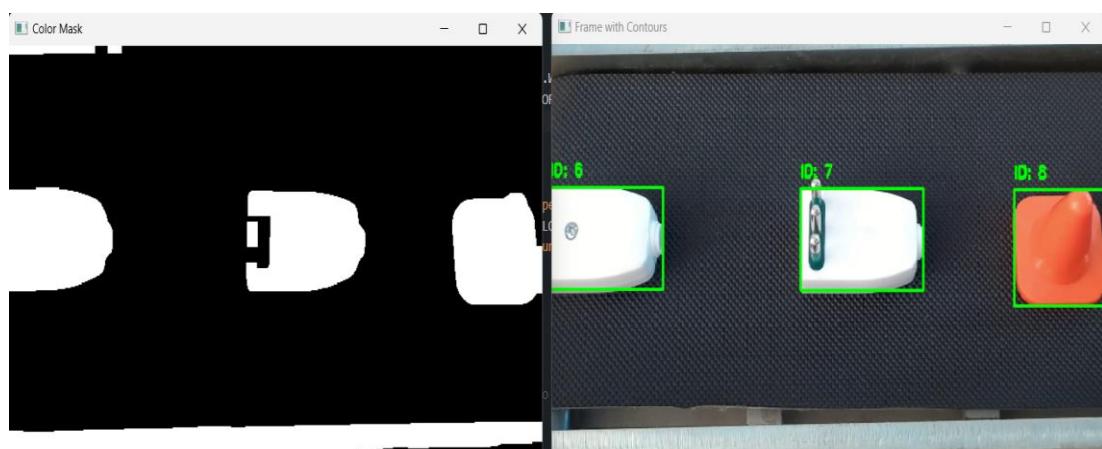
#### 4.3.1.4 Kết hợp phát hiện đối tượng Machine Learning với SORT

Sử dụng SORT kết hợp với thuật toán phát hiện đối tượng dựa trên màu:

Các bước thực hiện bao gồm:

- Bước 1: Khởi tạo
  - Tải thư viện cần thiết:
  - Tải các thư viện OpenCV và NumPy.
  - Tải thư viện SORT tracker.
  - Mở video: Mở file video cần xử lý, kiểm tra xem video có thể mở không.
- Bước 2: Đọc Video và Tiên xử lý
  - Đọc từng khung hình: Đọc từng khung hình từ video.
  - Nếu không đọc được khung hình, kết thúc xử lý.
  - Tiên xử lý khung hình: Giảm kích thước khung hình để tăng tốc độ xử lý và áp dụng Gaussian Blur để giảm nhiễu.
- Bước 3: Phát hiện Màu Sắc
  - Thiết lập ngưỡng cho màu sắc: Thiết lập ngưỡng màu sắc cho các màu cần phát hiện (trắng và cam).
  - Tạo mặt nạ từ ngưỡng màu sắc: Chuyển đổi khung hình từ không gian màu BGR sang HSV và tạo mặt nạ từ ngưỡng màu sắc.
  - Biến đổi hình thái để làm sạch mặt nạ: Áp dụng các phép biến đổi hình thái để làm sạch mặt nạ.
- Bước 4: Tìm và Xử lý Contours
  - Tìm contours: Tìm các contours từ các mặt nạ đã được làm sạch.
  - Lọc contours và chuẩn bị danh sách phát hiện cho SORT: Lọc các contours dựa trên diện tích, tỉ lệ khung hình thêm các bounding boxes của các contours vào danh sách phát hiện.

- Bước 5: Cập nhật SORT Tracker
  - Cập nhật SORT tracker với các phát hiện mới: Cập nhật SORT tracker với các bounding boxes từ Color Detection.
- Bước 6: Vẽ Bounding Boxes và ID
  - Vẽ bounding boxes và ID lên khung hình: Vẽ bounding boxes và ID của đối tượng lên khung hình và hiển thị kết quả.
- Bước 7: Hoàn tất



Hình 49. Kết quả khi kết hợp SORT và phát hiện đối tượng dựa trên màu  
Kết quả thực nghiệm với Sort và ROI:



Hình 50. Lỗi khi kết hợp thêm ROI vào SORT với phát hiện dựa trên màu

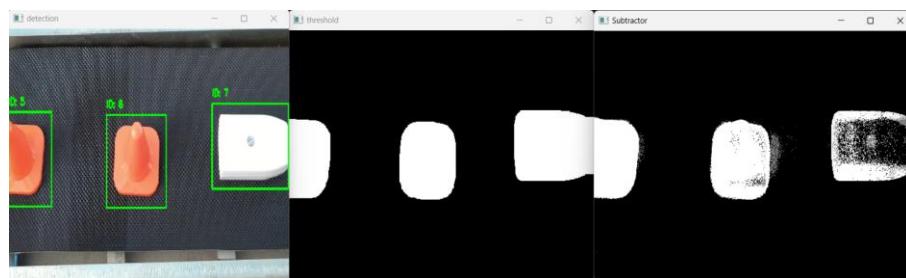
Nhận xét: Việc phát hiện và đếm đối tượng được thực hiện khá ổn có tốc độ xử lý nhanh, vẫn có lỗi sai sót do bounding box khoanh vùng background nhưng việc đếm số lượng vật thể được thực hiện chính xác. Tuy nhiên dữ liệu trên là tập dữ liệu 1 nên có độ đa dạng thấp nên việc tìm ngưỡng và xác định đối tượng trên ngưỡng màu hoạt động rất tốt, nếu dữ liệu được thay thế phức tạp hơn nhiều màu sắc hơn thì lỗi sai sẽ rất nhiều và không phù hợp cho việc sử dụng thuật toán này như trên hình 56 ta cũng thấy được giữa object 1 và 3 nó không thể phân biệt được. Chưa kể đến sẽ phải xét kích thước và lớp của vật thể trên bằng chuyền thì thuật toán này không khả thi.

Sử dụng SORT kết hợp với thuật toán trù nền:

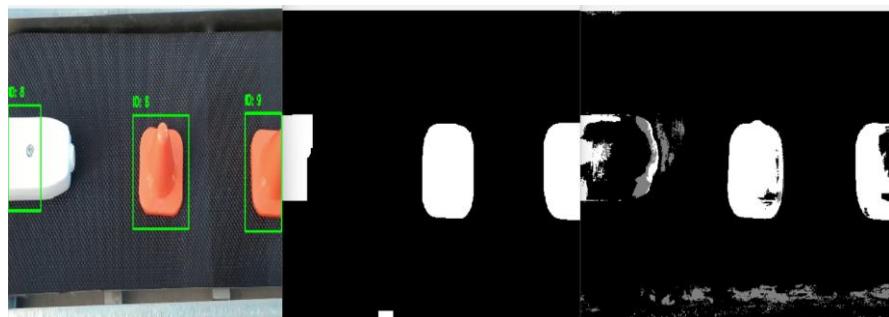
Các bước thực hiện:

- Bước 1: Khởi tạo bộ trù nền và SORT tracker
  - bg\_subtractor sử dụng KNN hoặc MOG2 để phát hiện các đối tượng di chuyển bằng cách trù nền.
  - tracker là đối tượng của SORT để theo dõi các đối tượng qua các khung hình.
- Bước 2: Xử lý video
  - Mở video và lặp qua từng khung hình.
  - Giảm nhiễu bằng Gaussian Blur.
  - Áp dụng background subtraction để lấy mặt nạ tiền cảnh.
  - Chuyển đổi mặt nạ tiền cảnh thành ảnh nhị phân và làm sạch bằng các phép biến đổi hình thái học.
- Bước 3: Phát hiện và theo dõi đối tượng
  - Tìm các contours trong ảnh.
  - Lọc các contours dựa trên diện tích và tỷ lệ khung hình để xác định các bounding boxes của đối tượng.
  - Cập nhật SORT tracker với các bounding boxes này để theo dõi các đối tượng qua các khung hình.

- Bước 4: Vẽ Bounding Boxes và ID
  - Vẽ các bounding boxes và ID của đối tượng lên khung hình hiện tại.
- Bước 5: Hiển thị kết quả
  - Hiển thị khung hình hiện tại với các bounding boxes và ID của đối tượng.
- Bước 6: Hoàn tất

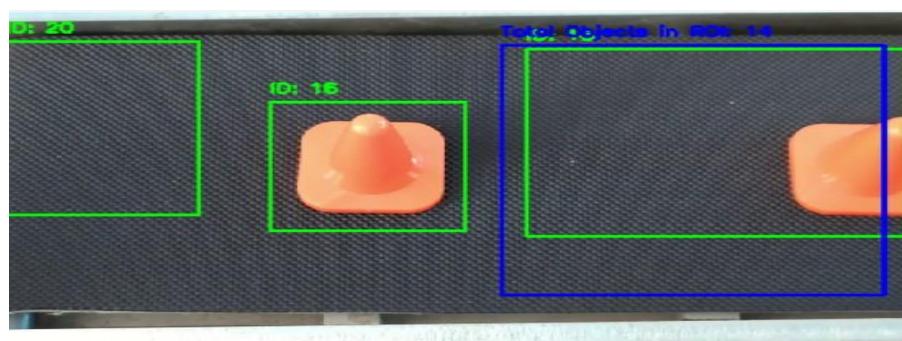


Hình 51. Kết quả khi thực hiện Sort và thuật toán trừ nền(KNN)



Hình 52. Kết quả khi thực hiện Sort và thuật toán trừ nền(MOG2)

Kết quả thực nghiệm khi kết hợp cả SORT và ROI:

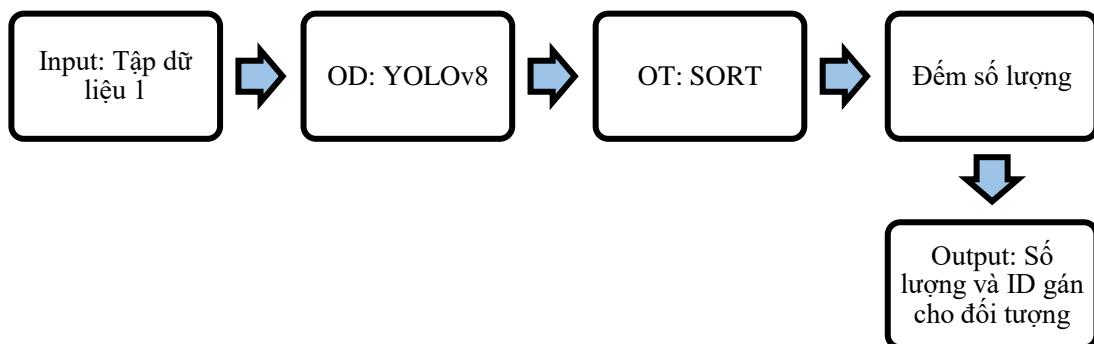


Hình 53. Lỗi khi thực hiện SORT, ROI và trừ nền

---

Nhận xét: Rất nhiều bounding box bị đánh sai vị trí dẫn tới ID khi theo vết với SORT sai theo. Điều này có thể là do ngưỡng sáng và các thuật toán biến đổi hình thái chưa thực hiện tốt tuy nhiên nếu áp dụng vào thực tế thì cần phải mô trường lý tưởng mới có thể thực hiện nó một cách tốt. Tuy rằng số lượng vật thể đếm được giống với thuật toán trên, độ phù hợp với thực tế cũng cao hơn nhưng đối với bài toán băng chuyền thì không hoàn toàn phù hợp do dễ nhầm lẫn vật thể với các vùng background khác đặc biệt là băng chuyền cùng với đó việc gắn nhãn cụ thể cho bounding box không thực hiện được. Do vậy thuật toán trên mặc dù thực tế hơn phát hiện màu nhưng không phù hợp dẫn đến rất nhiều lỗi sai.

#### 4.3.2 Phát hiện đối tượng bằng Deep Learning



##### 4.3.2.1 Phát hiện đối tượng bằng YOLOv8 (Deep Learning)

- Các siêu tham số dùng train mô hình cho bộ dữ liệu 1:
  - Epoch: 40
  - Batch size: 16
  - Image size: 640
  - Learning rate: 0.01
  - Optimizer: auto

- Các bước thực hiện:

- Cắt frame ảnh từ video conveyor\_belt.mp4 với độ dài 5 khung hình trên giây.
- Tạo nhãn cho toàn bộ dữ liệu ảnh thu thập được
- Chia dữ liệu ảnh và dữ liệu nhãn theo tỷ lệ 8.1.1( 80% train, 10% valid, 10 test) .
- Cấu hình file yaml để xác định lớp và đường dẫn cho từng tập train, valid, test.
- Import thư viện ultralytic và các thư viện cần thiết.
- Load model YOLOv8 lên.
- Chọn đường dẫn data là file yaml đã cấu hình.
- Train và hiển thị kết quả.



Hình 54. Thực nghiệm phát hiện đối tượng bằng YOLOv8 với tập dữ liệu 1

#### 4.3.2.2 Kết hợp YOLOv8 (Deep Learning) với SORT

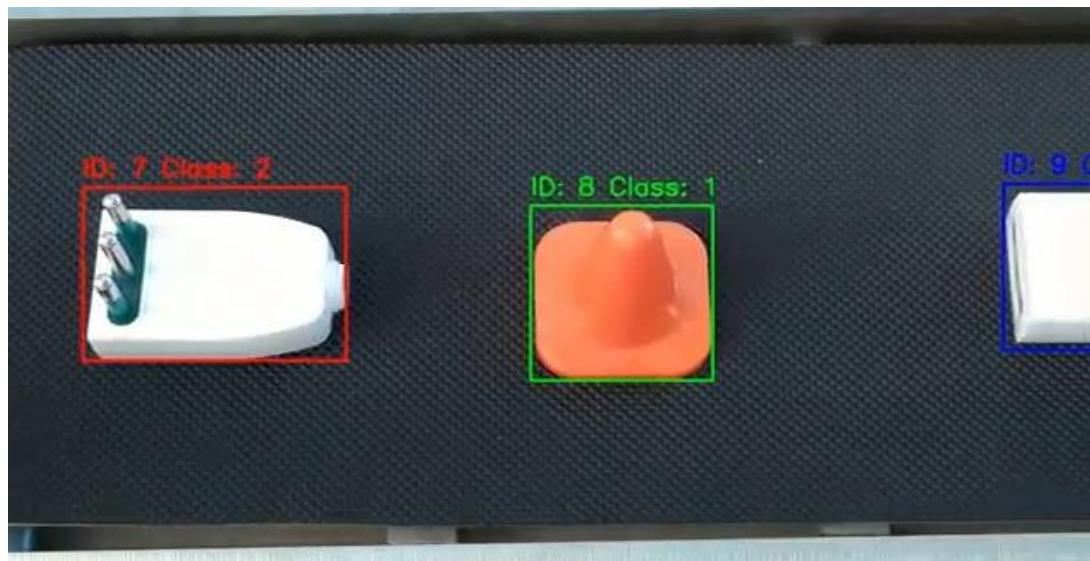
Các bước thực hiện bao gồm:

- Bước 1: Khởi tạo

- Tải mô hình YOLO đã được huấn luyện (ở đây là YOLOv8 Custom Data).

- Khởi tạo bộ theo dõi SORT với các tham số điều chỉnh: max\_age, min\_hits, và iou\_threshold.
- Bước 2: Đọc Video và Tiền xử lý
  - Mở file video cần xử lý.
  - Thiết lập bộ ghi video để lưu kết quả.
- Bước 3: Xử lý từng khung hình
  - Đọc từng khung hình từ video.
  - Nếu không đọc được khung hình, kết thúc xử lý.
  - Tăng bộ đếm khung hình.
  - Nếu khung hình không cần xử lý (dựa trên frame\_skip), tiếp tục với khung hình tiếp theo.
- Bước 4: Phát hiện đối tượng với YOLO
  - Sử dụng mô hình YOLO để phát hiện các đối tượng trong khung hình.
  - Lưu trữ các phát hiện dưới dạng danh sách các bounding boxes và độ tin cậy (score).
- Bước 5: Cập nhật SORT Tracker
  - Cập nhật SORT tracker với các bounding boxes từ YOLO.
  - Lưu trữ các bounding boxes đã được gán ID.
- Bước 6: Vẽ Bounding Boxes và ID
  - Vẽ bounding boxes và ID của đối tượng lên khung hình.
  - Ghi khung hình đã xử lý vào file video đầu ra.
- Bước 7: Hoàn tất

Có một lưu ý nhỏ khi thực hiện trên Google Colab chúng ta cần phải đổi dòng code này “matplotlib.use('TkAgg')” thành “matplotlib.use('nbAgg')” do khi chạy TkAggs sẽ bị lỗi do Google Colab không hỗ trợ.



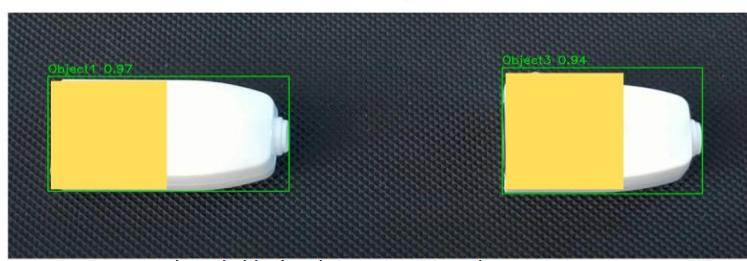
Hình 55. Kết quả khi kết hợp YOLO với SORT

Sử dụng Sort kết hợp với YOLOv8 và vùng ROI để thực hiện đếm đối tượng:

Do việc train YOLOv8 với tập dữ liệu và khi label Object 1 và Object 3 khá giống nhau khi bị che một nửa dưới đây là hình minh họa và Yolo sẽ có thể xác định vật thể sai.



Hình ảnh của Object 1 và 3



Hình ảnh khi bị che 1 nửa của Object 1 và 3

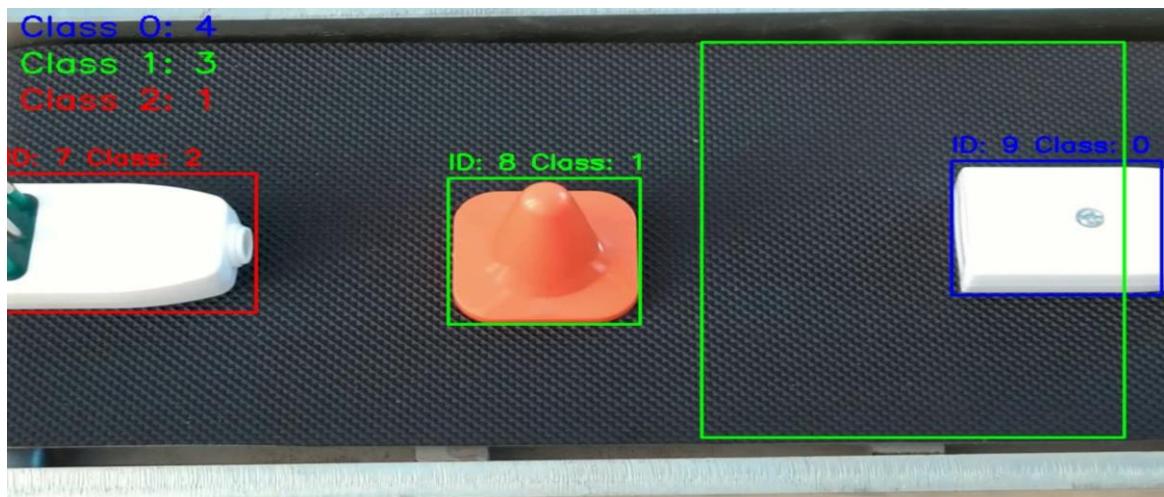
Hình 56. So sánh Object 1 và 3



Hình 57. Hình ảnh Yolo xác định sai khi Object 3 bị che 1 nửa

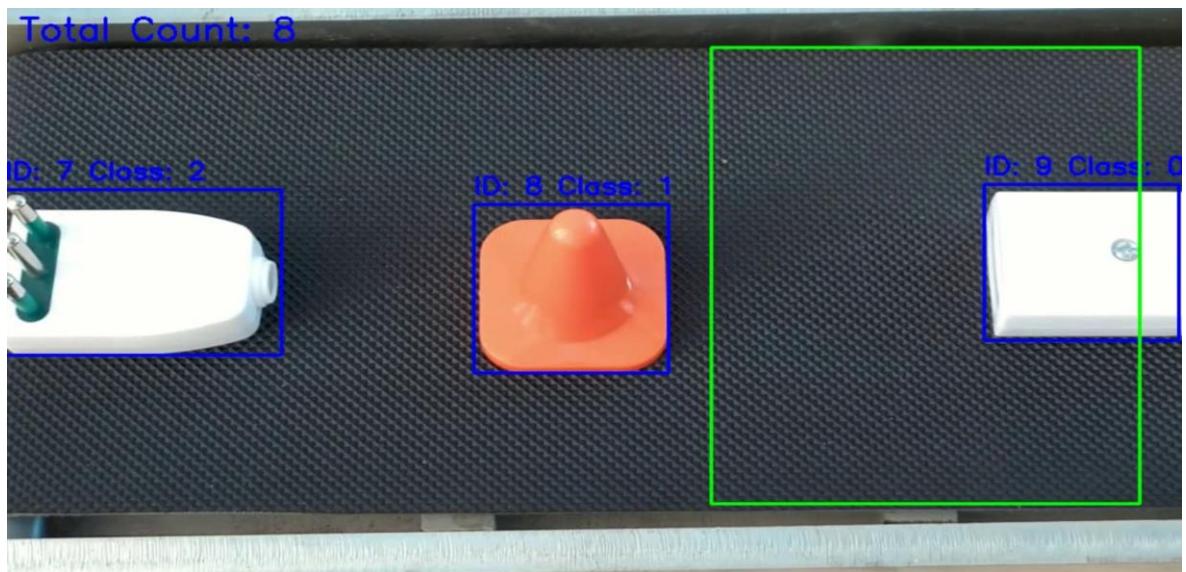
Nên khi thực hiện Sort với ROI nhóm đã thực hiện là là thêm NMS và khi sử dụng NMS thì nhận thấy rằng kết quả khi sử dụng ID sẽ không bị nhảy và dẫn đến sai ID dù class có nhận dạng sai một lúc rồi giật lại đúng class kết quả cho lại giúp track theo đối tượng ổn hơn. Có tổng cộng là 18 đối tượng trong 3 class và số lượng đối tượng trong class 0 là chiếm số lượng nhiều nhất và ít nhất là class 2 chỉ có 1 đối tượng.

Đếm đối tượng theo số lượng có phân loại để đếm số lượng từng đối tượng: Được chia theo Class 0 tương ứng với Object1 và Class 1 tương ứng với Object 2 và Class 2 tương ứng với Object 3:



Hình 58. SORT và ROI với Yolov8 và ROI hình chữ nhật có phân loại

Đếm đối tượng theo số lượng không phân loại để đếm tổng số lượng của tất cả đối tượng :



Hình 59. SORT và ROI với Yolov8 và ROI hình chữ nhật không phân loại  
Các bước thực hiện của việc kết hợp giữa Yolov8 và SORT và vẽ ROI hình chữ nhật:

#### Bước 1: Khởi Tạo Môi Trường

- Cài Đặt Thư Viện: Đảm bảo cài đặt các thư viện opencv-python, numpy, ultralytics.
- Đọc Video và Khởi Tạo Model: Đọc video đầu vào và khởi tạo model YOLOv8 cùng với SORT tracker.

#### Bước 2: Định Nghĩa ROI Hình Chữ Nhật

- Chọn Tọa Độ của ROI: Xác định các tọa độ (x1, y1) và (x2, y2) của hai góc đối diện để tạo thành ROI.
- Điều Chỉnh Tọa Độ: Điều chỉnh các tọa độ này cho phù hợp với kích thước khung hình nếu cần.

#### Bước 3: Xử Lý Từng Khung Hình

- Đọc Khung Hình: Đọc từng khung hình từ video.

- Giảm Kích Thước Khung Hình: Nếu cần giảm kích thước khung hình để tối ưu hóa xử lý.
- Điều Chỉnh Tọa Độ ROI: Điều chỉnh tọa độ ROI theo kích thước khung hình hiện tại sau khi giảm kích thước.

#### Bước 4: Phát Hiện Đối Tượng

- Inference YOLOv8: Sử dụng model YOLOv8 để phát hiện các đối tượng trong khung hình.
- Non-Max Suppression: Sẽ áp dụng Non-Max Suppression để loại bỏ các bounding box trùng lặp.

#### Bước 5: Theo Dõi Đối Tượng

- Update SORT: Cập nhật SORT tracker với các phát hiện.
- Lưu Thông Tin Lớp: Lưu thông tin lớp cho mỗi đối tượng mới phát hiện vào class\_dict.

#### Bước 6: Đếm Đối Tượng Trong ROI

- Kiểm Tra Bounding Box Trong ROI: Kiểm tra xem toàn bộ bounding box có nằm trong ROI hình chữ nhật hay không.

#### Đếm Đối Tượng:

- Phân Lớp và Đếm Theo Từng Lớp: Đếm số lượng đối tượng theo từng lớp đi qua ROI và lưu vào roi\_counts.
- Không Phân Lớp và Đếm Tổng Số Lượng: Đếm tổng số lượng đối tượng đi qua ROI và lưu vào total\_count.

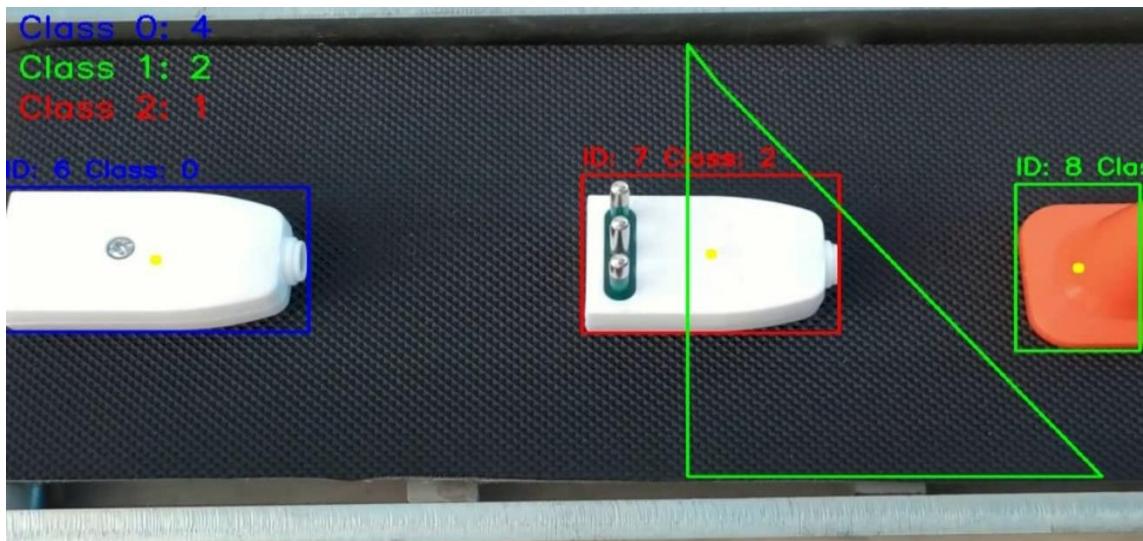
#### Bước 7: Vẽ Bounding Box và ROI

- Vẽ Bounding Box: Vẽ bounding box và ID của đối tượng lên khung hình.
- Vẽ ROI: Vẽ ROI hình chữ nhật lên khung hình.

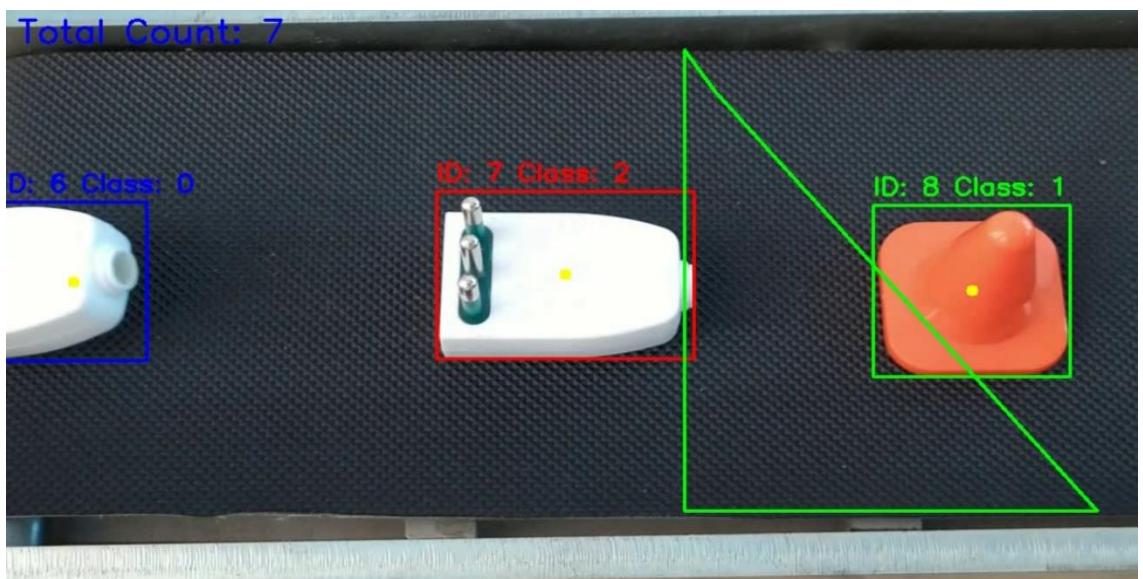
#### Bước 8: Hiển Thị Thông Tin và Lưu Video

---

- Hiển Thị Thông Tin Đếm:
  - Phân Lớp và Đếm Theo Từng Lớp: Hiển thị thông tin về số lượng đối tượng theo từng lớp lên khung hình.
  - Không Phân Lớp và Đếm Tổng Số Lượng: Hiển thị thông tin về tổng số lượng đối tượng lên khung hình.
- Lưu Khung Hình: Ghi khung hình vào file video đầu ra.



Hình 60. SORT và ROI với Yolov8 và ROI đa giác có phân loại



Hình 61. SORT và ROI với Yolov8 và ROI đa giác không phân loại

Các bước thực hiện của việc kết hợp giữa Yolov8 và SORT và vẽ ROI đa giác:

Bước 1: Khởi Tạo Môi Trường

- Cài Đặt Thư Viện: Đảm bảo cài đặt các thư viện opencv-python, numpy, ultralytics.
- Đọc Video và Khởi Tạo Model: Đọc video đầu vào và khởi tạo model YOLOv8 cùng với SORT tracker.

Bước 2: Định Nghĩa ROI Polygon

- Chọn Tọa Độ của Các Điểm: Xác định các tọa độ (x, y) của các đỉnh đa giác để tạo thành ROI.
- Điều Chỉnh Tọa Độ: Điều chỉnh các tọa độ này cho phù hợp với kích thước khung hình nếu cần.

Bước 3: Xử Lý Từng Khung Hình

- Đọc Khung Hình: Đọc từng khung hình từ video.
- Giảm Kích Thước Khung Hình: Nếu cần, giảm kích thước khung hình để tối ưu hóa xử lý.
- Điều Chỉnh Tọa Độ ROI: Điều chỉnh tọa độ ROI theo kích thước khung hình hiện tại sau khi giảm kích thước.

Bước 4: Phát Hiện Đối Tượng

- Inference YOLOv8: Sử dụng model YOLOv8 để phát hiện các đối tượng trong khung hình.
- Non-Max Suppression: Sẽ áp dụng Non-Max Suppression để loại bỏ các bounding box trùng lặp.

Bước 5: Theo Dõi Đối Tượng

- Update SORT: Cập nhật SORT tracker với các phát hiện.

- Lưu Thông Tin Lớp: Lưu thông tin lớp cho mỗi đối tượng mới phát hiện vào class\_dict.

#### Bước 6: Đếm Đối Tượng Trong ROI

- Kiểm Tra Trọng Tâm Trong ROI: Kiểm tra xem trọng tâm của bounding box có nằm trong ROI polygon hay không.
- Phân Lớp và Đếm Theo Từng Lớp: Đếm số lượng đối tượng theo từng lớp đi qua ROI và lưu vào roi\_counts.
- Không Phân Lớp và Đếm Tổng Số Lượng: Đếm tổng số lượng đối tượng đi qua ROI và lưu vào total\_count.

#### Bước 7: Vẽ Bounding Box và ROI

- Vẽ Bounding Box: Vẽ bounding box và ID của đối tượng lên khung hình.
- Vẽ ROI: Vẽ ROI polygon lên khung hình.

#### Bước 8: Hiển Thị Thông Tin và Lưu Video

- Hiển Thị Thông Tin Đếm:
  - Phân Lớp và Đếm Theo Từng Lớp: Hiển thị thông tin về số lượng đối tượng theo từng lớp lên khung hình.
  - Không Phân Lớp và Đếm Tổng Số Lượng: Hiển thị thông tin về tổng số lượng đối tượng lên khung hình.
- Lưu Khung Hình: Ghi khung hình vào file video đầu ra.

Nhận xét: Đây là phương pháp phù hợp nhất nó có thể xác định, theo dõi chính xác và phân lớp chính xác các vật thể. Nó có ưu điểm là nhanh hơn các mô hình học sâu cũ nhưng tốc độ xử lý so với các thuật toán machine learning truyền thống thì không bằng nhưng bù lại độ chính xác cao hơn và xác với thực tế hơn. Điểm ảnh hưởng chính là cấu hình thực nghiệm chỉ cần phần cứng ổn thì hiệu quả mang lại sẽ cao hơn những phương pháp truyền thống. Vì thế đây chính là phương pháp mà nhóm sẽ lựa chọn để thực hiện với tập dữ liệu 2 là tập dữ liệu thực tế mà nhóm tự thu thập.

---

#### 4.3.2.3 So sánh đánh giá tổng quan hai phương pháp

Các phương pháp thực nghiệm	Số lần Tracking sai	Runtime	Phần cứng sử dụng	Ưu điểm	Nhược điểm
Phát hiện màu kết + SORT (ML)	0	57.33s	CPU	Tốc độ xử lý nhanh	Nhạy cảm với các yếu tố môi trường
Trừ nền + SORT (ML)	5	71.06s	CPU	Tốc độ xử lý nhanh	Nhạy cảm với các yếu tố môi trường
YOLOv8 + SORT (DL)	0	500.12s	CPU	Rất chính xác	Tốn tài nguyên
YOLOv8 + SORT (DL)	0	200.6s	GPU	Rất chính xác	Tốn tài nguyên, yêu cầu phần cứng cao hơn

Bảng 3. Bảng so sánh các phương pháp khi không phân loại

Class	Images	Instances	Precision (P)	Recall (R)	mAP50	mAP50-95
All	119	239	0.97	0.997	0.994	0.985
Object1	91	125	0.94	0.992	0.993	0.976
Object2	78	95	0.969	1.0	0.995	0.987
Object3	19	19	1.0	1.0	0.995	0.99

Bảng 4. Bảng độ chính xác của YOLOv8 có phân loại

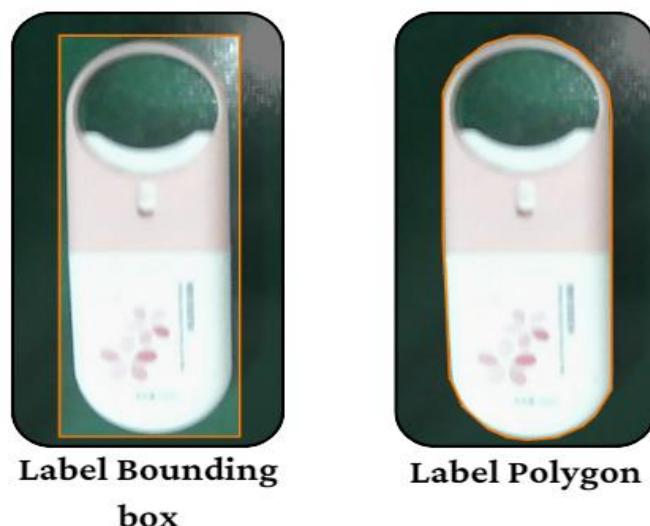
---

Nhận xét: Có thể thấy YOLOv8 kết hợp với SORT và ROI là phương pháp tối ưu nhất và tốt nhất nó có độ chính xác cao và không bị ảnh hưởng bởi các tác động bên ngoài nhiều sẽ phù hợp với bài toán theo thời gian thực và phù hợp với sản phẩm demo của nhóm hơn.

#### 4.4 Thực nghiệm với tập dữ liệu 2

Trong phần này đây chính là kết quả đề xuất chính của nhóm em nhằm cho việc thực hiện triển khai thực tế. Do đã nhận thấy được những ràng buộc và khó khăn khi thực hiện các phương pháp với tập dữ liệu 1 nên khi thực hiện ở bộ dữ liệu tự thu thập nhóm em sẽ không sử dụng các phương pháp truyền thống như trừ nền hoặc phát hiện dựa trên màu mà tiến hành chính thức với phương pháp hiện đại là YOLOv8 kết hợp với SORT và vùng ROI hình chữ nhật để đếm đối tượng và tính kích thước của nó dựa trên những tiền đề đã thực hiện ở tập dữ liệu 1.

Ở đây nhóm sẽ thực hiện cả 2 cách đánh nhãn dữ liệu được thực hiện ở Roboflow bao gồm cả việc đánh nhãn theo cách bình thường ở các bài toán xác định đối tượng đó là theo kiểu bounding box và cách còn lại là sẽ đánh nhãn theo kiểu polygon.



Hình 62. Hiển thị đánh nhãn 2 cách

## Chương 4. Giải pháp đề xuất

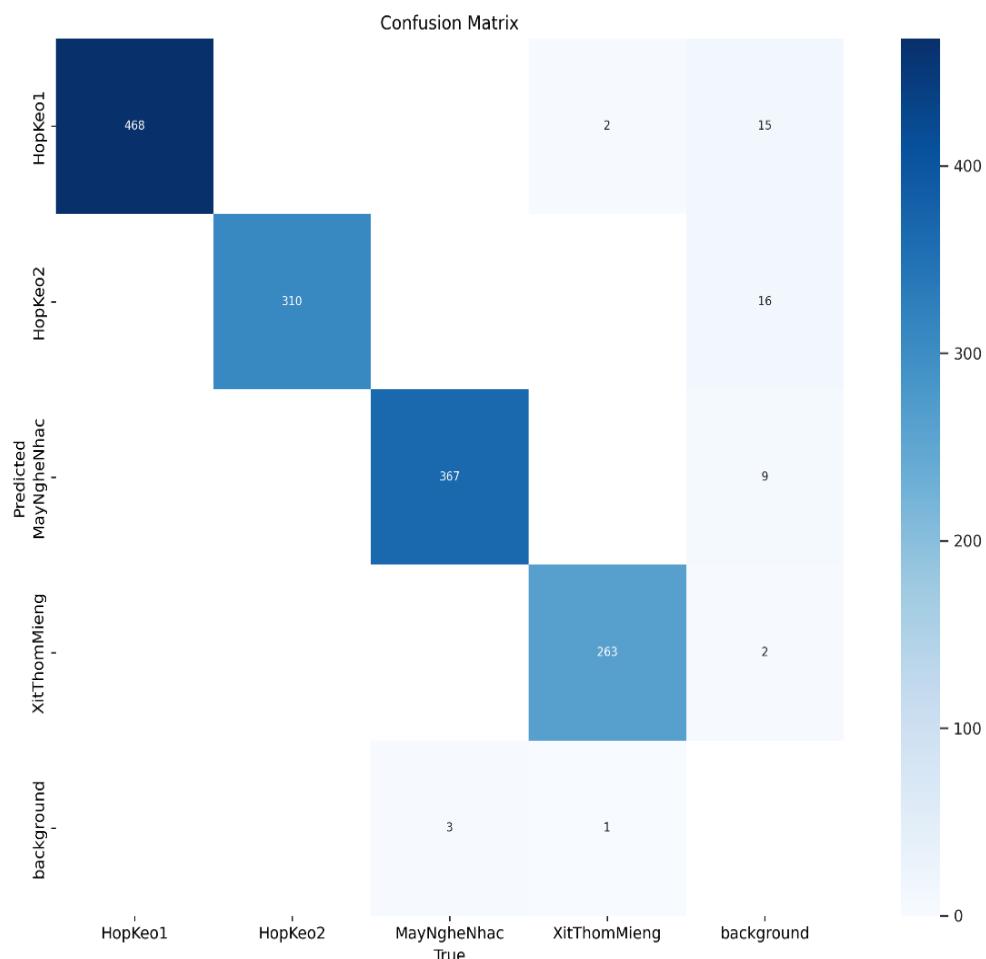
---

Sau khi đánh nhãn nhôm sẽ tiến hành train model dựa trên model YOLOv8-n.

Các siêu tham số được dùng để train cho cả 2 cách label dữ liệu:

- Epoch: 60
- Batch size: 16
- Image size: 640
- Learning rate: 0.01
- Optimizer: auto

Cách đánh nhãn thứ nhất sẽ thực hiện ở mô hình 1 sau khi train ta có thể xem ma trận nhầm lẫn ở dưới đây.

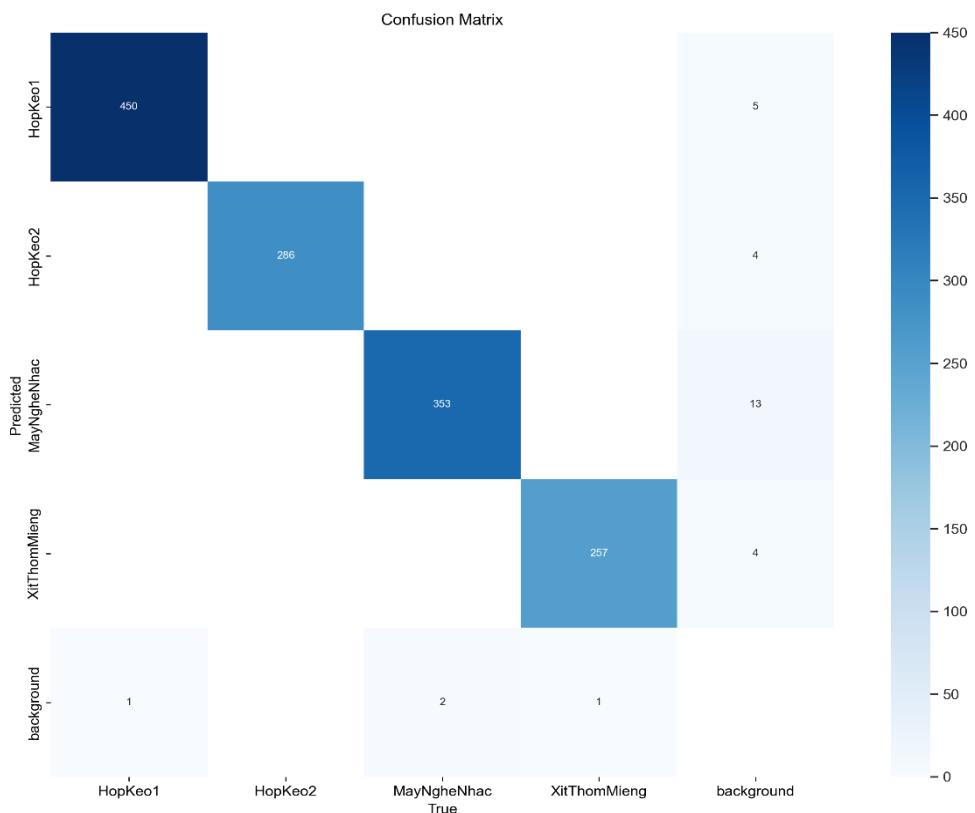


Hình 63. Ma trận nhầm lẫn của mô hình 1

Các nhãn khi được đánh nhãn theo cách một được phân loại bao gồm HopKeo1, HopKeo2, MayNgheNhac, XitThomMieng và nền (background).

Kết quả cho thấy mô hình có hiệu suất khá cao khi phân loại đúng số lượng lớn các mẫu của từng nhãn. Cụ thể thì mô hình đã phân loại đúng 468 mẫu HopKeo1, 310 mẫu HopKeo2, 367 mẫu MayNgheNhac, 263 mẫu XitThomMieng và 1 mẫu nền. Tuy nhiên vẫn tồn tại một số lỗi phân loại như 15 mẫu của HopKeo1 bị nhầm lẫn thành background và 9 mẫu của MayNgheNhac bị nhầm lẫn thành background.

Sau cách đánh nhãn thứ nhất thì sẽ đến cách đánh nhãn thứ 2 và sẽ thực hiện ở mô hình 2 sau khi train ta có thể xem ma trận nhầm lẫn ở dưới đây.



Hình 64. Ma trận nhầm lẫn của mô hình 2

Các nhãn khi được đánh nhãn theo cách hai được phân loại bao gồm HopKeo1, HopKeo2, MayNgheNhac, XitThomMieng và nền (background).

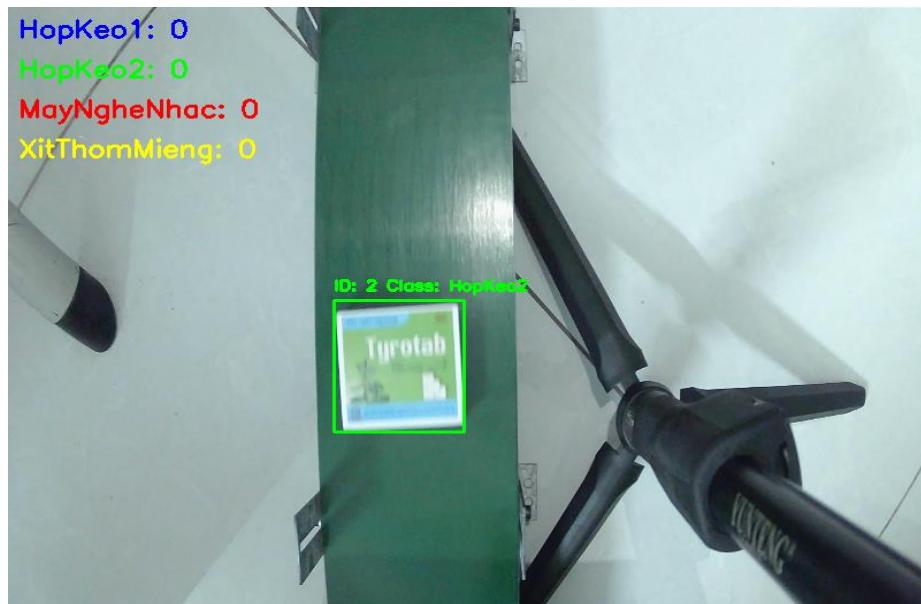
---

Kết quả từ ma trận nhầm lẫn cho thấy mô hình có hiệu suất khá cao khi phân loại đúng số lượng lớn các mẫu của từng nhãn. Cụ thể, mô hình đã phân loại đúng 450 mẫu của nhãn HopKeo1, 286 mẫu của nhãn HopKeo2, 353 mẫu của nhãn MayNgheNhac, 257 mẫu của nhãn XitThomMieng, và 1 mẫu của nhãn background. Tuy nhiên vẫn tồn tại một số lỗi phân loại như 5 mẫu của HopKeo1 bị nhầm lẫn thành background và 13 mẫu của MayNgheNhac bị nhầm lẫn thành background.

Có thể thấy hai mô hình khá tương đồng về hiệu năng do sử dụng chung tập dữ liệu được chia theo cùng tỷ lệ và chỉ khác cách đánh nhãn nên bước tiếp theo nhóm sẽ thử khả năng xem mô hình nào tạo ra bouding box chính xác với vật thể hơn để tạo ra tiền đề cho việc tính kích thước và giảm độ sai lệch do bouding box không fit với vật thể.



Hình 65. Test với mô hình 1



Hình 66. Test với mô hình 2

Kết quả cho thấy mô hình hai với cách label polygon thì nó sẽ cho hiệu xuất tốt hơn bounding box sẽ fit với vật thể hơn nên nhóm sẽ sử dụng mô hình hai để thực hiện bước tiếp theo để tính pixel\_per\_metric để thực hiện tính kích thước vật thể.

Bước tiếp theo nhóm sẽ thực hiện tính pixel\_per\_metric của object tham chiếu.



Hình 67. Vật thể tham chiếu

Đây chính là vật thể HopKeo2 có kích thước thực tế chiều rộng là 5.6cm và chiều cao là 4.8cm sau đó nhóm tiến hành đặt góc quay cố định và tính ra được pixel\_per\_metric là 67.81 pixel = 1cm (kích thước này có thể thay đổi nếu nhóm có sự chỉnh lại camera và điều chỉnh góc quay) .

---

Bước thứ ba sau khi tính được pixel\_per\_metric nhóm sẽ đưa vào và tính thử chiều rộng và chiều cao của bounding box của vật thể HopKeo có kích thước thực tế chiều rộng là 6.2cm và chiều cao là 5cm và nó trả lại kết quả khá chính xác



Hình 68. Hình ảnh khi vật thể thực hiện tính kích thước

Có thể thấy được kích thước gần chính xác và chỉ sai lệch rất ít. Bước cuối thì nhóm thực hiện chạy nhiều lần thử để xem tracking có đúng và có đếm được object khi toàn bộ boundingbox của vật thể đi vào vùng ROI không.

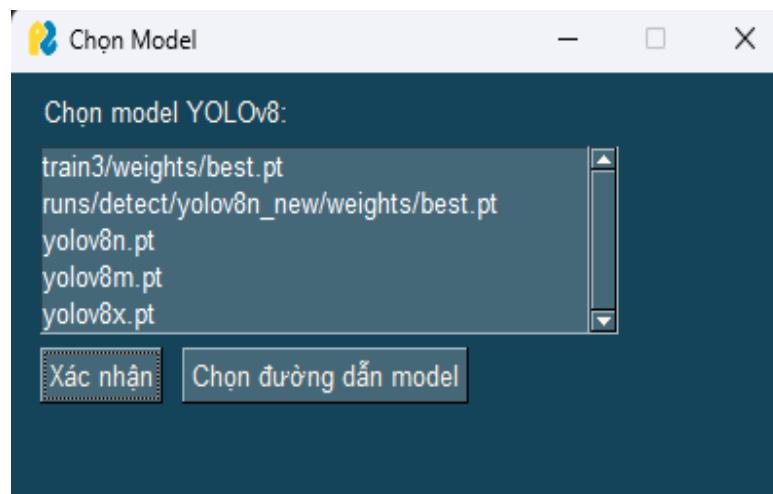


Hình 69. Hình ảnh thực nghiệm với tập dữ liệu 2

Kết quả cho ra khá tốt và có thể thấy được mô hình được nhóm huấn luyện có khả năng phát hiện đối tượng và tracking bằng SORT cũng có thể thấy ID của đối tượng có sự thay đổi và có thể đếm được số lượng đối tượng đi qua vùng ROI. Cũng thể

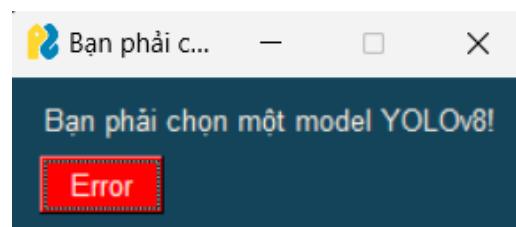
hiện được kích thước của vật thể một cách gần chính xác. Sau đó thì nhóm tiến hành xây dựng ứng dụng.

#### 4.4.1.1 Giao diện ứng dụng



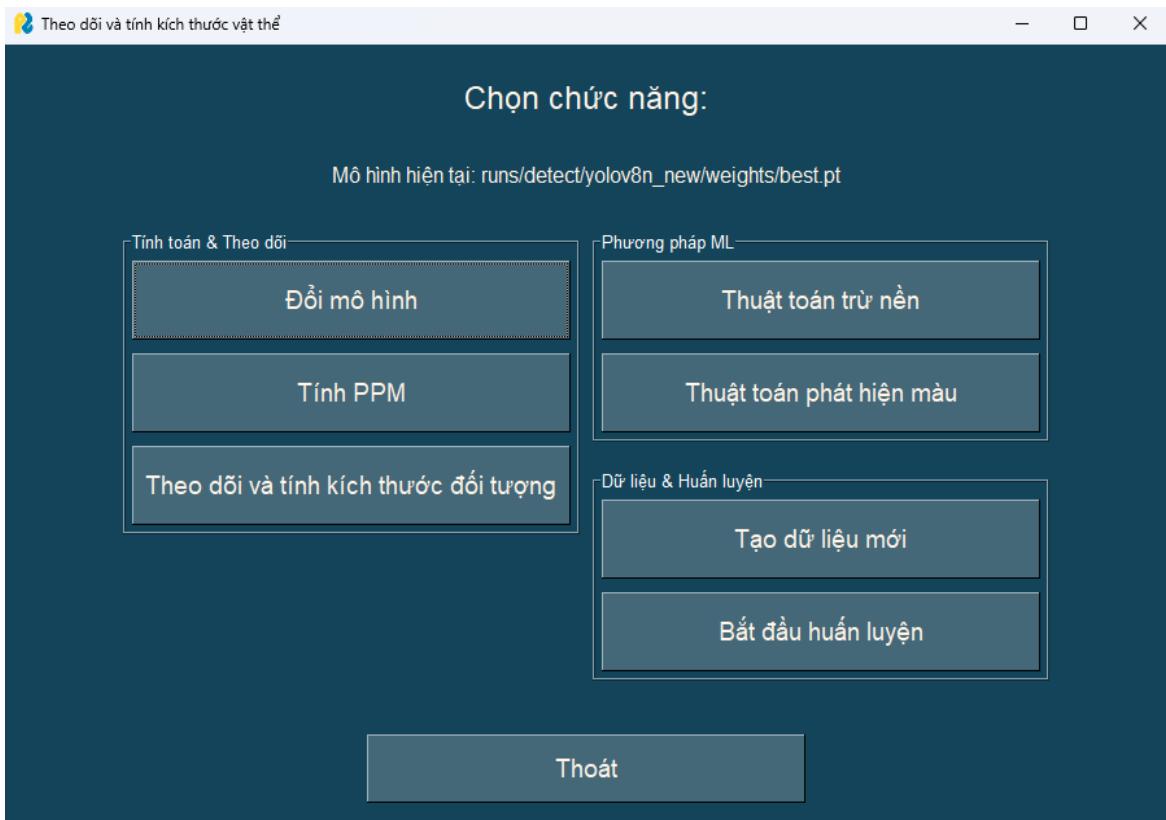
Hình 70. Giao diện chọn và đổi Model

Mô tả: Người dùng buộc phải chọn model để vào màn hình chính mới thực hiện được theo dõi và tính kích thước của đối tượng. Ở giao diện này người dùng có thể chọn bất kỳ mô hình nào đã được thêm sẵn hoặc có thể chọn mô hình YOLOv8 khác bằng cách chọn đường dẫn.



Hình 71. Màn hình báo lỗi khi người dùng không chọn model

Mô tả: Đây là màn hình báo lỗi khi người dùng không chọn model và ứng dụng sẽ thoát ra.

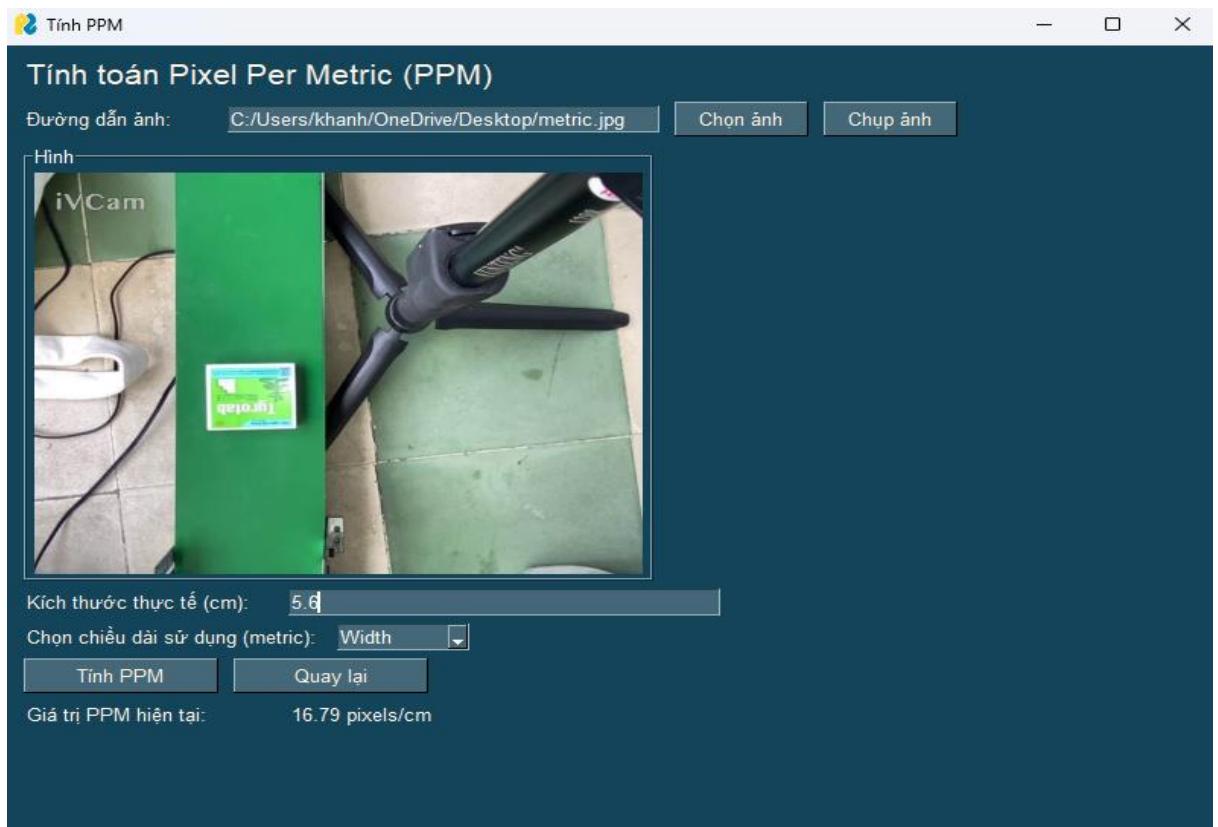


Hình 72. Màn hình chính của ứng dụng

Mô tả: Giao diện màn hình chính trong đó có các chức năng

- Tính toán & theo dõi
  - Đổi mô hình: Có thể đổi các mô hình YOLOv8 khác nhau để thực hiện việc theo dõi và tính kích thước các đối tượng khác.
  - Tính PPM: Đây sẽ là tính năng sẽ tính pixel per metric của object tham chiếu.
  - Theo dõi và tính kích thước đối tượng: Sẽ thực hiện việc theo dõi và tính kích thước bao gồm cả trên source video và trực tiếp từ webcam.
- Phương pháp ML
  - Thuật toán trừ nền: Đây là chức năng thể hiện demo của thuật toán trừ nền với tập dữ liệu 1

- Thuật toán phát hiện màu: Đây là chức năng thể hiện demo của thuật toán phát hiện màu với tập dữ liệu 1.
- Dữ liệu & huấn luyện
  - Tạo dữ liệu mới (đang phát triển): Đây là tính năng chụp hình của vật thể bao gồm từ source video hoặc trực tiếp từ webcam và chia nó theo từng thư mục theo class để hỗ trợ cho việc label và train model.
  - Bắt đầu huấn luyện đây là tính năng thực hiện huấn luyện mô hình trực tiếp trên ứng dụng.
- Thoát: Đây là tính năng sẽ thoát ứng dụng.



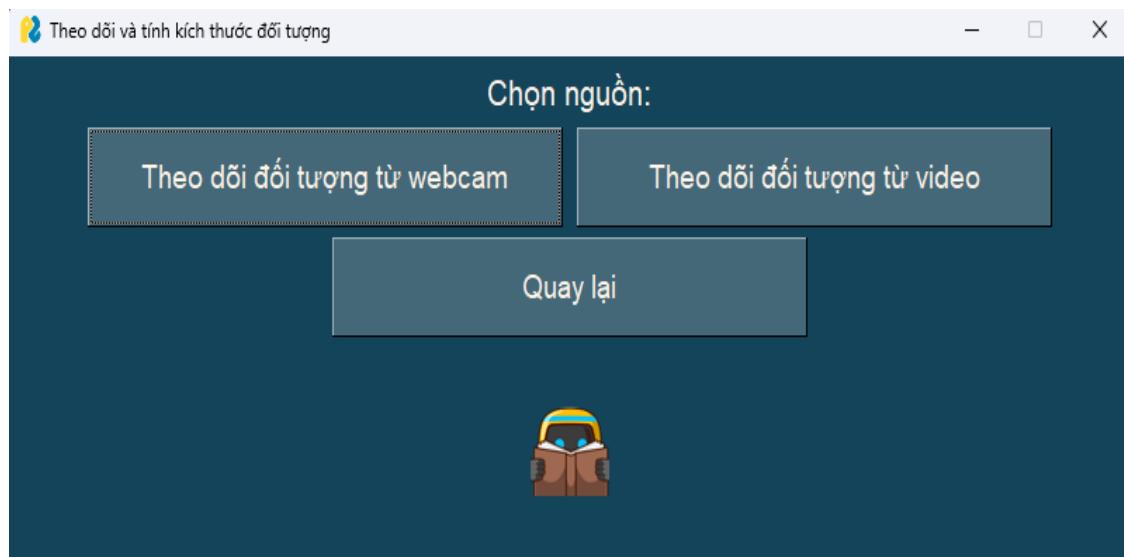
Hình 73. Chức năng tính toán PPM

Mô tả: Đây là chức năng giúp người dùng tính Pixel per metric người dùng chọn đường dẫn ảnh và cho biết kích thước thực tế. Sau đó chọn chiều dài thực tế tham chiếu sẽ tính ra được pixel per metric.

---

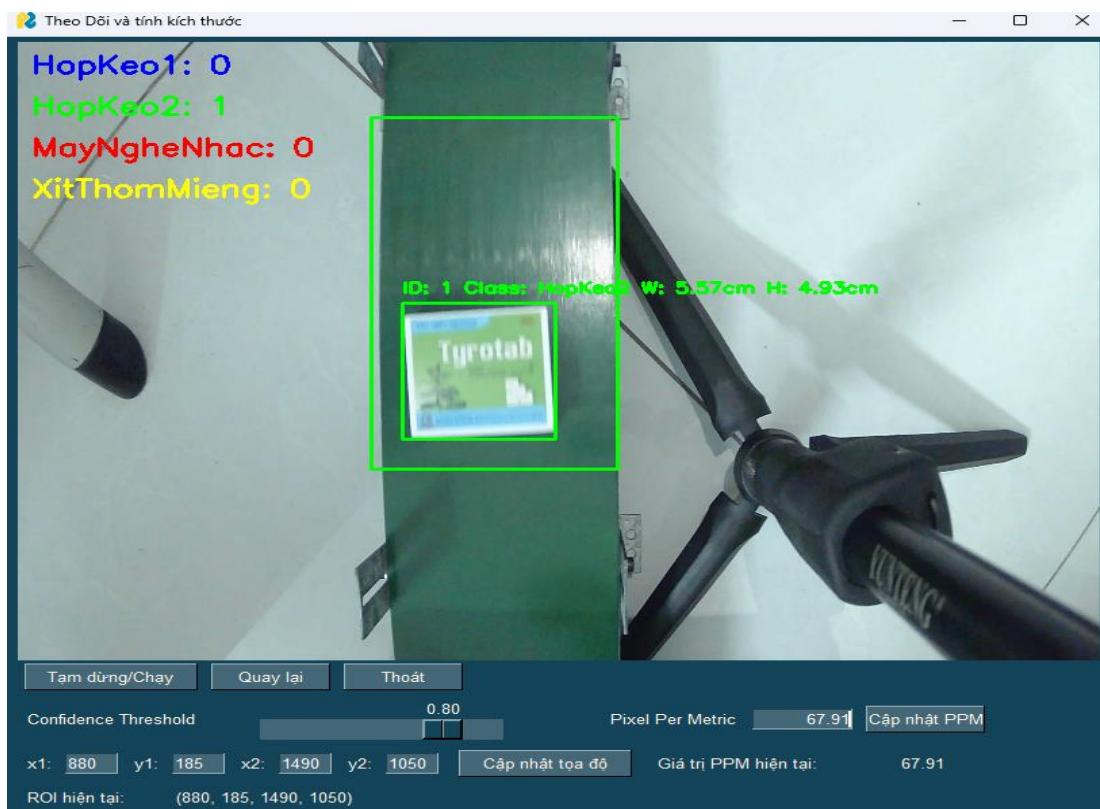
#### Chương 4. Giải pháp đề xuất

---



Hình 74. Chức năng theo dõi và tính kích thước đối tượng

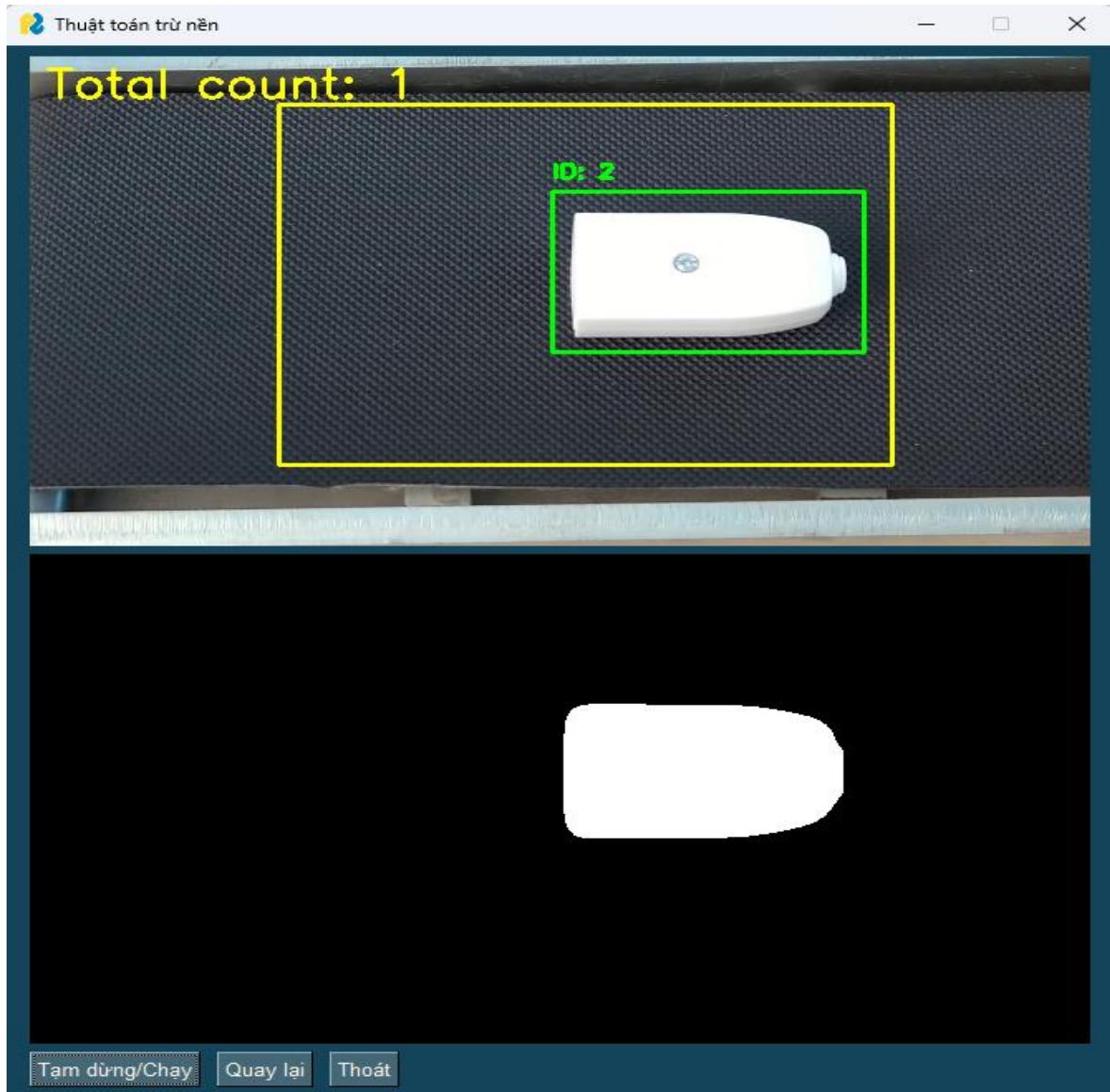
Mô tả: Có thể chọn được nguồn webcam hoặc video để thực hiện theo dõi và tính kích thước vật thể.



Hình 75. Màn hình khi theo dõi và tính kích thước đối tượng

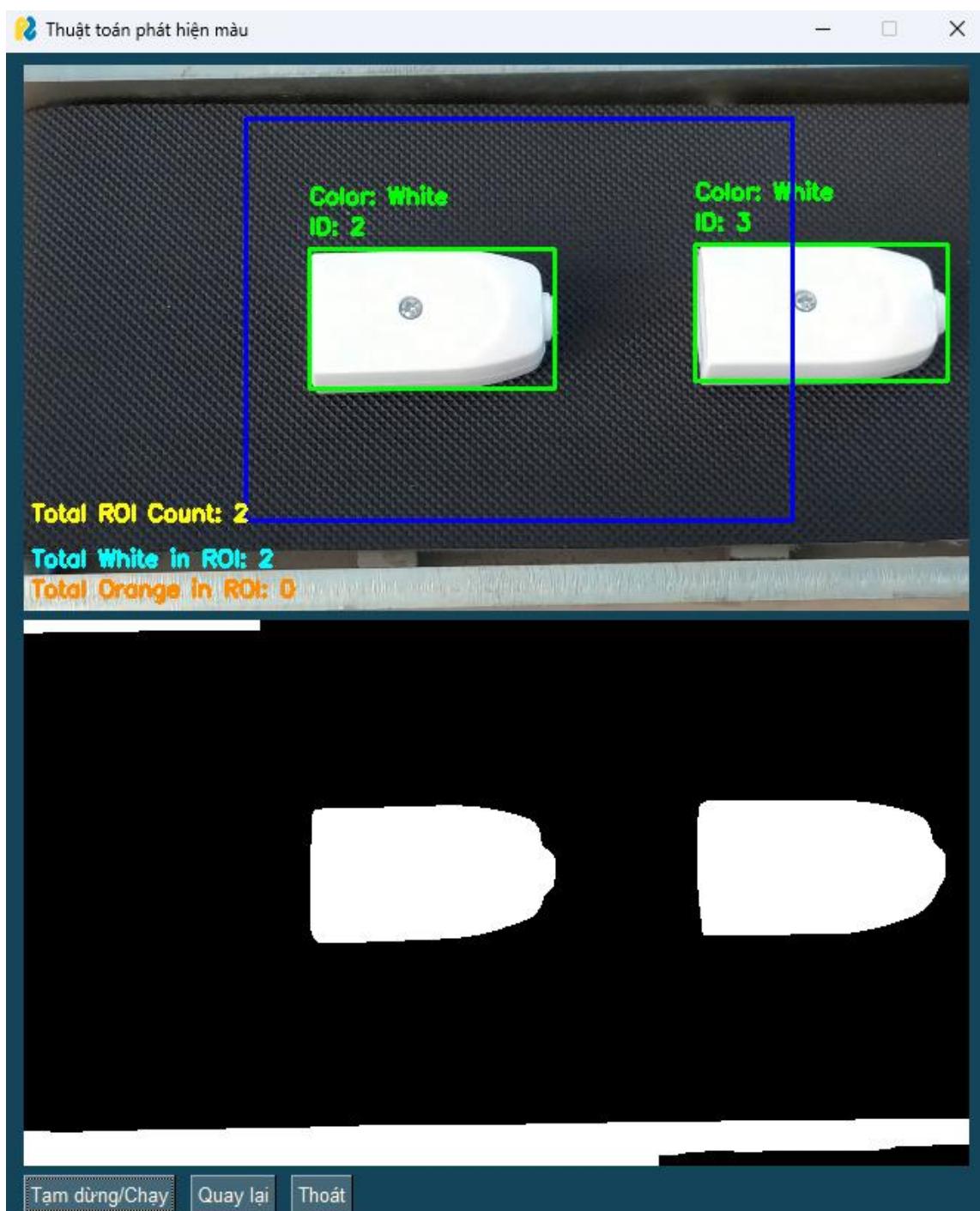
---

Mô tả: Sẽ thực hiện phát hiện đối tượng, theo dõi và tính kích thước. Hiển thị class của từng object được huấn luyện. Có thể điều chỉnh và cập nhật giá trị ROI, confidence và pixel per metric trực tiếp trên ứng dụng.



Hình 76. Giao diện thuật toán trừ nền

Mô tả: Đây là giao diện demo của thuật toán trừ nền với tập dữ liệu 1.

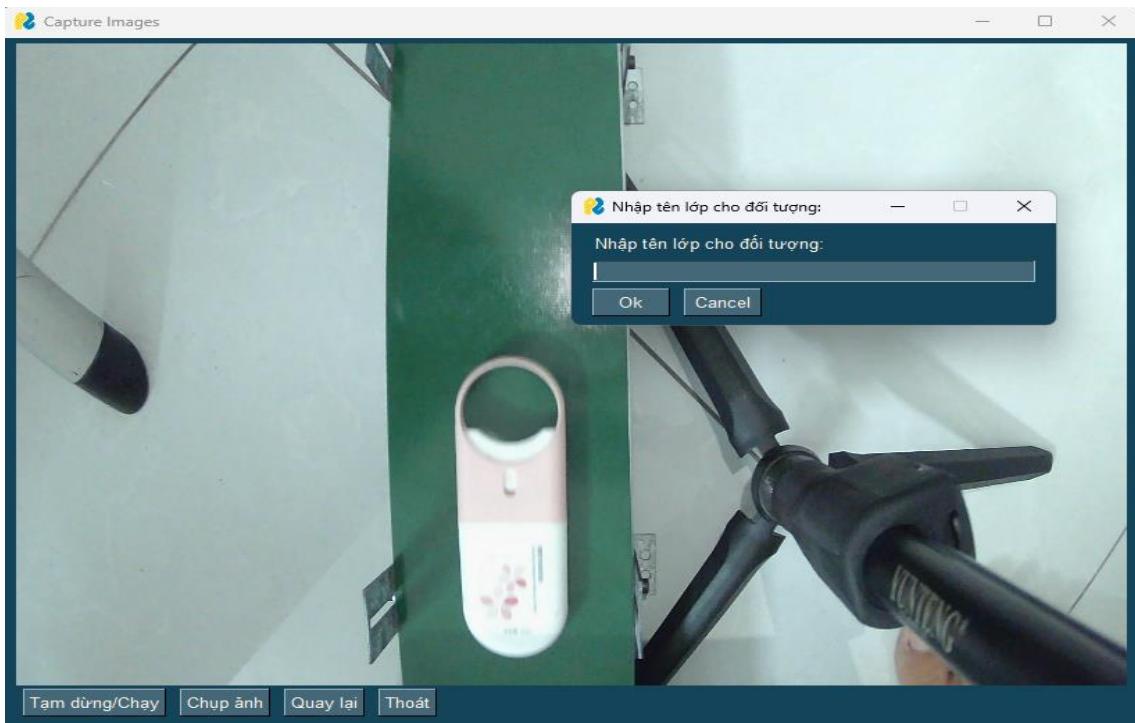


Hình 77. Giao diện thuật toán phát hiện màu

Mô tả: Đây là giao diện demo của thuật toán phát hiện màu với tập dữ liệu 1.

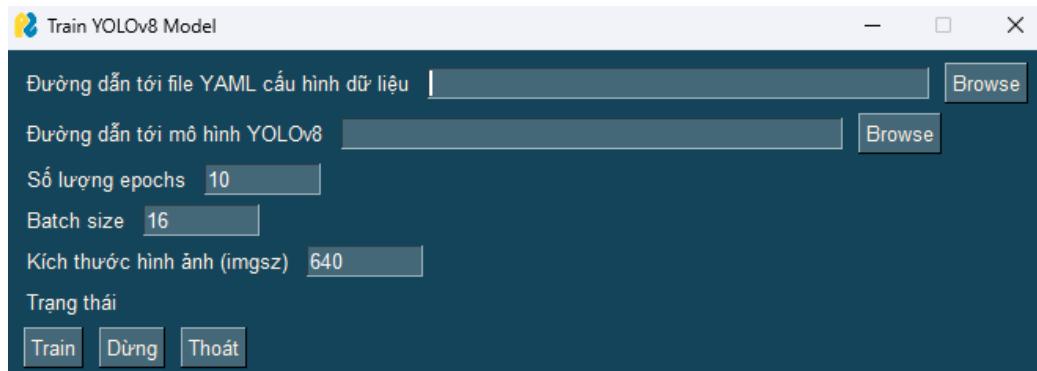
## Chương 4. Giải pháp đề xuất

---



Hình 78. Màn hình tạo dữ liệu mới

Mô tả: Đây là chức năng thêm mới dữ liệu phục vụ cho hướng phát triển mới trong tương lai đó là label. Có chức năng chụp và tạo thư mục cho dữ liệu theo từng class mà người dùng đặt tên.



Hình 79. Màn hình huấn luyện trực tiếp

Mô tả : Đây là chức năng huấn luyện mô hình trực tiếp trên ứng dụng yêu cầu người dùng nhập các tham số và đường dẫn model cần train và file YAML cấu hình của YOLO.

## KẾT LUẬN

### Những điều đã làm được

Đã tìm hiểu được các phương pháp về xác định đối tượng từ những kỹ thuật truyền thống đến hiện đại, theo dõi đối tượng bằng SORT và đếm đối tượng trong ROI. Hiểu được các phương pháp Machine Learning có điểm mạnh là tốc độ nhanh hơn Deep Learning và không yêu cầu phần cứng cao nhưng lại bị nhạy cảm với ánh sáng và độ nhiễu và không có khả năng phân loại tốt và gần như không thể phân loại. Còn những phương pháp Deep Learning thì tốc độ xử lý lâu hơn do yêu cầu về phần cứng cao và phức tạp hơn nhưng bù lại độ chính xác cao và ít bị ảnh hưởng bởi nhiễu và độ sáng hơn các phương pháp truyền thống và khả năng loại rất tốt. Nhóm đã xây dựng thành công ứng dụng demo về nhận diện, đếm sản phẩm và tính kích thước của sản phẩm trên băng chuyền.

### Những điều chưa làm được

Chưa thực hiện được với nhiều mô hình khác ngoài YOLO để xem hiệu suất và chưa giảm thiểu được khả năng phát hiện sai dù ít nhưng vẫn còn một ít sai sót do chất lượng camera và đầu vào đa dạng nhưng chưa đủ. Chưa thực hiện được tính năng label trực tiếp trên ứng dụng để tiện cho người dùng thêm sản phẩm và huấn luyện. Tốc độ xử lý còn chưa quá hoàn hảo do hạn chế về phần cứng.

### Hướng phát triển

Tìm hiểu thêm những mô hình khác xây dựng thêm các chức năng như cho phép người dùng label trực tiếp trên ứng dụng. Cùng với đó có thể phát triển thêm tính năng phát hiện lỗi nếu kích thước được đo chuẩn hơn áp dụng thực tế hơn cho doanh nghiệp. Có thể hướng tới DeepSort và các phương pháp truy vết và xác định đối tượng tốt và hiện đại hơn. Sẽ tăng cường phần cứng để có tốc độ xử lý nhanh và ổn định hơn, bao gồm cả việc sử dụng camera tốt và hiệu chỉnh tốt hơn để có kích thước chính xác triệt để. Xây dựng giao diện đẹp và thân thiện với người dùng.

---

## TÀI LIỆU THAM KHẢO

- [1] sourabh\_sinha, “Find and Draw Contours using OpenCV | Python,” 04/01/2023. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/>. [Đã truy cập 07/05/2024].
  - [2] V. Anh, “Lesson 05: Color Spaces,” 05/05/2023. [Trực tuyến]. Khả dụng: <https://www.vietanh.dev/courses/opencv/05-color-spaces>. [Đã truy cập 07/05/2024].
  - [3] goodday451999, “Multiple Color Detection in Real-Time using Python-OpenCV,” 22/03/2023. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/>. [Đã truy cập 08/05/2024].
  - [4] sourabh\_sinha, “Color Spaces in OpenCV | Python,” 04/01/2023. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/color-spaces-in-opencv-python/>. [Đã truy cập 01/05/2024].
  - [5] I. Berrios, “Introduction to Motion Detection: Part 3,” 03/11/2023. [Trực tuyến]. Khả dụng: <https://medium.com/@itberrios6/introduction-to-motion-detection-part-3-025271f66ef9>. [Đã truy cập 07/05/2024].
  - [6] ayushmankumar7, “Python OpenCV – Background Subtraction,” 03/01/2023. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/python-opencv-background-subtraction/>. [Đã truy cập 07/05/2024].
  - [7] L. M. Chiến, “Phát hiện đối tượng chuyển động bằng giải thuật trừ nền,” 04/05/2021. [Trực tuyến]. Khả dụng: <https://viblo.asia/p/phat-hien-doi-tuong-tru-nen>.
-

chuyen-dong-bang-giai-thuat-tru-nen-1VgZv6q1ZAw. [Đã truy cập 30/05/2024].

- [8] soumyadip, “Mastering All YOLO Models from YOLOv1 to YOLOv9: Papers Explained (2024),” 26/12/2023. [Trực tuyến]. Khả dụng: <https://learnopencv.com/mastering-all-yolo-models/>. [Đã truy cập 10/05/2024].
- [9] V. Hoàng, “Tìm hiểu về YOLO trong bài toán real-time object detection,” 16/11/2019. [Trực tuyến]. Khả dụng: <https://viblo.asia/p/tim-hieu-ve-yolo-trong-bai-toan-real-time-object-detection-yMnKMdvrl57P>. [Đã truy cập 10/05/2024].
- [10] R. T. Juan và M. C.-E. Diana , “A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS,” 04/02/2024. [Trực tuyến]. Khả dụng: <https://arxiv.org/abs/2304.00501>. [Đã truy cập 10/05/2024].
- [11] J. Solawetz và F. , “What is YOLOv8? The Ultimate Guide.(2024),” 11/01/2023. [Trực tuyến]. Khả dụng: <https://blog.roboflow.com/whats-new-in-yolov8/>. [Đã truy cập 10/05/2024].
- [12] et-al, “Cấu tạo và nguyên lý hoạt động của băng tải,” [Trực tuyến]. Khả dụng: <https://www.bangtaihang.com/tu-van-ky-thuat/cau-tao-va-nguyen-ly-hoat-dong-cua-bang-tai/>. [Đã truy cập 01/05/2024].
- [13] kumar\_satyam, «Computer Vision Tutorial,» 29/04/2024. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/computer-vision/>. [Đã truy cập 30/05/2024].

- [14] H. Vedoveli, “Image Filtering Techniques in Image Processing — Part 1,” 26/08/2023.[Trực tuyến]. Khả dụng: <https://medium.com/@henriquevedoveli/image-filtering-techniques-in-image-processing-part-1-d03362fc73b7>. [Đã truy cập 05/06/2024].
- [15] TAPIT, «Giới thiệu về Xử lý ảnh hình thái học – Morphological Image Processing,» 10/11/2021. [Trực tuyến]. Khả dụng: <https://tapit.vn/xu-ly-anh-hinh-thai-hoc-morphological-image-processing/>. [Đã truy cập 10/06/2024].
- [16] GeeksforGeeks, “Deep Learning Tutorial,” 09/05/2023. [Trực tuyến]. Khả dụng: <https://www.geeksforgeeks.org/deep-learning-tutorial/>. [Đã truy cập 05/06/2024].
- [17] P. V. Toàn, “[Handbook CV with DL - Phần 1] Các khái niệm cơ bản trong Computer Vision và Deep Learning,” 20/02/2019. [Trực tuyến]. Khả dụng: [https://viblo.asia/p/handbook-cv-with-dl-phan-1cac-khai-niem-co-ban-trong-computer-vision-va-deep-learning-maGK7p2MZj2#\\_perceptron-1](https://viblo.asia/p/handbook-cv-with-dl-phan-1cac-khai-niem-co-ban-trong-computer-vision-va-deep-learning-maGK7p2MZj2#_perceptron-1). [Đã truy cập 03/05/2024].
- [18] e. a. Đoàn Võ Duy Thành, “Đắm mình vào học sâu,” 18/04/2022. [Trực tuyến]. Khả dụng: [https://d2l.aivivn.com/chapter\\_multilayer-perceptrons/mlp\\_vn.html](https://d2l.aivivn.com/chapter_multilayer-perceptrons/mlp_vn.html). [Đã truy cập 05/05/2024].
- [19] T. V. Huu, “Machine Learning Cơ Bản,” 24/02/2017. [Trực tuyến]. Khả dụng: <https://machinelearningcoban.com/2017/02/24/mlp/>. [Đã truy cập 05/05/2024].
- [20] C. P. Van, “[Deep Learning] Tìm hiểu về mạng tích chập (CNN),” 12/10/2020. [Trực tuyến]. Khả dụng: <https://viblo.asia/p/deep-learning-tim-hieu-ve-mang-tich-chap-cnn>.

- tich-chap-cnn-maGK73bOKj2#\_2-lop-tich-chap---convolution-layer-1. [Đã truy cập 05/05/2024].
- [21] Z. Keita, “An Introduction to Convolutional Neural Networks (CNNs),” 11/11/2023. [Trực tuyến]. Khả dụng: <https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns>. [Đã truy cập 05/05/2024].
- [22] N. N. Hung, “Tìm hiểu về Convolutional Neural Networks (CNN),” 14/08/2021. [Trực tuyến]. Khả dụng: <https://viblo.asia/p/tim-hieu-ve-convolutional-neural-networks-cnn-naQZRkr0lvx>. [Đã truy cập 05/05/2024].
- [23] OpenCV, “Contours in OpenCV,” [Trực tuyến]. Khả dụng: [https://docs.opencv.org/4.x/d3/d05/tutorial\\_py\\_table\\_of\\_contents\\_contours.html](https://docs.opencv.org/4.x/d3/d05/tutorial_py_table_of_contents_contours.html). [Đã truy cập 07/05/2024].
- [24] B. Esme, “Kalman Filter For Dummies,” 03/2009. [Trực tuyến]. Khả dụng: <http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies>. [Đã truy cập 20/05/2024].
- [25] T. T. Bui, “SORT - Deep SORT : Một góc nhìn về Object Tracking (phần 1),” 16/12/2020. [Trực tuyến]. Khả dụng: [https://viblo.asia/p/sort-deep-sort-mot-goc-nhin-ve-object-tracking-phan-1-Az45bPooZxY#\\_32-bo-loc-kalman-kalman-filter-9](https://viblo.asia/p/sort-deep-sort-mot-goc-nhin-ve-object-tracking-phan-1-Az45bPooZxY#_32-bo-loc-kalman-kalman-filter-9). [Đã truy cập 20/05/2024].
- [26] T. Babb, “How a Kalman filter works, in pictures,” 11/08/2015. [Trực tuyến]. Khả dụng: <https://www.bzarg.com/p/how-a-kalman-filter-works-in-pictures/>. [Đã truy cập 20/05/2024].

- [27] A. Minisini, “Hungarian algorithm for solving the assignment problem,” 13/12/2023. [Trực tuyến]. Khả dụng: <https://cp-algorithms.com/graph/hungarian-algorithm.html>. [Đã truy cập 25/05/2024].
- [28] Brilliant, “Hungarian Maximum Matching Algorithm,” 03/07/2018. [Trực tuyến]. Khả dụng: <https://brilliant.org/wiki/hungarian-matching/>. [Đã truy cập 26/05/2024].
- [29] A. Bewley, “Simple Online and Realtime Tracking,” 07/07/2017. [Trực tuyến]. Khả dụng: <https://arxiv.org/abs/1602.00763>. [Đã truy cập 02/06/2024].
- [30] R. Janghu, “Region of Interest in Computer Vision,” 21/12/2023. [Trực tuyến]. Khả dụng: <https://www.scaler.com/topics/region-of-interest-opencv/>. [Đã truy cập 15/06/2024].
- [31] A. Rosebrock, “Measuring size of objects in an image with OpenCV,” 28/03/2016. [Trực tuyến]. Khả dụng: <https://pyimagesearch.com/2016/03/28/measuring-size-of-objects-in-an-image-with-opencv/>. [Đã truy cập 16/07/2024].

## PHỤ LỤC

### 1. Code của các thành phần trong SORT

```

class KalmanBoxTracker(object):
    count = 0
    def __init__(self,bbox):
        self.kf = KalmanFilter(dim_x=7, dim_z=4)
        self.kf.F =
        np.array([[1,0,0,0,1,0,0],[0,1,0,0,0,1,0],[0,0,1,0,0,0,1],[0,0,0
        ,1,0,0,0],[0,0,0,0,1,0,0],[0,0,0,0,0,1,0],[0,0,0,0,0,0,1]])
        self.kf.H =
        np.array([[1,0,0,0,0,0,0],[0,1,0,0,0,0,0],[0,0,1,0,0,0,0],[0,0,0
        ,1,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0],[0,0,0,0,0,0,0]])
        self.kf.R[2:,2:] *= 10.
        self.kf.P[4:,4:] *= 1000.
        self.kf.P *= 10.
        self.kf.Q[-1,-1] *= 0.01
        self.kf.Q[4:,4:] *= 0.01

        self.kf.x[:4] = convert_bbox_to_z(bbox)
        self.time_since_update = 0
        self.id = KalmanBoxTracker.count
        KalmanBoxTracker.count += 1
        self.history = []
        self.hits = 0
        self.hit_streak = 0
        self.age = 0
    def update(self,bbox):
        self.time_since_update = 0
        self.history = []
        self.hits += 1
        self.hit_streak += 1
        self.kf.update(convert_bbox_to_z(bbox))
    def predict(self):
        if((self.kf.x[6]+self.kf.x[2])<=0):
            self.kf.x[6] *= 0.0
        self.kf.predict()
        self.age += 1
        if(self.time_since_update>0):
            self.hit_streak = 0
        self.time_since_update += 1
        self.history.append(convert_x_to_bbox(self.kf.x))
        return self.history[-1]
    def get_state(self):
        return convert_x_to_bbox(self.kf.x)

```

Lớp KalmanBoxTracker:

- `__init__`: Khởi tạo Kalman Filter với các thông số ban đầu.
- `update()`: Cập nhật trạng thái Kalman Filter với bounding box mới.
- `predict()`: Dự đoán trạng thái kế tiếp của đối tượng.
- `get_state()`: Lấy trạng thái hiện tại của đối tượng.

Hungarian Algorithm: Còn được gọi là Linear Assignment Problem) được sử dụng để gán các detection mới với các tracker hiện có dựa trên chỉ số IoU.

Hàm `linear_assignment`:

```
def linear_assignment(cost_matrix):
    try:
        import lap
        _, x, y = lap.lapjv(cost_matrix, extend_cost=True)
        return np.array([[y[i], i] for i in x if i >= 0]) #
    except ImportError:
        from scipy.optimize import linear_sum_assignment
        x, y = linear_sum_assignment(cost_matrix)
        return np.array(list(zip(x, y)))
```

Hàm `linear_assignment` giải bài toán gán tuyén tính bằng hai phương pháp:

- Thư viện `lap` với thuật toán LAPJV.
- Thư viện `scipy` với hàm `linear_sum_assignment`

```

def associate_detections_to_trackers(detections,trackers,iou_threshold = 0.3):
    if(len(trackers)==0):
        return np.empty((0,2),dtype=int), np.arange(len(detections)),
        np.empty((0,5),dtype=int)
    iou_matrix = iou_batch(detections, trackers)
    if min(iou_matrix.shape) > 0:
        a = (iou_matrix > iou_threshold).astype(np.int32)
        if a.sum(1).max() == 1 and a.sum(0).max() == 1:
            matched_indices = np.stack(np.where(a), axis=1)
        else:
            matched_indices = linear_assignment(-iou_matrix)
    else:
        matched_indices = np.empty(shape=(0,2))
    unmatched_detections = []
    for d, det in enumerate(detections):
        if(d not in matched_indices[:,0]):
            unmatched_detections.append(d)
    unmatched_trackers = []
    for t, trk in enumerate(trackers):
        if(t not in matched_indices[:,1]):
            unmatched_trackers.append(t)

    #filter out matched with low IOU
    matches = []
    for m in matched_indices:
        if(iou_matrix[m[0], m[1]]<iou_threshold):
            unmatched_detections.append(m[0])
            unmatched_trackers.append(m[1])
        else:
            matches.append(m.reshape(1,2))
    if(len(matches)==0):
        matches = np.empty((0,2),dtype=int)
    else:
        matches = np.concatenate(matches,axis=0)

    return matches, np.array(unmatched_detections),
    np.array(unmatched_trackers)

```

Hàm associate\_detections\_to\_trackers:

- iou\_matrix: Ma trận IoU giữa các detections và trackers.
- matched\_indices: Chỉ số của các cặp matched detection-tracker.
- unmatched\_detections: Các detections không được gán.
- unmatched\_trackers: Các trackers không được gán.

Quản lý danh tính đối tượng: Phần này quản lý các tracker và gán danh tính duy nhất cho từng đối tượng.

---

Lớp Sort:

```

class Sort(object):
    def __init__(self, max_age=1, min_hits=3, iou_threshold=0.3):
        self.max_age = max_age
        self.min_hits = min_hits
        self.iou_threshold = iou_threshold
        self.trackers = []
        self.frame_count = 0

    def update(self, dets=np.empty((0, 5))):
        self.frame_count += 1
        trks = np.zeros((len(self.trackers), 5))
        to_del = []
        ret = []
        for t, trk in enumerate(trks):
            pos = self.trackers[t].predict()[0]
            trk[:] = [pos[0], pos[1], pos[2], pos[3], 0]
            if np.any(np.isnan(pos)):
                to_del.append(t)
        trks = np.ma.compress_rows(np.ma.masked_invalid(trks))
        for t in reversed(to_del):
            self.trackers.pop(t)
        matched, unmatched_dets, unmatched_trks =
        associate_detections_to_trackers(dets, trks, self.iou_threshold)
        for m in matched:
            self.trackers[m[1]].update(dets[m[0], :])

        for i in unmatched_dets:
            trk = KalmanBoxTracker(dets[i,:])
            self.trackers.append(trk)
        i = len(self.trackers)
        for trk in reversed(self.trackers):
            d = trk.get_state()[0]
            if (trk.time_since_update < 1) and (trk.hit_streak >=
self.min_hits or self.frame_count <= self.min_hits):
                ret.append(np.concatenate((d,[trk.id+1])).reshape(1,-1))
            i -= 1
            # remove dead tracklet
            if(trk.time_since_update > self.max_age):
                self.trackers.pop(i)
        if(len(ret)>0):
            return np.concatenate(ret)
        return np.empty((0,5))

```

- `max_age`: Số khung hình tối đa để giữ một tracker mà không có detection mới.
- `min_hits`: Số lần hit tối thiểu để khởi tạo một tracker.
- `iou_threshold`: Nguồn IoU tối thiểu để gán detection và tracker.
- `update(dets)`: Cập nhật trackers với các detection mới.

Hàm hỗ trợ chuyển đổi boundingbox:

```
def convert_bbox_to_z(bbox):
    w = bbox[2] - bbox[0]
    h = bbox[3] - bbox[1]
    x = bbox[0] + w/2.
    y = bbox[1] + h/2.
    s = w * h      #scale is just area
    r = w / float(h)
    return np.array([x, y, s, r]).reshape((4, 1))
```

```
def convert_x_to_bbox(x, score=None):
    w = np.sqrt(x[2] * x[3])
    h = x[2] / w
    if(score==None):
        return np.array([x[0]-w/2., x[1]-
h/2., x[0]+w/2., x[1]+h/2.]).reshape((1,4))
    else:
        return np.array([x[0]-w/2., x[1]-
h/2., x[0]+w/2., x[1]+h/2., score]).reshape((1,5))
```

Hàm hỗ trợ tính toán IoU:

```
def iou_batch(bb_test, bb_gt):
    """
    From SORT: Computes IOU between two bboxes in the form
    [x1,y1,x2,y2]
    """
    bb_gt = np.expand_dims(bb_gt, 0)
    bb_test = np.expand_dims(bb_test, 1)

    xx1 = np.maximum(bb_test[:, 0], bb_gt[:, 0])
    yy1 = np.maximum(bb_test[:, 1], bb_gt[:, 1])
    xx2 = np.minimum(bb_test[:, 2], bb_gt[:, 2])
    yy2 = np.minimum(bb_test[:, 3], bb_gt[:, 3])
    w = np.maximum(0., xx2 - xx1)
    h = np.maximum(0., yy2 - yy1)
    wh = w * h
    o = wh / ((bb_test[:, 2] - bb_test[:, 0]) * (bb_test[:, 3] -
bb_test[:, 1]) +
            (bb_gt[:, 2] - bb_gt[:, 0]) * (bb_gt[:, 3] -
bb_gt[:, 1]) -
            wh)
    return(o)
```

## 2. Ví dụ về Kalman Filter

Để có thể dễ hình dung hơn về Kalman Filter ta sẽ bắt đầu với một ví dụ đơn giản mà nhóm tham khảo được trên internet ở tài liệu tham khảo số [26].

Chúng ta đã chế tạo một robot nhỏ có thể đi trong rừng và robot thì cần biết vị trí chính xác của nó để nó có thể định hướng.



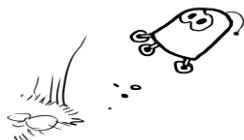
Hình 80. Ví dụ đơn giản về Kalman Filter

Chúng ta sẽ nói robot của chúng ta có trạng thái  $\vec{x}_k$  đó chỉ là vị trí và vận tốc

$$\vec{x}_k = (\vec{p}, \vec{v})$$

Có một lưu ý đây chỉ là ví dụ trạng thái chỉ là một danh sách các con số về cấu hình cơ bản của hệ thống của chúng ta. Nó có thể là bất cứ thứ gì trong ví dụ trên nó là vị trí và vận tốc. Nhưng nó cũng có thể là dữ liệu về lượng chất lỏng trong bình, nhiệt độ động cơ của ô tô, vị trí của một chiếc xe hoặc bất cứ thông tin nào mà ta cần theo dõi.

Tiếp tục với ví dụ thì robot của chúng ta cũng sẽ có cảm biến GPS chính xác đến khoảng 10m. Điều này là rất tốt nhưng nó cũng cần biết vị trí chính xác hơn 10m. Có rất nhiều rãnh và vách đá trong khu rừng này và nếu robot đi sai quá vài feet. Nó có thể rơi khỏi vách đá vì vậy bản thân chỉ GPS thôi thì chưa đủ tốt.



Hình 81. Ví dụ đơn giản về Kalman Filter

Chúng ta cũng có thể biết một số thông tin về cách robot di chuyển. Nó biết các lệnh được gửi đến động cơ bánh xe và nó biết rằng nếu nó đi về một hướng và không có gì cản trở thì ở thời điểm tiếp theo nó sẽ có khả năng tiến xa hơn theo hướng đó. Nhưng tất nhiên nó không biết mọi thứ về chuyển động của mình: Nó có thể bị gió thổi, bánh xe có thể bị trượt một chút hoặc lăn qua địa hình gồ ghề do đó số vòng của bánh xe có thể không thể hiện chính xác quãng đường robot đã đi được và dự đoán sẽ không hoàn hảo. Cảm biến GPS có thể cho ta biết điều gì đó về trạng thái nhưng nó chỉ là một cách gián tiếp và sẽ có một số điểm không chắc chắn hoặc sẽ không chính xác.

Dự đoán của chúng ta cũng chỉ cho chúng ta biết một số thông tin về cách robot di chuyển nhưng cũng chỉ là gián tiếp và cũng sẽ có một số điểm không chắc chắn và không chính xác.

Nhưng nếu chúng ta sử dụng tất cả thông tin có sẵn thì liệu chúng ta có thể nhận được câu trả lời tốt hơn so với mỗi ước lượng riêng lẻ có thể cho chúng ta không? Câu trả lời tất nhiên là có và đó chính xác là mục đích của Kalman Filter.

Ví dụ thực hiện thuật toán Kalman Filter:

Giả sử chúng ta muốn ước lượng một hằng số ngẫu nhiên chẳng hạn như một “giá trị đo điện áp” từ một nguồn. Giả sử rằng giá trị điện áp này là một hằng số  $a$  volt nhưng tất nhiên chúng ta có một số giá trị đo nhiều xung quanh  $a$  volt. Độ lệch chuẩn của nhiều đo lường là 0.1V.

Bắt đầu với việc xây dựng mô hình được nhóm lấy ví dụ ở tài liệu thao khảo [19].

$$\begin{aligned}x_k &= Ax_{k-1} + Bu_k + w_{k-1} \\&= x_{k-1} + w_k \quad [19]\end{aligned}$$

$$\begin{aligned} z_k &= Hx_k + v_k \\ &= x_k + v_k \quad [19] \end{aligned}$$

- Chúng ta gặp vấn đề về tín hiệu 1 chiều. Vì vậy mọi thực thể trong mô hình của chúng ta đều là một giá trị số, không phải là ma trận.
- Chúng ta không có tín hiệu điều khiển nên sẽ không có  $u_k$
- Các giá trị ma trận
- $A$  và  $H$  đều bằng 1, vì tín hiệu đo lường chỉ là giá trị thực của trạng thái cộng với nhiễu đo lường.

Thời gian (ms)	Giá trị (V)
1	0.39
2	0.50
3	0.48
4	0.29
5	0.25
6	0.32
7	0.34
8	0.48
9	0.41
10	0.35

Bảng 5. Ví dụ bảng giá trị đo của Kalman Filter

Phía trên là bảng các giá trị đo của chúng ta. Vậy thì để tiếp tục chúng ta nên bắt đầu từ một giá trị nào đó chẳng hạn như  $k = 0$ . Chúng ta nên tìm hoặc giả định một trạng thái ban đầu. Ở đây ta sẽ đưa ra một số giá trị ban đầu giả sử như  $x_0 = 0$  và  $P_0 = 1$ . Vậy thì tại sao chúng cho không chọn  $P_0 = 0$ ? Vì nếu ta chọn như vậy thì đồng nghĩa với việc sẽ không có nhiễu trong môi trường và giả định này sẽ dẫn đến tất cả các ước lượng của  $X$  ở các trạng thái  $k$  tiếp theo đều bằng không (giữ nguyên như trạng thái ban đầu). Vì vậy, chúng ta chọn  $P_0$  là một giá trị khác 0. Tiếp theo ta sẽ viết phương trình cập nhật thời gian và cập nhật đo lường.

Phương trình cập nhật thời gian (dự đoán):

$$\begin{aligned} \hat{x}_k^- &= \hat{x}_{k-1} \\ P_k^- &= P_{k-1} \quad [19] \end{aligned}$$

Phương trình cập nhật đo lường (chỉnh sửa):

$$\begin{aligned} K_k &= \frac{P_k^-}{P_k^- + R} \\ \hat{x}_k &= \hat{x}_k^- + K_k(z_k - \hat{x}_k^-) \quad [19] \\ P_k &= (1 - K_k)P_k^- \end{aligned}$$

Bắt đầu thực hiện phép tính  $\hat{x}_k$  cho mỗi lần lặp

$k$	$z_k$	$\hat{x}_{k-1}$	$P_k^-$	Update Time	Update Measurement	$\hat{x}_k$	$P_k$
1	0.390	0	1	$\hat{x}_k^- = 0$ $P_k^- = 1$	$K_k = \frac{1}{1 + 0.1}$ = 0.909  $\hat{x}_k = 0.909 \times (0.390 - 0) = 0.35$  $P_k = (1 - 0.909) \times 1$ = 0.091	0.355	0.091
2	0.500	0.355	0.091	$\hat{x}_k^- = 0.355$ $P_k^- = 0.091$	$K_k = \frac{0.091}{0.091 + 0.1}$ = 0.476  $\hat{x}_k = 0.355 \times 0.476$ $x (0.500 - 0.355)$ = 0.424  $P_k = (1 - 0.476) \times 0.091 = 0.048$	0.424	0.048
...	...	...	...	...	...	...	...
10	0.450	0.380	0.011	...	...	0.387	0.010

Bảng 6. Ví dụ thực hiện Kalman Filter

Các bước tiếp theo tương tự ở đây ta chỉ thực hiện 2 bước vì đây chỉ là ví dụ cho biết cách thức hoạt động. Mỗi bước  $k$  sẽ sử dụng các giá trị ước lượng và hiệp phương sai từ bước trước đó để tính toán. Kết quả, sau 10 bước, các giá trị ước lượng  $\hat{x}_k$  dần hội tụ về giá trị thực của điện áp, giảm dần độ nhiễu và cải thiện độ chính xác của ước lượng

### 3. Ví dụ về thuật toán Hungarian

Ví dụ về thuật toán Hungarian: Đây là ví dụ mà nhóm tham khảo được ở tài liệu tham khảo số [28].

Công ty	Âm nhạc	Nấu ăn	Dọn dẹp
A	108	125	150
B	150	135	175
C	122	148	250

Bảng 7. Ví dụ thuật toán Hungarian

Bước 1: Tạo ma trận chi phí

$$\begin{matrix} 108 & 125 & 150 \\ 150 & 135 & 175 \\ 122 & 148 & 250 \end{matrix}$$

Bước 2: Trừ giá trị nhỏ nhất mỗi hàng

$$\begin{matrix} 0 & 17 & 42 \\ 15 & 0 & 40 \\ 0 & 26 & 128 \end{matrix}$$

Bước 3: Trừ giá trị nhỏ nhất trong mỗi cột

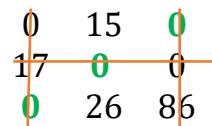
$$\begin{matrix} 0 & 17 & 2 \\ 15 & 0 & 0 \\ 0 & 26 & 88 \end{matrix}$$

Bước 4: Vẽ các đường qua các số 0 sao cho số đường là ít nhất

$$\begin{matrix} 0 & 17 & 2 \\ 15 & 0 & 0 \\ 0 & 26 & 88 \end{matrix}$$

### Bước 5: Điều chỉnh ma trận

Chúng ta cần 3 đường để che phủ tất cả các số 0. Ở đây ta thấy được sau các đường đã gạch thì ta thu được giá trị 2 là nhỏ nhất tiếp tục trừ các số còn lại với số 2 và cộng 2 với số nằm giữa giao điểm 2 đường thẳng ở đây đó là số 15.



Lặp lại các bước trên cho đến khi có đủ số đường che phủ bằng số hàng (hoặc cột). Cuối cùng, chúng ta sẽ có ma trận tối ưu và gán công việc theo các số 0 đã chọn. Sau đó ta thực hiện thay thế các giá trị ban đầu để thấy kết quả tối ưu nhất

$$\begin{array}{ccc} 108 & 125 & \textcolor{red}{150} \\ 150 & \textcolor{red}{135} & 175 \\ \textcolor{red}{122} & 148 & 250 \end{array}$$

Thuật toán Hungarian cho ta biết rằng khi đi cùng nhạc sĩ của công ty C, đầu bếp của công ty B và người dọn dẹp của công ty A là rẻ nhất.