

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

KHOA CÔNG NGHỆ THÔNG TIN

KHÓA LUẬN TỐT NGHIỆP

**ĐẾM SỐ ĐỐI TƯỢNG TỪ MÁY BAY
KHÔNG NGƯỜI LÁI**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Phạm Đình Thắng

SINH VIÊN THỰC HIỆN:

Nguyễn Ngọc Anh Thiên – 21DH114521

Đỗ Tín Thành – 21DH114107

TP. HỒ CHÍ MINH – 08/2024

TRƯỜNG ĐẠI HỌC NGOẠI NGỮ - TIN HỌC TP. HỒ CHÍ MINH
KHOA CÔNG NGHỆ THÔNG TIN

KHÓA LUẬN TỐT NGHIỆP

**ĐẾM SỐ ĐỐI TƯỢNG TỪ MÁY BAY
KHÔNG NGƯỜI LÁI**

GIẢNG VIÊN HƯỚNG DẪN: ThS. Phạm Đình Thắng

SINH VIÊN THỰC HIỆN:

Nguyễn Ngọc Anh Thiên – 21DH114521

Đỗ Tín Thành – 21DH114107

TP. HỒ CHÍ MINH – 08/2024

LỜI CẢM ƠN

Lời cảm ơn đầu tiên, chúng em xin phép gửi đến thầy hướng dẫn ThS. Phạm Đình Thắng. Chúng em rất biết ơn thầy vì đã tiếp nhận và nhiệt tình chỉ dạy, tạo điều kiện thuận lợi cho chúng em nghiên cứu khóa luận thông qua các tài liệu và kiến thức quý báu mà chúng em đã có thể nắm bắt và ghi nhận vào bài báo cáo này.

Tiếp theo, chúng em xin gửi lời cảm ơn chân thành đến Ban Giám hiệu nhà trường Đại học Ngoại ngữ - Tin học Thành phố Hồ Chí Minh, cùng quý thầy cô tại khoa Công nghệ thông tin đã tạo cơ hội cho chúng em thực hiện đề tài khóa luận được giao và nỗ lực giảng dạy, truyền đạt những kiến thức quý báu cho chúng em trong suốt những năm qua.

Khoảng thời gian thực hiện khóa luận đã giúp chúng em tổng hợp và hệ thống hóa kiến thức đã học, cũng như tìm tòi và mở rộng tri thức với những kiến thức mới. Mặc dù thời gian thực hiện không nhiều, nhưng quá trình làm khóa luận đã giúp chúng em học hỏi thêm nhiều điều, mở rộng khả năng nghiên cứu và tích lũy được thêm nhiều kiến thức bổ ích.

Do thời gian và kiến thức của chúng em còn hạn chế, nên bài báo cáo cũng như kết quả khóa luận không tránh khỏi những sai sót. Chúng em rất mong nhận được sự đánh giá và góp ý từ thầy cô và các bạn để chúng em có thể rút kinh nghiệm và hoàn thiện tốt hơn trong tương lai.

LỜI CAM ĐOAN

Chúng em xin cam đoan bài khóa luận "Đếm số đối tượng từ máy bay không người lái" là công trình nghiên cứu đã được chúng em thực hiện cũng như hoàn thành toàn bộ dưới sự hướng dẫn khoa học của ThS. Phạm Đình Thắng – Trường Đại học Ngoại Ngữ - Tin Học Thành phố Hồ Chí Minh. Những kết quả từ khóa luận này là hoàn toàn chưa từng được công bố trong những công trình nghiên cứu riêng biệt nào khác. Việc sử dụng các thuật toán và những trích dẫn từ tài liệu của những tác giả khác đã được chúng em đảm bảo thực hiện theo đúng các quy định khi làm khóa luận. Các thuật toán được sử dụng cũng như các tài liệu từ sách báo và thông tin tham khảo đã được đăng tải trên các tác phẩm cũng như các trang web được trình bày theo danh mục tài liệu tham khảo của khóa luận.

TÓM TẮT KHÓA LUẬN

Với sự tiến bộ của công nghệ máy bay không người lái (UAV), việc thu thập dữ liệu từ nhiều góc nhìn đã trở nên dễ dàng và hiệu quả hơn. Đề tài này nhằm phát triển một giải pháp tự động đếm số lượng một loại đối tượng cụ thể trong video được quay từ UAV. Điều này có thể hữu ích trong nhiều lĩnh vực như quản lý giao thông, nông nghiệp, và an ninh. Mục tiêu là tạo ra một hệ thống có khả năng nhận diện và đếm đối tượng một cách chính xác, giúp cải thiện hiệu quả công việc và ra quyết định trong các tình huống thực tế.

Để giải quyết bài toán đếm số lượng đối tượng từ video quay bằng máy bay không người lái, có thể áp dụng một số phương pháp kỹ thuật và thuật toán tiên tiến. Một trong những phương pháp phổ biến là sử dụng mạng nơ-ron tích chập (CNN) để nhận diện và đếm đối tượng thông qua quá trình tiền xử lý dữ liệu, huấn luyện mô hình, và áp dụng mô hình lên các khung hình. Ngoài ra, các phiên bản cải tiến của CNN như R-CNN, Fast R-CNN và Faster R-CNN cũng được sử dụng rộng rãi nhờ khả năng đề xuất vùng và phân loại đối tượng hiệu quả. Phương pháp You Only Look Once (YOLO) nổi bật với khả năng xử lý video theo thời gian thực bằng cách chia hình ảnh thành các lưới và dự đoán các hộp giới hạn cho từng ô lưới, cho phép phát hiện và đếm nhiều đối tượng cùng lúc. Single Shot MultiBox Detector (SSD) cung cấp sự cân bằng tốt giữa tốc độ và độ chính xác thông qua việc sử dụng các lưới đa tỉ lệ để phát hiện đối tượng ở các kích thước khác nhau. Ngoài ra, việc kết hợp các kỹ thuật học máy truyền thống với thị giác máy tính, như sử dụng các phương pháp lọc, phát hiện cạnh, và phân đoạn để xác định đối tượng, cùng với các thuật toán học máy như SVM hay Random Forest, cũng là một hướng tiếp cận hiệu quả.

Trong khóa luận này, phương pháp đề xuất bao gồm các bước chính nhằm đếm số lượng đối tượng từ video quay bằng máy bay không người lái. Đầu tiên, một tập dữ liệu được xây dựng từ các hình ảnh và video thu thập từ UAV, đảm bảo đa dạng về điều kiện môi trường và góc nhìn. Tiếp theo, mô hình YOLOv8, một phiên bản tiên tiến của mô hình Object Detection, được huấn luyện trên tập dữ liệu này để nhận diện chính xác các đối tượng mục tiêu. Sau khi mô hình được huấn luyện, nó được

áp dụng lên các hình ảnh để kiểm tra và tinh chỉnh độ chính xác. Tiếp đến, mô hình được sử dụng trên video để nhận diện đối tượng trong từng khung hình. Để theo dõi và đếm số lượng đối tượng, các thuật toán theo vết đối tượng như Simple Online and Realtime Tracking (SORT) và Scene Features-based Simple Online Real-Time Tracker (SFSORT) được tích hợp. Chúng giúp theo vết các đối tượng qua từng khung hình một cách mượt mà và liên tục. Cuối cùng, các đối tượng được đếm dựa trên kết quả theo dõi, cố gắng đảm bảo rằng mỗi đối tượng chỉ được đếm một lần dù xuất hiện trong nhiều khung hình.

Kết quả thử nghiệm cho thấy các phương pháp kết hợp này mang lại kết quả đếm khá tốt về độ chính xác và tốc độ so với các phương pháp khác. Với tập dữ liệu gồm 540 khung hình từ video UAV, hệ thống đạt được tỷ lệ phát hiện chính xác cao và khả năng theo dõi liên tục mà không bị gián đoạn. Đặc biệt là phương pháp kết hợp YOLOv8 với SFSORT đạt được độ chính xác cũng như gán ID cho các đối tượng khá ổn định và tốc độ xử lý rất nhanh. Tổng thể, phương pháp đề xuất chứng minh được khả năng xử lý hiệu quả và chính xác trong việc theo dõi đối tượng, mở ra triển vọng ứng dụng rộng rãi trong các hệ thống giám sát và phân tích video. Ngoài ra còn xây dựng được một ứng dụng đếm đối tượng với giao diện đẹp mắt, thân thiện với người dùng.

MỤC LỤC

LỜI CẢM ƠN	I
LỜI CAM ĐOAN	II
TÓM TẮT KHÓA LUẬN	III
MỤC LỤC.....	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 Tầm quan trọng	2
1.2.1 Tính mới.....	2
1.2.2 Tính hữu dụng	3
1.3 Giới hạn nghiên cứu	3
1.4 Bố cục khóa luận.....	4
Chương 2. Lý thuyết nền tảng.....	5
2.1 Thị giác máy tính	5
2.1.1 Quá trình hình thành ảnh.....	5
2.1.2 Dữ liệu ảnh	5
2.1.3 Các thao tác trên dữ liệu ảnh.....	6
2.1.4 Một số bài toán về thị giác máy tính.....	7
2.2 Deep learning	7
2.2.1 Mạng neuron đa tầng (MLP).....	7
2.2.2 Mạng neuron tích chập.....	9

2.2.2.1 Phép toán Convolution	10
2.2.2.2 Phép toán Padding	13
2.2.2.3 Phép toán Stride	13
2.3 Các phương pháp đánh giá bài toán	15
2.3.1 Ma trận nhầm lẫn	15
2.3.2 Intersection over Union (IoU)	15
2.3.3 Precision	17
2.3.4 Độ nhạy (Recall)	17
2.3.5 Average Precision (AP)	18
2.3.6 Mean Average Precision (mAP)	18
Chương 3. Các công trình liên quan	19
3.1 Mô hình YOLO	19
3.2 Kalman Filter	26
3.2.1 Giới thiệu:	26
3.2.2 Mục tiêu:	26
3.2.3 Các bước cơ bản:	26
3.2.4 Nhận xét:	28
3.3 Giải thuật Hungary:	29
3.3.1 Giới thiệu:	29
3.3.2 Mô hình toán học:	29
3.3.3 Giải thuật Hungary:	30
3.3.4 Các bước cụ thể của thuật toán Hungary:	30
3.3.5 Nhận xét:	35
3.4 SORT	35

3.4.1 Giới thiệu:	35
3.4.2 Nguyên lý hoạt động:	35
3.4.3 Các bước thực hiện:	36
3.4.4 Các thành phần chính của SORT:	37
3.4.5 Ưu điểm.....	41
3.4.6 Nhược điểm:.....	42
3.5 SFSORT	42
3.5.1 Giới thiệu:	42
3.5.2 Chỉ số tương đồng hợp giới hạn:.....	43
3.5.3 Các thành phần chính của SFSORT:.....	45
3.5.3.1 Module liên kết thứ nhất:	45
3.5.3.2 Module liên kết thứ hai:	46
3.5.3.3 Module quản lý theo vết đối tượng:	46
3.5.3.4 Các siêu tham số chính của SFSORT:	47
3.5.4 Ưu điểm của SFSORT:	48
Chương 4. Giải pháp đề xuất	50
4.1 Phân tích dữ liệu.....	50
4.2 Mô hình	51
4.2.1 Object Detection: YOLOv8	53
4.2.2 Kết hợp YOLOv8 với SORT	56
4.2.3 Kết hợp YOLOv8 với SFSORT	57
4.3 Thực nghiệm	59
KẾT LUẬN	66
TÀI LIỆU THAM KHẢO.....	69

DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

STT	KÝ HIỆU, CHỮ VIẾT TẮT	Ý NGHĨA
1	UAV	Máy bay không người lái
2	CNN	Mạng nơ-ron tích chập
3	YOLO	You Only Look Once
4	SSD	Single Shot MultiBox Detector
5	SORT	Simple Online and Realtime Tracking
6	SFSORT	Scene Features-based Simple Online Real-Time Tracker
7	ROI	Khu vực quan tâm
8	MLP	Mạng neuron đa tầng
9	DNN	Deep Neural Network
10	IoU	Intersection over Union
11	Recall	Độ nhạy
12	AP	Average Precision
13	PR curve	Precision-Recall curve
14	mAP	Mean Average Precision
15	BBSI	Chỉ số tương đồng hộp giới hạn

DANH MỤC CÁC BẢNG

Bảng 1: Các phương trình của Kalman Filter.	28
Bảng 2: Ví dụ về giải thuật Hungary (1).....	31
Bảng 3: Ví dụ về giải thuật Hungary (2).....	32
Bảng 4: Ví dụ về giải thuật Hungary (3).....	32
Bảng 5: Ví dụ về giải thuật Hungary (4).....	33
Bảng 6: Ví dụ về giải thuật Hungary (5).....	33
Bảng 7: Ví dụ về giải thuật Hungary (6).....	34
Bảng 8: Ví dụ về giải thuật Hungary (7).....	34
Bảng 9: Các siêu tham số chính của SFSORT.....	47
Bảng 10: Kết quả thực nghiệm theo vết đối tượng và đếm đối tượng của các phương pháp.	61

DANH MỤC CÁC HÌNH VẼ, BIỂU ĐỒ

Hình 1: Cấu trúc của MLP.	8
Hình 2: Cấu trúc của mạng CNN.	9
Hình 3: Phép toán 2D Convolution (1).	10
Hình 4: Phép toán 2D Convolution (2).	11
Hình 5: Phép toán 2D Convolution (3).	11
Hình 6: Phép toán 2D Convolution (4).	11
Hình 7: Phép toán 2D Convolution (5).	12
Hình 8: Phép toán 2D Convolution (6).	12
Hình 9: Padding.....	13
Hình 10: Stride (1).	14
Hình 11: Stride (2).	14
Hình 12: Stride (3).	15
Hình 13: Minh họa công thức tính IoU.....	16
Hình 14: Minh họa so sánh giá trị IoU.....	16
Hình 15: Kiến trúc chung YOLO.....	19
Hình 16: Output của YOLO.....	20
Hình 17: Tiến hóa của YOLO.....	21
Hình 18: Kiến trúc YOLOv1.....	21
Hình 19: Kiến trúc YOLOv3.....	23
Hình 20: Các bước của Kalman Filter.	26
Hình 21: Các bước thực hiện của SORT.....	36

Hình 22: Biểu đồ so sánh tốc độ và độ chính xác của các thuật toán theo dõi đối tượng.	41
Hình 23: Ví dụ minh họa về nhược điểm của SORT.	42
Hình 24: Trực quan hóa các chi tiết tính chỉ số BBSI.	43
Hình 25: So sánh độ chính xác và tốc độ của các thuật toán theo vết đối tượng dựa trên SORT.	48
Hình 26: Ví dụ minh họa ưu điểm của SFSORT.	49
Hình 27: Ví dụ về hình ảnh trong tập dữ liệu.	50
Hình 28: Ví dụ về nhãn trong tập dữ liệu.	51
Hình 29: Kiến trúc YOLOv8.	53
Hình 30: Quá trình huấn luyện mô hình.	56
Hình 31: Kết hợp YOLOv8 với SFSORT.	58
Hình 32: Ma trận nhầm lẫn của mô hình YOLOv8.	59
Hình 33: Kết quả huấn luyện mô hình YOLOv8.	60
Hình 34: Giao diện đếm đối tượng kết hợp YOLOv8 và SORT.	61
Hình 35: Giao diện vẽ ROI khi kết hợp YOLO với SORT.	62
Hình 36: Giao diện kết quả khi kết hợp YOLO với SORT.	62
Hình 37: Giao diện đếm đối tượng kết hợp YOLOv8 và SFSORT.	63
Hình 38: Giao diện vẽ ROI khi kết hợp YOLO với SFSORT.	63
Hình 39: Giao diện kết quả khi kết hợp YOLO với SFSORT.	64
Hình 40: Giao diện huấn luyện mô hình YOLOv8.	64

Chương 1. Giới thiệu bài toán

1.1 Câu hỏi nghiên cứu

Bài toán đếm số đối tượng từ máy bay không người lái đòi hỏi chính xác trong việc nhận diện và đếm các đối tượng cụ thể như con người, phương tiện hoặc động vật từ video hoặc hình ảnh được chụp bởi máy bay không người lái. Mục tiêu của nhiệm vụ là đếm số lượng đối tượng một cách chính xác và xác định vị trí của chúng trong mỗi khung hình hoặc toàn bộ đoạn video. Ứng dụng của bài toán là rất đa dạng, từ giám sát an ninh, quản lý nông nghiệp đến cứu hộ và nghiên cứu môi trường.

Đầu vào của bài toán bao gồm chuỗi các khung hình từ video được ghi lại bằng máy bay không người lái, cùng với các nhãn cho các đối tượng tương ứng trong mỗi khung hình. Mỗi khung hình là một hình ảnh độ phân giải cao, thể hiện cảnh quan từ góc nhìn của UAV và có thể chứa một hoặc nhiều đối tượng cần đếm. Các nhãn cho mỗi khung hình bao gồm thông tin về loại đối tượng có trong đó (ví dụ: xe cộ, con người, động vật) và vị trí của từng đối tượng trên hình ảnh (thường được biểu diễn bằng hộp giới hạn). Để đảm bảo độ chính xác của việc đếm đối tượng, nhãn phải được xác định một cách chính xác và đầy đủ. Đầu vào này sẽ được sử dụng để huấn luyện mô hình nhận diện đối tượng và đánh giá hiệu suất của giải pháp đề xuất.

Đầu ra của bài toán là số lượng đối tượng được đếm thông qua khu vực quan tâm (ROI) từ mỗi khung hình trong video quay từ máy bay không người lái. Mỗi khung hình được xử lý bởi phương pháp đề xuất để nhận diện và đếm số lượng đối tượng mục tiêu. Sau đó, kết quả đếm này được trả về dưới dạng số nguyên không âm. Đối với mỗi khung hình, số lượng đối tượng được đếm có thể là một số nguyên dương hoặc có thể là 0 nếu chưa từng có đối tượng nào được nhận diện nằm trong ROI. Kết quả này được tính toán một cách tự động và liên tục qua các khung hình của video, cung cấp thông tin về sự xuất hiện và phân bố của các đối tượng trong không gian quan sát của UAV. Đối với ứng dụng thực tế, kết quả này có thể được sử dụng để đánh giá lưu lượng giao thông, giám sát hoạt động nông nghiệp, hay quản lý sự kiện, giúp cải thiện quản lý và ra quyết định trong thời gian thực.

Ví dụ, giả sử có một tập dữ liệu gồm 100 khung hình về giao thông được thu thập từ máy bay không người lái và được gán nhãn. Sau quá trình huấn luyện mô hình, khi áp dụng vào dữ liệu thực tế, ta có kết quả như sau: Trong ảnh thứ nhất, có 5 xe được nhận diện, trong khi đó, ảnh thứ hai ghi nhận 6 xe. Đây là kết quả của quá trình dự đoán số lượng xe dựa trên mô hình đã được huấn luyện trên tập dữ liệu.

Tập dữ liệu được sử dụng để giải quyết bài toán bao gồm một chuỗi các khung hình về giao thông, được đọc và chuyển thành ảnh từ video quay từ máy bay không người lái. Mỗi khung hình trong tập dữ liệu thể hiện một tình huống giao thông cụ thể từ góc nhìn của máy bay không người lái. Trước khi áp dụng các phương pháp xử lý và nhận diện, mỗi khung hình được gán nhãn, đánh dấu vị trí và loại hình của các phương tiện, bao gồm xe ô tô, xe tải. Quá trình gán nhãn này đóng vai trò quan trọng trong việc tạo ra dữ liệu huấn luyện chất lượng, cung cấp thông tin cần thiết cho mô hình nhằm nhận diện và đếm số lượng đối tượng một cách chính xác. Điều này giúp tăng cường khả năng tự động hóa trong việc phân tích và quản lý giao thông, đóng góp vào việc cải thiện an toàn và hiệu suất của hệ thống giao thông đô thị.

1.2 Tầm quan trọng

1.2.1 Tính mới

Tính mới của đề tài là sự kết hợp giữa các phương pháp tiên tiến như YOLO với SORT và YOLO với SFSORT, tạo ra một giải pháp đếm số lượng đối tượng từ video ghi lại bằng máy bay không người lái một cách hiệu quả và chính xác hơn. Phương pháp YOLO được sử dụng để nhận diện và đếm số lượng đối tượng trong từng khung hình của video, cung cấp khả năng xử lý nhanh chóng và chính xác trong thời gian thực. Đồng thời, thuật toán SORT và SFSORT được áp dụng để theo dõi các đối tượng qua các khung hình liên tiếp, giúp loại bỏ sự trùng lặp trong việc đếm và cải thiện độ chính xác của kết quả đếm. Sự kết hợp giữa các phương pháp này mang lại tính linh hoạt và hiệu quả cao trong việc đếm số lượng đối tượng từ video ghi lại bằng UAV, phù hợp cho nhiều ứng dụng thực tiễn như giám sát giao thông, quản lý an ninh, hay nghiên cứu môi trường. Điều này đồng nghĩa với việc nâng cao

khả năng quản lý và phân tích dữ liệu từ các hệ thống giám sát không gian một cách thông minh và hiệu quả.

1.2.2 Tính hữu dụng

Đề tài đếm số lượng đối tượng từ video quay bằng máy bay không người lái đem lại giá trị hữu ích trong thực tế bằng cách giải quyết một loạt vấn đề quan trọng. Bằng cách tự động hóa quá trình thu thập và phân tích dữ liệu từ video, đề tài này cung cấp một công cụ linh hoạt và hiệu quả cho nhiều lĩnh vực ứng dụng.

Trong lĩnh vực giao thông, việc đếm số lượng xe cộ và các phương tiện khác từ video UAV giúp cải thiện quản lý giao thông, đánh giá tình trạng ùn tắc và phát triển biện pháp giảm ùn tắc hiệu quả. Nó cũng hữu ích trong lĩnh vực an ninh, nơi nó có thể được sử dụng để giám sát và phát hiện các hoạt động đáng ngờ, đảm bảo an ninh cho các khu vực nhạy cảm như khu vực biên giới hoặc sân bay.

Tóm lại, đề tài này mang lại giá trị hữu ích bằng cách cung cấp một công cụ linh hoạt và tiện lợi cho việc thu thập dữ liệu, phân tích và ra quyết định trong nhiều lĩnh vực ứng dụng, từ giao thông đến an ninh và môi trường.

1.3 Giới hạn nghiên cứu

Để giới hạn phạm vi của bài toán đếm số lượng đối tượng từ video quay bằng máy bay không người lái, ta áp dụng một số ràng buộc và giả định. Trước hết, chúng ta giả định rằng điều kiện ánh sáng trong video là ổn định và không thay đổi đột ngột, từ đó đơn giản hóa việc nhận diện đối tượng. Bài toán cũng chỉ tập trung vào việc đếm số lượng một loại đối tượng cụ thể, và không cần phải nhận diện nhiều loại đối tượng khác nhau. Đồng thời, giả định rằng các đối tượng trong video không chồng lấn lên nhau và không có chướng ngại vật che khuất phần lớn đối tượng. Bằng cách áp dụng các ràng buộc và giả định này, chúng ta có thể tập trung vào việc giải quyết một vấn đề cụ thể một cách hiệu quả, mặc dù cần lưu ý rằng các giả định này có thể cần điều chỉnh tùy thuộc vào ứng dụng cụ thể của bài toán.

1.4 Bố cục khóa luận

Khóa luận này được cấu trúc thành bốn chương, mỗi chương mang đến một góc nhìn sâu sắc và toàn diện về các khía cạnh khác nhau của nghiên cứu.

Chương 1: Giới thiệu bài toán mở ra với việc đặt ra câu hỏi nghiên cứu và nhấn mạnh tầm quan trọng của đề tài. Tính mới và hữu dụng của nghiên cứu được làm rõ, bên cạnh việc xác định các giới hạn nghiên cứu, nhằm mang đến một cái nhìn tổng quát và định hướng cho toàn bộ khóa luận.

Chương 2: Lý thuyết nền tảng là nền móng của nghiên cứu, nơi những kiến thức cơ bản về thị giác máy tính và deep learning được trình bày một cách chi tiết. Từ quá trình hình thành ảnh, các thao tác trên dữ liệu ảnh, đến các mạng neuron đa tầng và tích chập, tất cả được giải thích cặn kẽ. Chương này cũng bao gồm các phương pháp đánh giá quan trọng như ma trận nhầm lẫn, IoU, precision, recall, AP, và mAP, cung cấp nền tảng vững chắc cho các phân tích và thực nghiệm sau này.

Chương 3: Các công trình liên quan giới thiệu và phân tích sâu rộng các mô hình và thuật toán nổi bật trong lĩnh vực, bao gồm YOLO, Kalman Filter, giải thuật Hungary, SORT, và SFSORT. Mỗi phương pháp được mô tả từ nguyên lý hoạt động, ưu nhược điểm, đến các thành phần chính, giúp làm sáng tỏ bức tranh toàn cảnh và sự tiến bộ của các nghiên cứu liên quan.

Chương 4: Giải pháp đề xuất là điểm nhấn của khóa luận, nơi các phân tích dữ liệu và mô hình được đưa vào thực nghiệm. Khóa luận trình bày chi tiết quá trình huấn luyện Object Detection với YOLOv8 và sự kết hợp tinh tế giữa YOLOv8 với SORT và SFSORT. Những thực nghiệm được thiết kế tỉ mỉ để kiểm chứng và đánh giá hiệu quả của các giải pháp đề xuất, từ đó mang lại những kết quả đầy hứa hẹn và khả năng ứng dụng thực tiễn cao.

Khóa luận khép lại với những kết luận và tài liệu tham khảo phong phú, mở ra những hướng nghiên cứu mới và ứng dụng tiềm năng trong tương lai.

Chương 2. Lý thuyết nền tảng

2.1 Thị giác máy tính

2.1.1 Quá trình hình thành ảnh

Quá trình hình thành ảnh bao gồm các bước sau:

- Phát xạ ánh sáng: Nguồn sáng (tự nhiên hoặc nhân tạo) chiếu sáng các vật thể.
- Phản xạ và khúc xạ: Ánh sáng tương tác với vật thể thông qua phản xạ hoặc khúc xạ, mã hóa thông tin về thuộc tính vật thể như màu sắc, kết cấu và hình dạng.
- Thu nhận ánh sáng: Các cảm biến trong hệ thống thị giác máy tính tiếp nhận ánh sáng phản xạ.
- Chuyển đổi tín hiệu quang học: Cảm biến chuyển đổi ánh sáng thành tín hiệu điện tử.
- Xử lý và phân tích tín hiệu: Bộ vi xử lý hoặc các thuật toán xử lý hình ảnh phân tích tín hiệu điện tử để tạo ra hình ảnh số. Các bước xử lý bao gồm: nắn thẳng hình ảnh, lọc nhiễu và trích xuất đặc trưng.
- Lưu trữ và hiển thị: Hình ảnh số được lưu trữ dưới dạng tệp dữ liệu và có thể được hiển thị trên các thiết bị đầu ra như màn hình hoặc sử dụng cho các ứng dụng thị giác máy tính như nhận dạng đối tượng, phân loại ảnh và theo dõi đối tượng.

2.1.2 Dữ liệu ảnh

- Ảnh đơn kênh: Mỗi điểm ảnh được biểu diễn bằng một giá trị duy nhất, thể hiện độ sáng từ đen (0) đến trắng (255) trong ảnh 8-bit.
- Ảnh đa kênh: Mỗi điểm ảnh được biểu diễn bằng một tổ hợp của nhiều giá trị, thường là ba giá trị tương ứng với ba kênh màu (RGB – Red, Green, Blue). Mỗi kênh màu thường có giá trị từ 0 đến 255 trong ảnh 8-bit.

- Độ phân giải: Số lượng điểm ảnh trên mỗi đơn vị diện tích của hình ảnh, thường được đo bằng số lượng điểm ảnh ngang và dọc (ví dụ: 1920x1080 pixel).
- Định dạng tệp ảnh: Các tệp chứa dữ liệu ảnh có thể ở nhiều định dạng khác nhau như JPEG, PNG, BMP, TIFF, GIF, ... Mỗi định dạng có các đặc điểm riêng về nén và chất lượng ảnh.
- Ảnh chuỗi (Video): Dữ liệu video là tập hợp liên tiếp các khung hình ảnh, thường kèm theo âm thanh, để tạo ra chuyển động. Video cũng có thể được mã hóa dưới nhiều định dạng như MP4, AVI, ...
- Ảnh đa phổ: Các ảnh được chụp trong nhiều dải bước sóng khác nhau, từ khả kiến đến cận hồng ngoại và cận tử ngoại, giúp cung cấp thông tin phong phú hơn về vật thể.

2.1.3 Các thao tác trên dữ liệu ảnh

- Chuyển đổi định dạng: Chuyển đổi giữa các định dạng ảnh khác nhau như JPEG, PNG, BMP, TIFF, ...
- Thay đổi kích thước: Thay đổi kích thước của ảnh để phù hợp với yêu cầu của các thuật toán hoặc mô hình, thường sử dụng phương pháp nội suy.
- Cắt ảnh: Lấy một phần cụ thể của ảnh, loại bỏ phần không cần thiết.
- Xoay và lật ảnh: Xoay ảnh theo các góc khác nhau hoặc lật ảnh theo trục ngang hoặc dọc.
- Điều chỉnh độ sáng và độ tương phản: Thay đổi độ sáng và độ tương phản của ảnh để cải thiện chất lượng hoặc làm nổi bật các chi tiết.
- Lọc và làm mờ: Áp dụng các bộ lọc như lọc Gaussian, lọc trung bình, lọc median để làm mờ ảnh hoặc giảm nhiễu.
- Cân bằng màu: Điều chỉnh các kênh màu để cân bằng màu sắc trong ảnh.
- Biến đổi hình học: Thực hiện các phép biến đổi như dịch chuyển, co giãn, biến đổi phối cảnh để thay đổi hình học của ảnh.
- Trích xuất đặc trưng: Trích xuất đặc trưng quan trọng từ ảnh.

- Nâng cao độ phân giải: Sử dụng các thuật toán để tăng độ phân giải của ảnh, cải thiện chi tiết và độ sắc nét.
- Chú thích ảnh: Gán nhãn các đối tượng hoặc vùng trong ảnh để phục vụ cho các bài toán nhận dạng đối tượng, phân loại ảnh.
- Tăng cường dữ liệu: Tạo ra nhiều biến thể của ảnh gốc bằng các thao tác như xoay, lật, cắt, thay đổi độ sáng để tăng cường dữ liệu huấn luyện cho mô hình.

2.1.4 Một số bài toán về thị giác máy tính

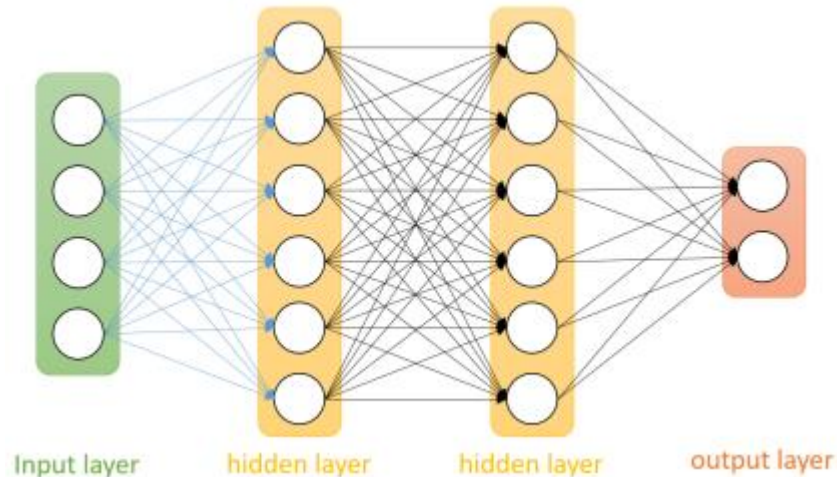
- Phát hiện biên cạnh (Edge Detection): Phát hiện các biên cạnh trong ảnh, nổi bật sự khác biệt giữa các vùng khác nhau của hình ảnh.
- Phân loại ảnh (Image Classification): Đưa ra dự đoán về loại hoặc lớp của ảnh dựa trên nội dung của nó, ví dụ như xe hơi, người, động vật,...
- Phát hiện vật thể (Object Detection): Bài toán này nhằm xác định và phân loại tất cả các đối tượng trong một hình ảnh hoặc video, thường bằng cách sử dụng các hộp giới hạn để bao quanh các vật thể và nhãn để chỉ ra loại đối tượng.
- Theo dõi vật thể (Object Tracking): Theo dõi vị trí của các vật thể qua các khung hình liên tiếp trong một video, từ khung hình này sang khung hình khác.
- Phân đoạn ngữ nghĩa (Semantic Segmentation): Phân chia hình ảnh thành các vùng khác nhau tương ứng với các lớp đối tượng khác nhau mà không cần sử dụng các hộp giới hạn. Mỗi pixel trong hình ảnh được gán nhãn với loại đối tượng tương ứng.
- Phân đoạn theo đối tượng (Instance Segmentation): Mở rộng của phân đoạn ngữ nghĩa, phân đoạn theo đối tượng không chỉ phân loại các vật thể mà còn phân biệt giữa các phiên bản riêng biệt của cùng một loại đối tượng.

2.2 Deep learning

2.2.1 Mạng neuron đa tầng (MLP)

Mạng neuron đa tầng là một thuật toán học máy có giám sát thuộc lớp mạng nơ-ron nhân tạo, được xây dựng trên cơ sở của các tầng perceptron. Được xây dựng từ nhiều perceptron, MLP tổ chức các perceptron này thành từng nhóm tương ứng

với các tầng. Thuật toán này được huấn luyện trên dữ liệu để học một hàm phi tuyến tính, với mục đích phân loại hoặc hồi quy, sử dụng một tập hợp các đặc trưng đầu vào và một biến mục tiêu để đưa ra dự đoán. Đây là một loại mạng neuron truyền thẳng, tức là dữ liệu di chuyển theo một hướng từ tầng đầu vào, qua các tầng ẩn, đến tầng đầu ra mà không có các chu trình phản hồi.



Hình 1: Cấu trúc của MLP.

- Cấu trúc của MLP bao gồm:
 - Tầng đầu vào (Input layer): Nhận các đặc trưng đầu vào của dữ liệu.
 - Các tầng ẩn (Hidden layers): Các tầng neuron giúp học các biểu diễn phức tạp từ dữ liệu.
 - Tầng đầu ra (Output layer): Tầng cuối cùng cho ra dự đoán hoặc phân loại dựa trên dữ liệu đầu vào và các tham số học được.
- Mọi neuron trong một tầng đều kết nối đầy đủ với các neuron trong tầng kế tiếp.
- Mọi tầng (trừ tầng đầu ra), các neuron đều có giá trị bias.
- MLP có hai hay nhiều tầng ẩn gọi là deep neural network (DNN).
- Lan truyền tiến: Dùng để tính toán giá trị đầu ra $\hat{y}^{(i)}$ của $x^{(i)}$:
 - Dữ liệu $x^{(i)}$ được đưa vào mạng neuron thông qua tầng đầu vào.

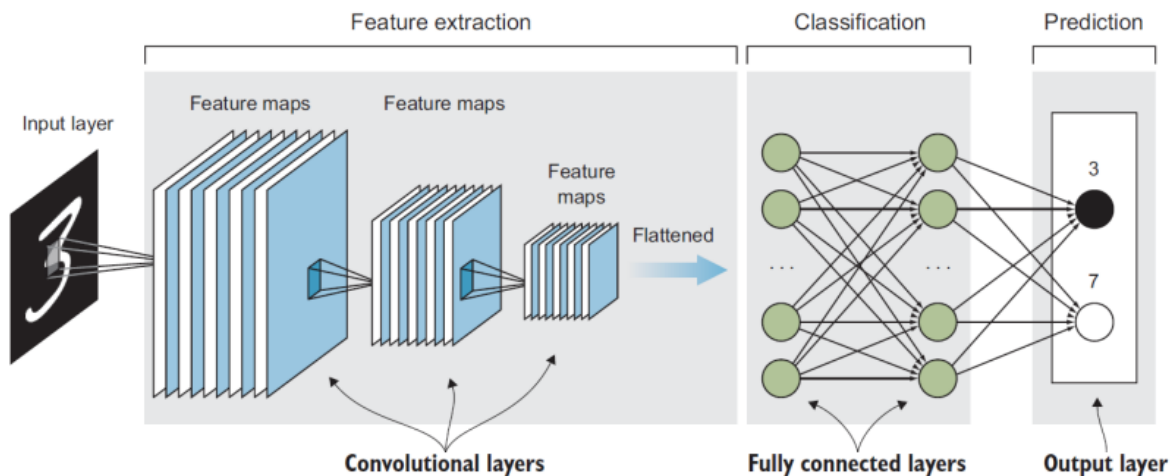
- Giá trị đầu ra của mỗi neuron tính toán tuần tự qua từng tầng, từ tầng ẩn đầu tiên đến tầng cuối cùng.
- Kết quả tại tầng đầu ra là $\hat{y}^{(i)}$ của $x^{(i)}$.
- Lan truyền ngược: Dùng để cập nhật các ma trận trọng số

$W = \{W^{[1]}, W^{[2]}, \dots, W^{[L]}\}$ và vector bias $b = \{b^{[1]}, b^{[2]}, \dots, b^{[L]}\}$

- Tính toán giá trị lỗi của mạng tại tầng đầu ra.
- Xác định mức độ đóng góp của từng neuron trong tầng ẩn cuối cùng vào lỗi của neuron đầu ra và lan truyền ngược lại từ tầng cuối cùng đến tầng đầu vào.

2.2.2 Mạng neuron tích chập

Mạng neuron tích chập là một loại mô hình học sâu được thiết kế để xử lý dữ liệu dưới dạng lưới, chẳng hạn như hình ảnh. CNN đặc biệt hiệu quả trong các tác vụ liên quan đến thị giác máy tính, như nhận diện hình ảnh, phân loại ảnh, và phát hiện đối tượng.



Hình 2: Cấu trúc của mạng CNN.

Cấu trúc của mạng CNN:

- Tầng Convolution: Sử dụng phép toán Convolution và hàm Activation. Tầng này sử dụng các bộ lọc hoặc kernel trượt qua dữ liệu đầu vào để tạo ra các đặc

trung. Mỗi bộ lọc sẽ tạo ra một bản đồ đặc trưng thể hiện các đặc trưng cụ thể của hình ảnh, chẳng hạn như cạnh, góc, hay kết cấu.

- Tầng Pooling: Sử dụng phép toán Pooling. Tầng pooling giảm kích thước không gian của bản đồ đặc trưng, giữ lại các thông tin quan trọng và giảm số lượng tham số và tính toán trong mạng. Các kỹ thuật pooling thông dụng bao gồm max pooling và average pooling.
- Tầng Fully connected và Tầng output: Sử dụng MLP. Nó thường được đặt ở cuối mạng để thực hiện tác vụ phân loại dựa trên các đặc trưng đã trích xuất.

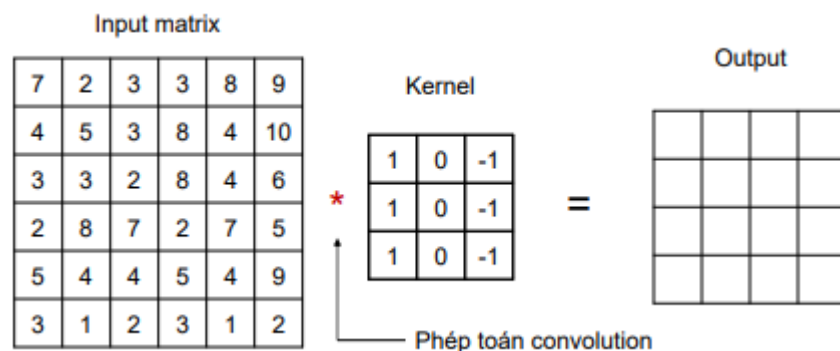
2.2.2.1 Phép toán Convolution

Phép toán Convolution trong lĩnh vực xử lý ảnh và thị giác máy tính là một phương pháp quan trọng, đặc biệt trong mạng nơ-ron tích chập.

- Phép Toán 2D Convolution:

Phép toán Convolution 2D là quá trình chạy một kernel (hay filter) trượt qua toàn bộ hình ảnh. Mỗi neuron trong tầng convolutional layer thực hiện một phép toán convolutional kernel, trong đó các trọng số được sắp xếp thành ma trận convolutional. Cụ thể, điểm ảnh trên hình ảnh gốc được nhân với các trọng số tương ứng trong kernel và tổng hợp lại thành giá trị mới.

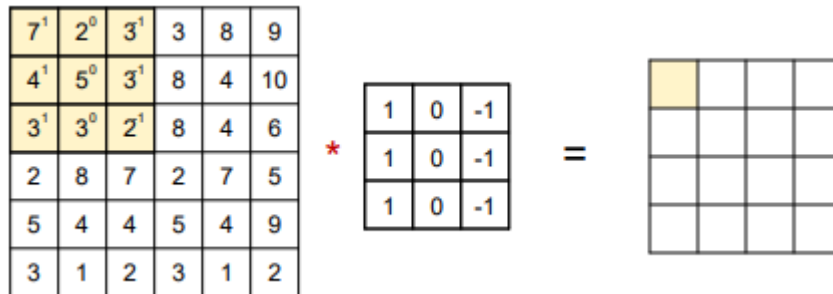
Dưới đây là một ví dụ về phép toán 2D Convolution:



Hình 3: Phép toán 2D Convolution (1).

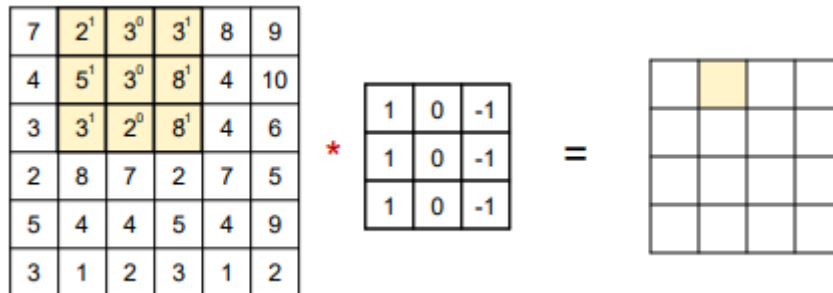
Di chuyển kernel qua từng vị trí trên ma trận đầu vào. Tại mỗi vị trí của kernel, nhân từng phần tử của kernel với các phần tử tương ứng trong ma trận đầu vào. Cộng

tất cả các kết quả nhân lại để có một giá trị duy nhất cho vị trí đó trong ma trận đầu ra.



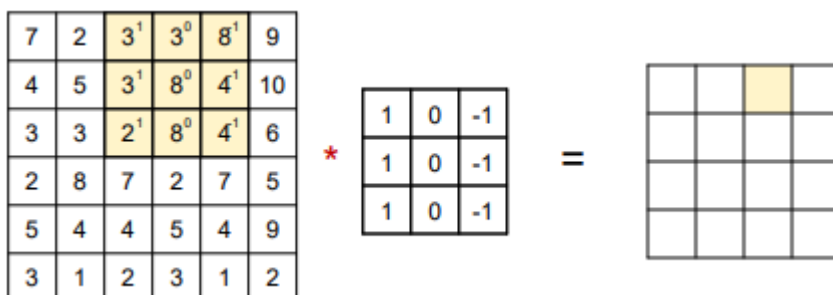
Hình 4: Phép toán 2D Convolution (2).

Tương tự như vậy cho lần di chuyển thứ hai:



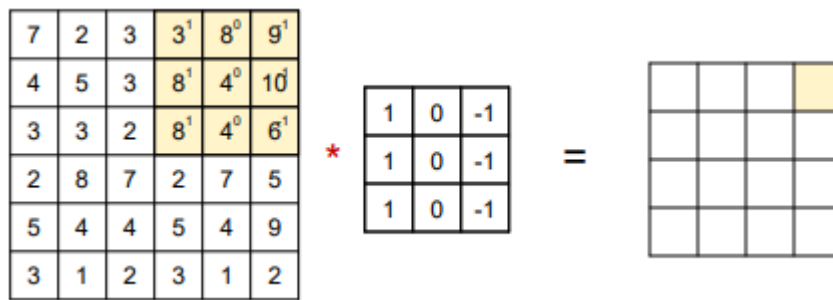
Hình 5: Phép toán 2D Convolution (3).

Tương tự như vậy cho lần di chuyển thứ ba:



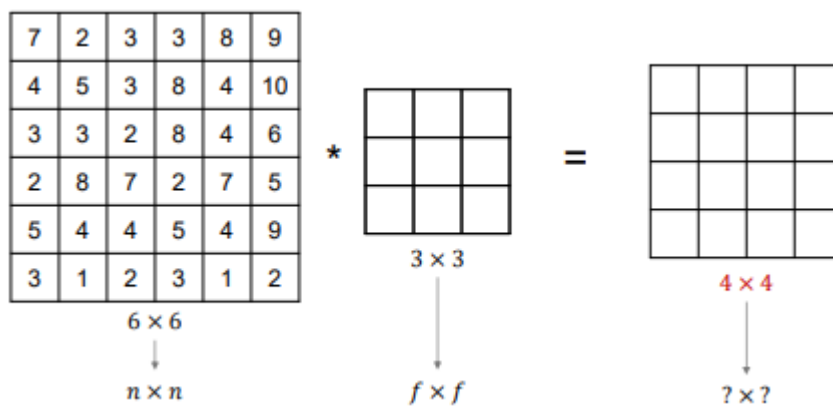
Hình 6: Phép toán 2D Convolution (4).

Tương tự như vậy cho lần di chuyển thứ tư:



Hình 7: Phép toán 2D Convolution (5).

Cứ tiếp tục như vậy đến khi di chuyển hết ma trận đầu vào. Ta sẽ thu được một ma trận đầu ra.



Hình 8: Phép toán 2D Convolution (6).

- Phép toán 3D Convolution:

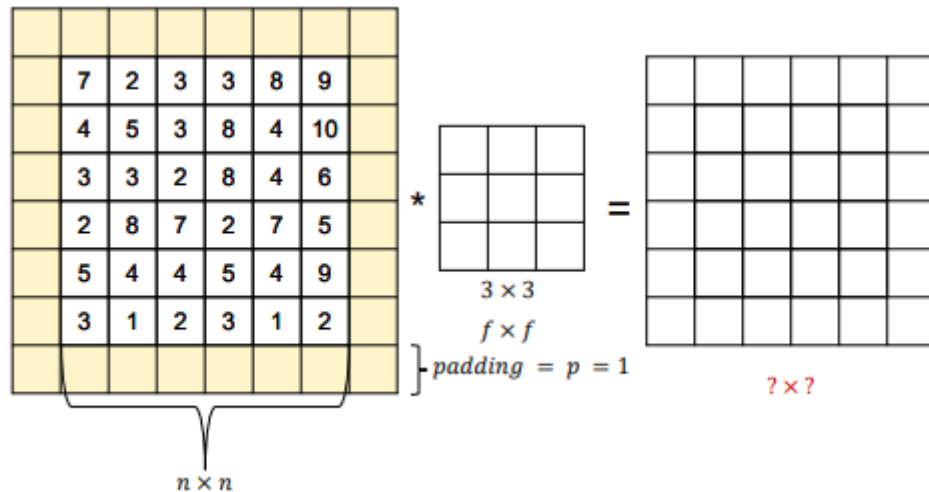
Trong một số ứng dụng như phân tích dữ liệu không gian, phép toán convolution có thể mở rộng sang 3 chiều. Ở đây, kernel không chỉ trượt trên chiều rộng và chiều cao của ảnh mà còn trượt theo chiều sâu.

- Vấn đề của phép toán Convolution:

- Ảnh kết quả nhỏ lại: Do quá trình trượt của kernel, kích thước của ảnh kết quả thường nhỏ hơn so với ảnh ban đầu.
- Biên của ảnh bị mất: Các điểm ảnh ở biên có ít cơ hội được kết hợp với các điểm ảnh khác, dẫn đến việc thông tin biên bị mất.

2.2.2.2 Phép toán Padding

Để giải quyết vấn đề kích thước ảnh kết quả nhỏ lại và mất biên, ta có thể sử dụng kỹ thuật padding. Padding là việc thêm các pixel vào biên của ảnh trước khi áp dụng convolution. Các giá trị pixel được thêm thường là 0 (zero padding). Hình dưới ví dụ về padding:



Hình 9: Padding.

2.2.2.3 Phép toán Stride

Stride là bước nhảy mỗi khi kernel trượt qua ảnh. Bằng cách điều chỉnh Stride, chúng ta có thể kiểm soát kích thước của ảnh kết quả và tốc độ tính toán.

Dưới đây là một ví dụ về phép toán stride = 2. Kernel được đặt vào vị trí đầu tiên của ma trận đầu vào có kích thước bằng với kích thước của kernel.

• Ví dụ: stride = 2

1^0	1^0	2^1	1	0	1	2
3^0	2^1	0^0	2	0	0	1
3^1	1^0	1^0	2	0	2	3
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

 $*$

0	0	1
0	1	0
1	0	0

 $=$

Hình 10: Stride (1).

Sau đó nó sẽ trượt qua 2 bước cho lần di chuyển thứ hai.

• Ví dụ: stride = 2

1	1	2^0	1^0	0^1	1	2
3	2	0^0	2^1	0^0	0	1
3	1	1^1	2^0	0^0	2	3
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

 $*$

0	0	1
0	1	0
1	0	0

 $=$

Hình 11: Stride (2).

Tương tự với những lần di chuyển còn lại.

1	1	2	1	0^0	1^0	2^1
3	2	0	2	0^0	0^1	1^0
3	1	1	2	0^1	2^0	3^0
1	2	1	2	2	0	3
1	3	4	3	2	2	1
0	3	4	3	1	3	2
0	3	4	1	1	1	1

 $*$

0	0	1
0	1	0
1	0	0

 $=$

Hình 12: Stride (3).

2.3 Các phương pháp đánh giá bài toán

2.3.1 Ma trận nhầm lẫn

Ma trận nhầm lẫn là một công cụ hiệu quả để đánh giá hiệu suất của một mô hình phân loại. Nó thể hiện số lượng dự đoán chính xác và sai lầm của từng lớp dưới dạng một bảng.

2.3.2 Intersection over Union (IoU)

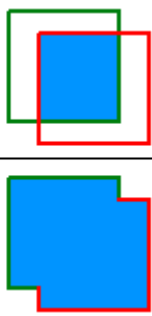
IoU là một thuật ngữ được sử dụng trong lĩnh vực học sâu, đặc biệt là trong các bài toán liên quan đến phát hiện đối tượng và phân vùng hình ảnh. Đây là một chỉ số đo lường độ chồng lấp giữa hai hộp giới hạn. Nó gồm một hộp giới hạn thực tế (B_{gt}) và một hộp giới hạn dự đoán (B_p).

Công thức tính IoU:

$$IoU = \frac{\text{Diện tích } (B_p \cap B_{gt})}{\text{Diện tích } (B_p \cup B_{gt})} \quad (1)$$

Trong đó:

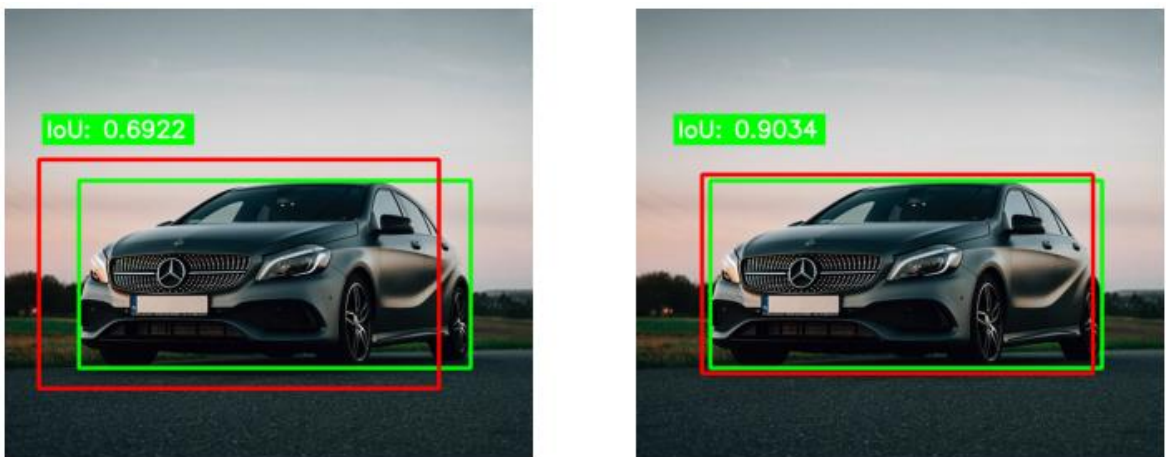
- Diện tích ($B_p \cap B_{gt}$) là diện tích phần giao nhau giữa hai hộp giới hạn.
- Diện tích ($B_p \cup B_{gt}$) là diện tích phần hợp nhất của hai hộp giới hạn, được tính bằng tổng diện tích của cả hai hộp giới hạn trừ đi diện tích phần giao nhau.

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \frac{\text{area of overlap}}{\text{area of union}}$$


Hình 13: Minh họa công thức tính IoU.

Hình trên minh họa IoU giữa một hộp giới hạn thực tế (màu xanh lá) và một hộp giới hạn được phát hiện (màu đỏ).

IoU có giá trị thuộc khoảng từ 0 đến 1 (từ 2 hộp giới hạn không chồng lấp nhau đến 2 hộp giới hạn trùng khớp hoàn toàn).



Hình 14: Minh họa so sánh giá trị IoU.

Ví dụ giả sử hình trên với IoU 0.6922 và 0.9034 thì rõ ràng với mức IoU 0.9034 nhận dự đoán sát với nhãn thực hơn.

Đối với bài toán phát hiện đối tượng, dựa vào IoU ta có các mức:

- True Positive (TP): Phát hiện chính xác. Phát hiện với $IoU \geq \text{ngưỡng}$.
- False Positive (FP): Phát hiện sai. Phát hiện với $IoU < \text{ngưỡng}$.
- False Negative (FN): Một đối tượng không được phát hiện.

- True Negative (TN): Không áp dụng. Trong bài toán phát hiện đối tượng, có nhiều hộp giới hạn có thể không được phát hiện trong ảnh. Do đó, TN sẽ là tất cả các hộp giới hạn không được phát hiện là chính xác (rất nhiều hộp có thể có trong một hình ảnh). Đó là lý do tại sao nó không được sử dụng.

IoU thường được sử dụng để đánh giá hiệu quả của các mô hình phát hiện đối tượng và phân đoạn hình ảnh. Một giá trị IoU cao cho thấy mô hình phát hiện hoặc phân đoạn đối tượng chính xác.

2.3.3 Precision

Precision là một trong những chỉ số quan trọng được sử dụng để đánh giá hiệu suất của mô hình. Precision được định nghĩa là tỷ lệ giữa số lượng dự đoán đúng trên tổng số lượng dự đoán mà mô hình cho là đúng.

Công thức tính precision như sau:

$$Precision = \frac{TP}{TP + FP} \quad (2)$$

Precision phản ánh mức độ chính xác của các dự đoán của mô hình trong việc phát hiện đối tượng. Một precision cao nghĩa là mô hình ít đưa ra các dự đoán sai, tức là nó rất cẩn thận trong việc xác định một đối tượng nào đó.

2.3.4 Độ nhạy (Recall)

Độ nhạy là một chỉ số quan trọng khác để đánh giá hiệu suất của mô hình. Độ nhạy đo lường khả năng của mô hình trong việc phát hiện ra tất cả các đối tượng thực sự có trong hình ảnh.

Công thức tính độ nhạy như sau:

$$Recall = \frac{TP}{TP + FN} \quad (3)$$

Độ nhạy phản ánh khả năng của mô hình trong việc phát hiện tất cả các đối tượng cần nhận diện. Một độ nhạy cao nghĩa là mô hình có khả năng phát hiện ra hầu hết các đối tượng thực sự có trong hình ảnh, tức là ít bỏ sót các đối tượng.

2.3.5 Average Precision (AP)

Từ precision và recall đã tính được như đã trình bày, ta có thể vẽ được Precision-Recall curve (PR curve) cho từng lớp đối tượng riêng biệt. Đường cong này minh họa sự thay đổi của precision đối với recall với các ngưỡng độ tin cậy khác nhau. AP là diện tích phần nằm dưới đường cong PR curve. Giá trị AP càng cao cho thấy mô hình có khả năng phát hiện đối tượng với độ chính xác cao và độ nhạy tốt hơn. Điều này chỉ ra rằng mô hình đang hoạt động hiệu quả hơn trong việc nhận diện các đối tượng trong tập dữ liệu. AP được sử dụng rộng rãi để so sánh và đánh giá các mô hình phát hiện đối tượng, giúp người dùng lựa chọn và cải thiện hiệu suất của mô hình.

2.3.6 Mean Average Precision (mAP)

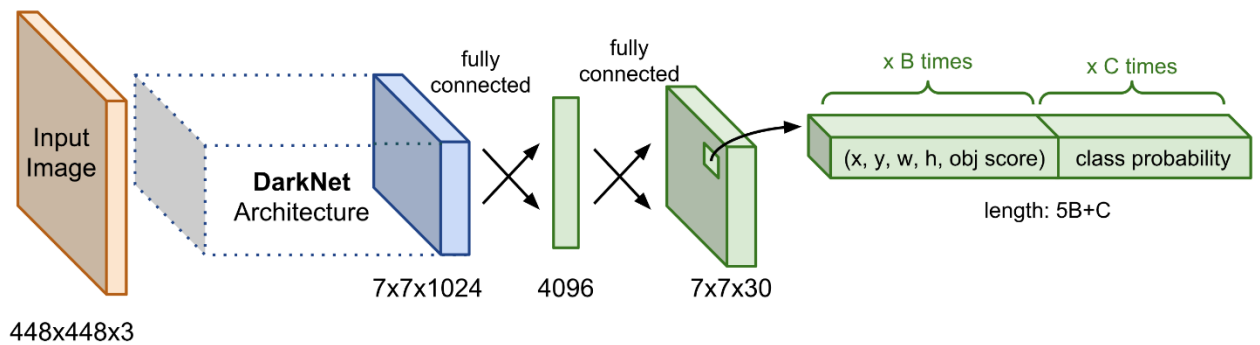
mAP là một chỉ số tổng hợp được sử dụng phổ biến trong các bài toán phát hiện đối tượng. Nó tính trung bình của các giá trị AP của từng lớp đối tượng trong một tập dữ liệu. mAP là một chỉ số quan trọng để đánh giá tổng thể hiệu suất của một mô hình phát hiện đối tượng, bởi vì nó cung cấp cái nhìn tổng quát về khả năng phân loại và nhận diện của mô hình trên các lớp đối tượng khác nhau. mAP càng cao, mô hình càng chính xác và đáng tin cậy.

Chương 3. Các công trình liên quan

3.1 Mô hình YOLO

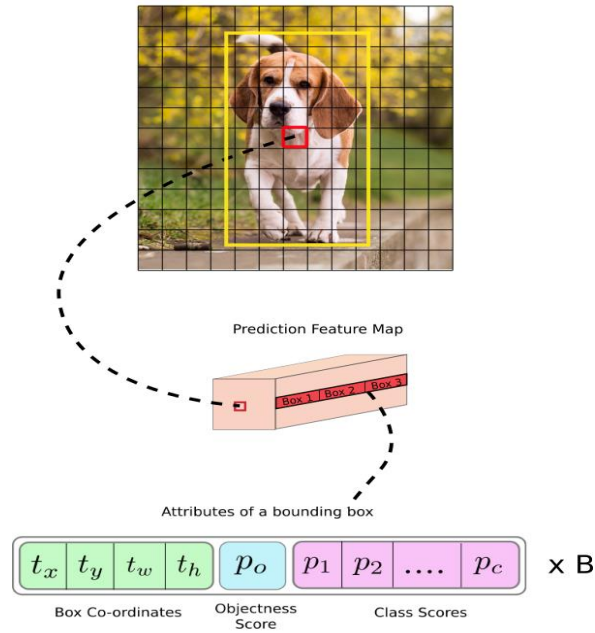
- Kiến trúc chung YOLO:

YOLO [1] có kiến trúc khá cơ bản, bao gồm base network và extra layers. Base network là các mạng Convolution có nhiệm vụ trích xuất đặc trưng ảnh. Extra layers là các layer cuối được áp dụng để phân tích các đặc trưng và phát hiện vật thể trên feature map của base network. Base network thường được dùng là Darknet. Base network của YOLO sử dụng chủ yếu là các tầng convolutional và các tầng fully connected. Kiến trúc chung của YOLO được mô tả như hình sau:



Hình 15: Kiến trúc chung YOLO.

- Output của YOLO:



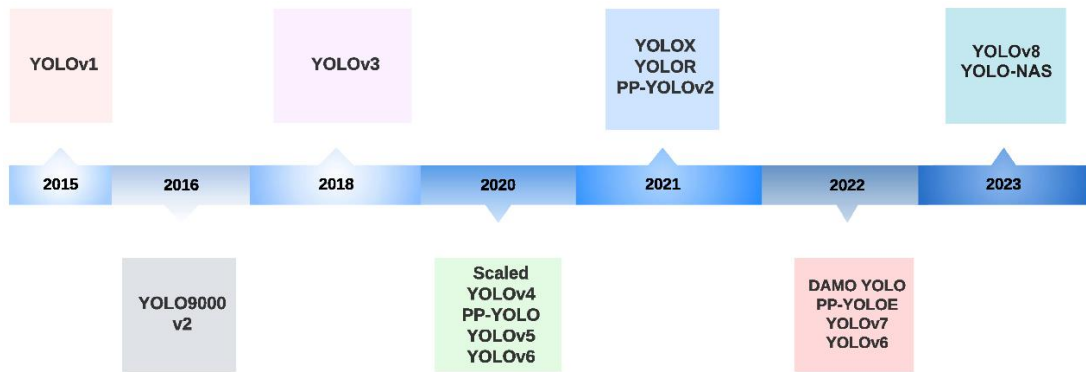
Hình 16: Output của YOLO.

Output của mô hình YOLO là một véc tơ bao gồm các thành phần:

$$y^T = [p_0, (t_x, t_y, t_w, t_h), (p_1, p_2, \dots, p_c)]$$

Trong đó:

- p_0 là xác suất dự báo vật thể xuất hiện trong hộp giới hạn.
 - (t_x, t_y, t_w, t_h) giúp xác định hộp giới hạn. Trong đó t_x, t_y là tọa độ tâm và t_w, t_h là kích thước rộng, dài của hộp giới hạn.
 - (p_1, p_2, \dots, p_c) là véc tơ phân phối xác suất dự báo của các lớp.
- Tiến hóa của Yolo [4, 5]:

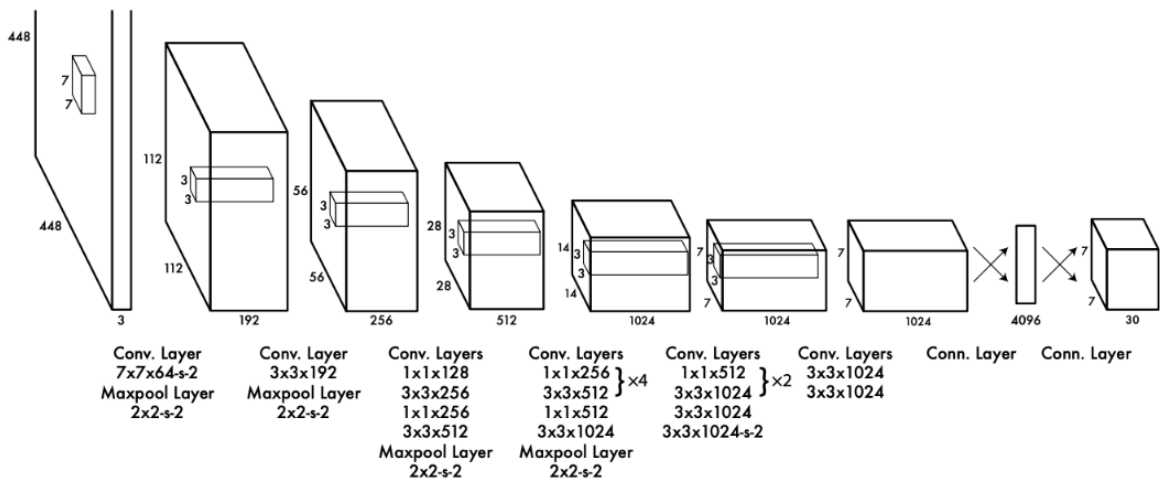


Hình 17: Tiến hóa của YOLO.

YOLOv1:

- Giới thiệu:

YOLOv1 được giới thiệu bởi Joseph Redmon et al. vào năm 2015. Đây là phiên bản đầu tiên của YOLO và đã mang lại một cách tiếp cận mới mẻ cho bài toán phát hiện đối tượng. Kiến trúc của YOLOv1 được mô tả ở hình dưới:



Hình 18: Kiến trúc YOLOv1.

- Điểm mới:

- Sử dụng một mạng neuron duy nhất để dự đoán các hộp giới hạn và tên lớp của các đối tượng trực tiếp từ hình ảnh đầu vào.
- Chia ảnh đầu vào thành các lưới và mỗi ô lưới sẽ dự đoán hộp giới hạn nếu trung tâm của đối tượng rơi vào ô đó.

- Ưu điểm:
 - YOLOv1 có khả năng xử lý hình ảnh nhanh chóng nhờ vào việc xử lý chỉ trong một lần.
- Nhược điểm:
 - So với các phương pháp truyền thống như R-CNN, YOLOv1 thường có độ chính xác thấp hơn.
 - Do sử dụng grid cells lớn, YOLOv1 gặp khó khăn trong việc phát hiện các đối tượng nhỏ hoặc bị che khuất.

YOLOv2:

- Giới thiệu:

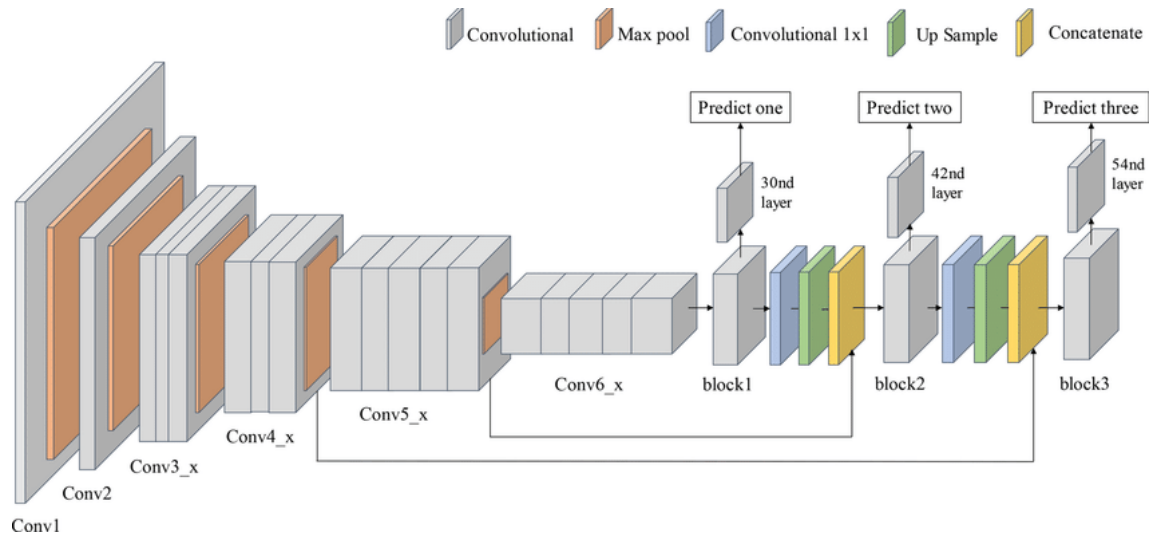
YOLOv2, còn được gọi là YOLO9000, được giới thiệu vào năm 2016 với nhiều cải tiến đáng kể so với YOLOv1.

- Điểm mới:
 - Sử dụng anchor boxes để cải thiện khả năng phát hiện các đối tượng với kích thước và tỷ lệ khác nhau.
 - Sử dụng batch normalization để tăng tốc độ huấn luyện và cải thiện hiệu suất.
- Ưu điểm:
 - Nhờ vào các cải tiến như anchor boxes và batch normalization, YOLOv2 có độ chính xác cao hơn YOLOv1.
 - Vẫn duy trì được tốc độ nhanh như phiên bản trước.
- Nhược điểm:
 - Việc sử dụng anchor boxes và các cải tiến khác làm tăng độ phức tạp của mô hình.

YOLOv3:

- Giới thiệu:

YOLOv3 được giới thiệu vào năm 2018, mang đến nhiều cải tiến về mặt kiến trúc và hiệu suất. Hình dưới mô tả kiến trúc của YOLOv3:



Hình 19: Kiến trúc YOLOv3.

- Điểm mới:
 - Sử dụng mạng backbone Darknet-53 mạnh mẽ hơn.
 - Dự đoán các đối tượng ở ba thang đo khác nhau để cải thiện khả năng phát hiện các đối tượng nhỏ và lớn.
 - Sử dụng hàm sigmoid cho các lớp đầu ra thay vì softmax.
- Ưu điểm:
 - Cải thiện đáng kể về độ chính xác so với YOLOv2.
 - YOLOv3 cải thiện khả năng phát hiện các đối tượng nhỏ.
- Nhược điểm:
 - Sử dụng Darknet-53 làm tăng kích thước và độ phức tạp của mô hình.

YOLOv4:

- Giới thiệu:

YOLOv4 được giới thiệu vào năm 2020, tiếp tục cải tiến hiệu suất và độ chính xác.

- Điểm mới:

- Sử dụng CSPDarknet53 làm backbone để giảm thiểu tính toán và cải thiện hiệu suất.
- Sử dụng hàm kích hoạt Mish thay vì ReLU.
- Ưu điểm:
 - Cải thiện đáng kể về độ chính xác và tốc độ so với YOLOv3.
 - Dễ dàng tích hợp và triển khai trên nhiều nền tảng.
- Nhược điểm:
 - Các cải tiến làm tăng độ phức tạp của mô hình, đòi hỏi nhiều tài nguyên tính toán hơn.

YOLOv5:

- Giới thiệu:

YOLOv5 được giới thiệu bởi Ultralytics vào năm 2020. Mặc dù không phải là phiên bản chính thức từ tác giả ban đầu của YOLO, YOLOv5 nhanh chóng trở nên phổ biến nhờ vào hiệu suất và tính tiện dụng.

- Điểm mới:
 - Được triển khai hoàn toàn trên PyTorch, dễ dàng sử dụng và mở rộng.
 - Tự động điều chỉnh các anchor boxes để phù hợp với dữ liệu huấn luyện.
 - Nhiều cải tiến trong quy trình huấn luyện giúp tăng tốc độ và hiệu suất.
- Ưu điểm:
 - Có sẵn các công cụ và tài liệu hỗ trợ tốt, dễ dàng tích hợp và triển khai.
 - Đạt được sự cân bằng tốt giữa tốc độ và độ chính xác.
- Nhược điểm:
 - Không được phát triển bởi tác giả ban đầu của YOLO, có thể thiếu một số cải tiến từ các phiên bản chính thức.
 - Yêu cầu tài nguyên tính toán cao.

YOLOv6:

- Giới thiệu:

YOLOv6 được giới thiệu vào năm 2022 bởi công ty Meituan. Đây là một phiên bản nâng cao với nhiều cải tiến về mặt kỹ thuật.

- Điểm mới:

- Chuyển từ anchor-based sang anchor-free, giúp đơn giản hóa mô hình và tăng hiệu suất.
- Sử dụng RepVGG để cải thiện tốc độ suy luận và hiệu suất.

- Ưu điểm:

- Tối ưu hóa cho các thiết bị có tài nguyên hạn chế.
- Đạt được độ chính xác cao, tốc độ nhanh.

- Nhược điểm:

- Yêu cầu tài nguyên tính toán cao hơn.

YOLOv7:

- Giới thiệu:

YOLOv7 được giới thiệu vào năm 2023 với nhiều cải tiến vượt bậc về mặt hiệu suất và độ chính xác.

- Điểm mới:

- Mạng lưới được mở rộng và cải tiến để xử lý tốt hơn các đối tượng nhỏ và phức tạp.
- Áp dụng các kỹ thuật augmentation tiên tiến để cải thiện khả năng tổng quát hóa của mô hình.
- Kết hợp các kỹ thuật anchor-based và anchor-free để tận dụng ưu điểm của cả hai.

- Ưu điểm:

- Cải thiện đáng kể về độ chính xác và tốc độ so với các phiên bản trước.

- Nhược điểm:

- Mô hình phức tạp hơn, yêu cầu nhiều tài nguyên tính toán hơn để huấn luyện và triển khai.

3.2 Kalman Filter

3.2.1 Giới thiệu:

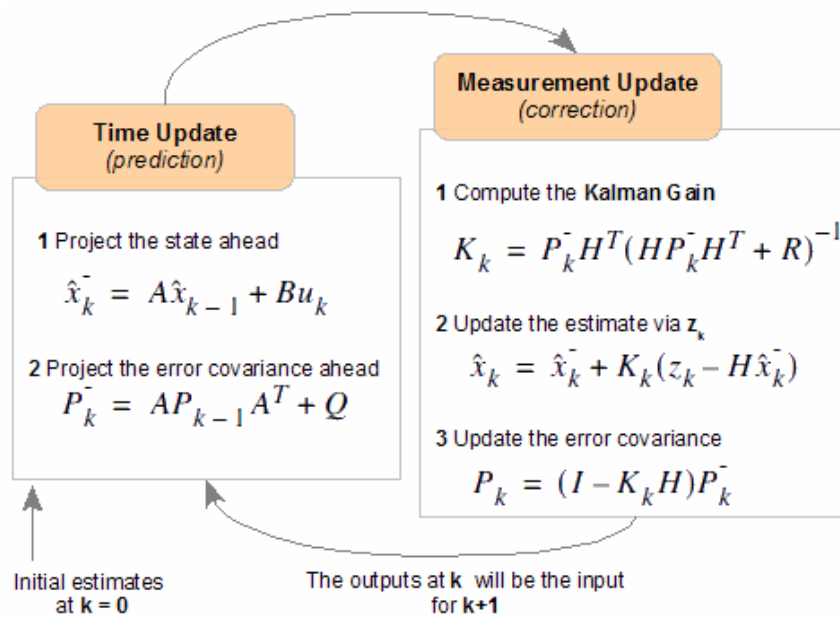
Kalman Filter [9] là một công cụ mạnh mẽ trong lĩnh vực xử lý tín hiệu và ước lượng thống kê. Nó được phát triển bởi Rudolf E. Kálmán vào những năm 1960 và đã được áp dụng rộng rãi trong nhiều lĩnh vực như xử lý hình ảnh, điều khiển tự động, thị trường tài chính, và nhiều ứng dụng khác. Bộ lọc này cho phép ta ước lượng trạng thái của một hệ thống dựa trên dữ liệu đo và mô hình dự đoán.

3.2.2 Mục tiêu:

Mục tiêu chính của Bộ lọc Kalman là ước lượng trạng thái của một hệ thống dựa trên các đo lường và mô hình hệ thống. Bộ lọc này tối ưu hóa việc ước lượng bằng cách sử dụng các thông tin trước đó và dự đoán tương lai.

3.2.3 Các bước cơ bản:

Các bước cơ bản của Kalman Filter được mô tả như sau:



Hình 20: Các bước của Kalman Filter.

- Bước 1: Xây dựng mô hình:

Hai phương trình của Bộ lọc Kalman như sau:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \quad (4)$$

$$z_k = Hx_k + v_k \quad (5)$$

Trong đó:

- x_k : Là trạng thái ước lượng tại thời điểm k .
- u_k : Là vector điều khiển tại thời điểm k .
- w_{k-1} : Là nhiễu quá trình, biểu thị sự không chắc chắn trong dự đoán trạng thái.
- z_k : Là đo lường tại thời điểm k , có thể là các giá trị thực sự mà chúng ta đo được từ hệ thống.
- H : Là ma trận quan hệ giữa trạng thái hệ thống và các đo lường.
- v_k : Là nhiễu đo lường, biểu thị sự không chắc chắn trong việc đo lường.
- A : Là ma trận chuyển trạng thái liên hệ giữa trạng thái trước đó và trạng thái hiện tại.
- B : Là ma trận vào điều khiển.

Trong bước này, chúng ta xác định các thành phần của mô hình Kalman như ma trận chuyển đổi trạng thái (A), ma trận điều khiển (B), ma trận đo lường (H), và các nhiễu quá trình và đo lường.

- Bước 2: Bắt đầu quá trình:

Trong bước này, chúng ta xác định các giá trị ban đầu và các thông số cần thiết khác cho quá trình ước lượng. Điều này bao gồm cả ước lượng ban đầu của trạng thái (x_0) và ma trận hiệp phương sai ban đầu (P_0).

Các phương trình của bộ lọc Kalman được chia làm hai nhóm: phương trình dự đoán và phương trình cập nhật. Phương trình dự đoán có chức năng dự đoán trạng thái hiện tại và ước lượng hiệp phương sai để có tiền ước lượng cho thời gian tiếp

theo. Phương trình cập nhật có chức năng hiệu chỉnh hậu ước lượng bằng cách kết hợp kết quả phép đo và tiên ước lượng. Các phương trình được mô tả như bảng sau:

Dự đoán	Cập nhật
<p>Trạng thái hiện tại được dự đoán:</p> $\hat{x}_{\bar{k}} = A\hat{x}_{k-1} + Bu_k \quad (6)$ <p>Ma trận hiệp phương sai sai số được dự đoán:</p> $P_{\bar{k}} = AP_{k-1}A^T + Q \quad (7)$	<p>Độ lợi Kalman được tính toán:</p> $K_k = P_{\bar{k}}H^T(HP_{\bar{k}}H^T + R)^{-1} \quad (8)$ <p>Sau đó thực hiện phép đo thực tế z_k.</p> <p>Cập nhật trạng thái dự đoán:</p> $\hat{x}_k = \hat{x}_{\bar{k}} + K_k(z_k - H\hat{x}_{\bar{k}}) \quad (9)$ <p>Cập nhật hiệp phương sai sai số:</p> $P_k = (I - K_kH)P_{\bar{k}} \quad (10)$ <p>(I là ma trận đơn vị)</p>

Bảng 1: Các phương trình của Kalman Filter.

– Bước 3: Lặp lại:

Bước này bao gồm lặp lại các phương trình dự đoán và cập nhật cho mỗi bước thời gian. Trong phương trình dự đoán, chúng ta dự đoán trạng thái tiếp theo dựa trên trạng thái hiện tại và dữ liệu điều khiển. Trong phương trình cập nhật, chúng ta cập nhật ước lượng của trạng thái dựa trên dữ liệu đo lường mới.

3.2.4 Nhận xét:

Bộ Lọc Kalman là một công cụ quan trọng trong xử lý tín hiệu và ước lượng thống kê. Nó cho phép ta ước lượng trạng thái của một hệ thống dựa trên dữ liệu đo và mô hình dự đoán, và đã được áp dụng rộng rãi trong nhiều lĩnh vực.

3.3 Giải thuật Hungary:

3.3.1 Giới thiệu:

Giải thuật Hungary [10], được phát triển và công bố vào năm 1955, là một phương pháp quan trọng trong giải quyết bài toán phân công công việc. Bài toán này đặt ra vấn đề làm sao để phân công n người cho n công việc sao cho tổng chi phí là nhỏ nhất.

3.3.2 Mô hình toán học:

Bài toán được mô hình hóa dưới dạng một bài toán tối ưu với mục tiêu tìm ra phương án phân công với chi phí nhỏ nhất. Mô hình toán học của bài toán là:

$$z = \sum_{i=1}^n \sum_{j=1}^n c_{ij} x_{ij} \rightarrow \min \quad (11)$$

Trong đó:

- c_{ij} là chi phí của việc giao cho người thứ i làm công việc thứ j .
- x_{ij} là biến nhị phân, bằng 1 nếu người i được giao công việc j , và bằng 0 nếu ngược lại.

Các ràng buộc của bài toán là:

- Mỗi người chỉ được giao một công việc:

$$\sum_{j=1}^n x_{ij} = 1, \text{ với } i = 1, 2, \dots, n \quad (12)$$

- Mỗi công việc chỉ được giao cho một người:

$$\sum_{i=1}^n x_{ij} = 1, \text{ với } j = 1, 2, \dots, n \quad (13)$$

- Biến nhị phân:

$$x_{ij} = 0 \text{ hoặc } 1, \text{ với } i = 1, 2, \dots, n \text{ và } j = 1, 2, \dots, n \quad (14)$$

Các số x_{ij} thỏa mãn các điều kiện trên gọi là một phương án phân công, hay ngắn gọn là một phương án. Một phương án đạt cực tiểu của z được gọi là một phương án tối ưu hay lời giải của bài toán.

3.3.3 Giải thuật Hungary:

Giải thuật Hungary dựa trên hai định lý quan trọng:

- Định lý 1: Giả sử ma trận chi phí của bài toán giao việc là không âm và có ít nhất n phần tử bằng 0. Hơn nữa nếu n phần tử 0 này nằm ở n hàng khác nhau và n cột khác nhau thì phương án giao cho người i thực hiện công việc tương ứng với số 0 này ở hàng i sẽ là phương án tối ưu (lời giải) của bài toán.
- Định lý 2: Cho $C = [c_{ij}]$ là ma trận chi phí của bài toán giao việc (n người, n việc) và $X^* = [x_{ij}]$ là một lời giải (phương án tối ưu) của bài toán này. Giả sử C' là ma trận nhận được từ C bằng cách thêm số $\alpha \neq 0$ (dương hoặc âm) vào mỗi phần tử ở hàng r của C . Khi đó X^* cũng là lời giải của bài toán giao việc với ma trận chi phí C' .

Thuật toán Hungary dựa vào 2 định lý này, từ đó hình thành được hướng xử lý bài toán: Biến đổi ma trận (cộng trừ vào các hàng hoặc cột) để đưa về ma trận có n phần tử bằng 0 nằm ở các hàng và cột khác nhau. Sau đó, lấy ra phương án tối ưu là các vị trí chứa các phần tử 0 này.

3.3.4 Các bước cụ thể của thuật toán Hungary:

- Bước 1 (Bước chuẩn bị). Trừ các phần tử trên mỗi hàng của C cho phần tử nhỏ nhất trên hàng đó, tiếp theo trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó. Kết quả ta nhận được ma trận C' có tính chất: trên mỗi hàng, cột có ít nhất một phần tử 0 và bài toán giao việc với ma trận C' có cùng lời giải như bài toán với ma trận C .
- Bước 2: Vẽ một số tối thiểu các đường thẳng trên dòng và cột để đảm bảo mọi phần tử 0 đều được đi qua.

- Bước 3: Nếu có n đường thẳng được vẽ, kết thúc thuật toán và tiến hành phân công công việc. Nếu số đường thẳng được vẽ nhỏ hơn n , vẫn chưa tìm được phương án phân công tối ưu, tiến hành bước tiếp theo.
- Bước 4: Tìm phần tử chưa bị gạch nhỏ nhất và lấy tất cả các phần tử chưa bị gạch trừ đi phần tử nhỏ nhất đó; các số bị gạch bởi 2 đường thẳng cộng với số đó; còn các số khác giữ nguyên. Ta được ma trận C'' có cùng lời giải với ma trận C' . Sau đó quay lại Bước 2.

Ví dụ:

Trong một tổ sản xuất có 4 công việc I, II, III, IV cần bố trí cho 4 công nhân A, B, C, D.

Thời gian thực hiện cho mỗi công việc của từng công nhân cho ở bảng sau.

Tìm phương án bố trí công việc sao cho tổng thời gian thực hiện các công việc là nhỏ nhất.

Đơn vị tính: phút

Công việc Nhân viên	I	II	III	IV
A	18	52	64	39
B	75	55	19	48
C	35	57	8	65
D	27	25	14	16

Bảng 2: Ví dụ về giải thuật Hungary (1).

- Bước 1: Trừ các phần tử trên mỗi hàng của C cho phần tử nhỏ nhất trên hàng đó, tiếp theo trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó.

Công việc Nhân viên	I	II	III	IV
A	0	23	46	19
B	56	25	0	27
C	27	38	0	55
D	13	0	0	0

Bảng 3: Ví dụ về giải thuật Hungary (2).

- Bước 2: Vẽ một số tối thiểu các đường thẳng trên dòng và cột để đảm bảo mọi phần tử 0 đều được đi qua.

Công việc Nhân viên	I	II	III	IV
A	0	23	46	19
B	56	25	0	27
C	27	38	0	55
D	13	0	0	0

Bảng 4: Ví dụ về giải thuật Hungary (3).

Số đường thẳng được vẽ nhỏ hơn 4, vẫn chưa tìm được phương án phân công tối ưu, tiến hành bước tiếp theo.

- Bước 4: Tìm phần tử chưa bị gạch nhỏ nhất và lấy tất cả các phần tử chưa bị gạch trừ đi phần tử nhỏ nhất đó; các số bị gạch bởi 2 đường thẳng cộng với số đó; còn các số khác giữ nguyên.

Công việc Nhân viên	I	II	III	IV
A	0	4	46	0
B	56	6	0	8
C	27	19	0	36
D	32	0	19	0

Bảng 5: Ví dụ về giải thuật Hungary (4).

- Quay lại Bước 2:

Công việc Nhân viên	I	II	III	IV
A	0	4	46	0
B	56	6	0	8
C	27	19	0	36
D	32	0	19	0

Bảng 6: Ví dụ về giải thuật Hungary (5).

Số đường thẳng được vẽ vẫn nhỏ hơn 4, vẫn chưa tìm được phương án phân công tối ưu, tiến hành bước tiếp theo.

Công việc Nhân viên	I	II	III	IV
A	0	4	52	0
B	50	0	0	2
C	21	13	0	30
D	32	0	25	0

Bảng 7: Ví dụ về giải thuật Hungary (6).

- Quay lại bước 2:

Công việc Nhân viên	I	II	III	IV
A	0	4	52	0
B	50	0	0	2
C	21	13	0	30
D	32	0	25	0

Bảng 8: Ví dụ về giải thuật Hungary (7).

Số đường thẳng được vẽ đã bằng 4, đã tìm được phương án phân công tối ưu.
Kết thúc giải thuật.

Như vậy ta bố trí:

- Nhân viên A thực hiện công việc 1 với thời gian là 18 phút;
- Nhân viên B thực hiện công việc 2 với thời gian là 55 phút;
- Nhân viên C thực hiện công việc 3 với thời gian là 8 phút;

- Nhân viên D thực hiện công việc 4 với thời gian là 16 phút;

Tổng thời gian thực hiện công việc của cả 4 nhân viên là 97 phút.

3.3.5 Nhận xét:

Giải thuật Hungary là một phương pháp hiệu quả để giải quyết bài toán phân công công việc, đặc biệt là trong trường hợp chi phí không âm và có ít nhất n phần tử 0 khác nhau. Việc áp dụng thuật toán này không chỉ giúp giảm thiểu chi phí mà còn tăng hiệu suất và hiệu quả của các quy trình phân công công việc trong thực tế. Ngoài ra, trong bài toán theo vết đối tượng nó được sử dụng để xác định gán nhãn tốt nhất giữa các đối tượng được phát hiện trong khung hình hiện tại và các đối tượng đã được theo dõi từ các khung hình trước đó sao cho sai số liên kết nhỏ nhất.

3.4 SORT

3.4.1 Giới thiệu:

SORT [11-13] là một thuật toán theo dõi các đối tượng trong các hình ảnh hoặc video. SORT kết hợp bộ lọc Kalman và thuật toán Hungary để theo dõi các đối tượng trong video. Nó được phát triển để cải thiện hiệu suất của hệ thống theo dõi trong thời gian thực trong các ứng dụng như theo dõi đối tượng trong các hệ thống giám sát an ninh, xe tự hành và robot di động.

3.4.2 Nguyên lý hoạt động:

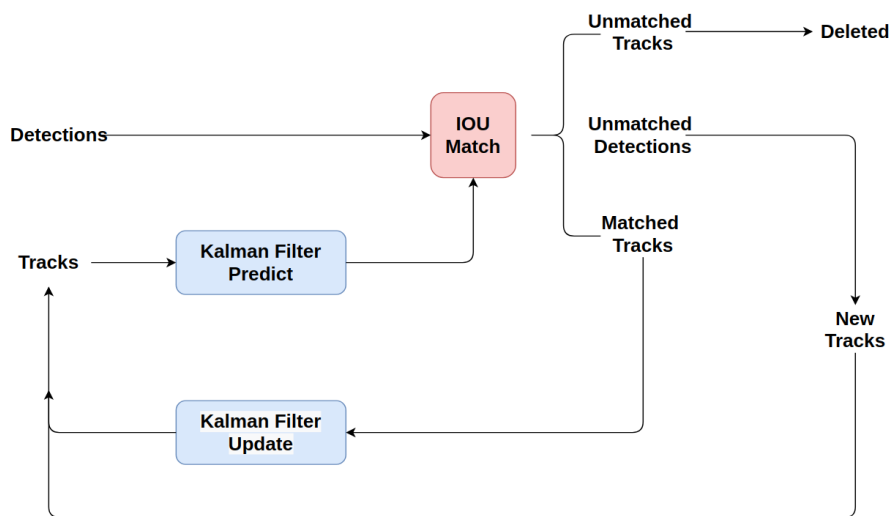
SORT là một thuật toán trong họ thuật toán theo dõi và phát hiện (Tracking-by-detection), được thiết kế đặc biệt cho các ứng dụng theo dõi thời gian thực. Phương pháp này nổi bật với khả năng nhận dạng các đối tượng nhanh chóng. Một điểm đặc biệt của các thuật toán Tracking-by-detection là chúng phân tách việc phát hiện đối tượng và theo dõi chúng thành hai bài toán riêng biệt, từ đó tối ưu hóa kết quả trong mỗi bài toán. Quá trình xử lý bao gồm các bước chính sau:

- Phát hiện đối tượng: Phát hiện các đối tượng trong từng khung hình của video.

- Dự đoán vị trí: Dự đoán vị trí tiếp theo của các đối tượng dựa trên vị trí của chúng trong các khung hình trước đó.
- Liên kết: Các vị trí đã phát hiện được liên kết với các dự đoán để gán cho mỗi đối tượng một ID duy nhất, giúp theo dõi chúng qua các khung hình.

3.4.3 Các bước thực hiện:

Các bước thực hiện của thuật toán SORT được mô tả như hình sau:



Hình 21: Các bước thực hiện của SORT.

- Bước 1: Thuật toán SORT tiến hành áp dụng Kalman Filter để dự đoán các trạng thái track mới dựa trên các track trong quá khứ.
- Bước 2: Dựa trên các trạng thái track dự đoán và các phát hiện đối tượng thu được từ các bộ phát hiện đối tượng như YOLO, xây dựng ma trận chi phí cho bài toán phân bổ. Chi phí được sử dụng để đánh giá ở đây là giá trị IoU giữa các hộp giới hạn của track và các phát hiện đối tượng.
- Bước 3: Sử dụng giải thuật Hungary để giải quyết bài toán phân bổ với ma trận chi phí đã được xây dựng, nhằm xác định sự liên kết tối ưu giữa các track và các phát hiện đối tượng.

- Bước 4: Xử lý, phân loại các phát hiện đối tượng để đảm bảo chúng được phân loại chính xác.
- Bước 5: Sử dụng Kalman Filter để cập nhật những phát hiện đối tượng đã được liên kết với track.

3.4.4 Các thành phần chính của SORT:

- Hàm `linear_assignment`: Sử dụng giải thuật Hungary để gán các phát hiện vào các theo dõi dựa trên ma trận chi phí, thường là ma trận IoU giữa các phát hiện và theo dõi.

```
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)))
```

- Lớp `KalmanBoxTracker` được sử dụng để theo dõi các đối tượng trong hệ thống theo dõi video, sử dụng mô hình Kalman Filter để dự đoán và cập nhật trạng thái của các đối tượng được quan sát dưới dạng hộp giới hạn. Đây là một phần của hệ thống theo dõi đa đối tượng. Các phương thức chính:
 - `__init__(self, bbox)`: Phương thức khởi tạo, tạo một tracker mới dựa trên hộp giới hạn ban đầu. Các tham số của Kalman Filter được thiết lập để mô hình tốc độ vật lý cố định.

```
def __init__(self, bbox):
    """
    Initialises a tracker using initial bounding box.
    """
    #define constant velocity model
    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]
])

self.kf.R[2:,2:] *= 10.
self.kf.P[4:,4:] *= 1000. #give high uncertainty to the unobservable
initial velocities
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

```

- `update(self, bbox)`: Cập nhật trạng thái của một đối tượng đang được theo dõi với thông tin từ một hộp giới hạn mới được quan sát. Nó đảm bảo rằng trạng thái của đối tượng được duy trì chính xác và liên tục dựa trên các quan sát mới, đồng thời theo dõi số lần đối tượng được phát hiện và chuỗi lần phát hiện liên tiếp.

```

def update(self, bbox):
    """
    Updates the state vector with observed bbox.
    """
    self.time_since_update = 0
    self.history = []
    self.hits += 1
    self.hit_streak += 1
    self.kf.update(convert_bbox_to_z(bbox))

```

- `predict(self)`: Dự đoán vị trí hộp giới hạn tiếp theo của đối tượng. Phương thức này dự đoán các thông số của Kalman Filter để tính toán hộp giới hạn tiếp theo và cập nhật các thống kê như tuổi đối tượng và chuỗi phát hiện.

```
def predict(self):
    """
    Advances the state vector and returns the predicted bounding box
    estimate.
    """
    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]
```

- Lớp SORT: Các phương thức chính:

- Khởi tạo (`__init__`): Thiết lập các tham số quan trọng như `max_age` (thời gian tối đa một đối tượng có thể không được cập nhật trước khi bị loại bỏ), `min_hits` (số lần tối thiểu cần phát hiện để một đối tượng được coi là đang được theo dõi), và `iou_threshold` (ngưỡng IoU để liên kết các phát hiện với các theo dõi hiện có).

```
def __init__(self, max_age=1, min_hits=3, iou_threshold=0.3):
    """
    Sets key parameters for SORT
    """
    self.max_age = max_age
    self.min_hits = min_hits
    self.iou_threshold = iou_threshold
    self.trackers = []
    self.frame_count = 0
```

- Cập nhật (update): Phương thức này được gọi mỗi lần cho mỗi khung hình video, bao gồm cả khi không có phát hiện nào (sử dụng `np.empty((0, 5))`). Nó cập nhật vị trí dự đoán từ các theo dõi hiện có và cập nhật thông tin về các đối tượng đã được liên kết với các đối tượng mới. Nếu một đối tượng không được cập nhật trong một khoảng thời gian (`max_age`), nó sẽ bị loại bỏ.

```
def update(self, dets=np.empty((0, 5))):
    """
    Params:
        dets - a numpy array of detections in the format
        [[x1,y1,x2,y2,score],[x1,y1,x2,y2,score],...]
    Requires: this method must be called once for each frame even with
    empty detections (use np.empty((0, 5)) for frames without detections).
    Returns the a similar array, where the last column is the object ID.

    NOTE: The number of objects returned may differ from the number of
    detections provided.
    """
    self.frame_count += 1
    # get predicted locations from existing trackers.
    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)

    # update matched trackers with assigned detections
    for m in matched:
        self.trackers[m[1]].update(dets[m[0], :])

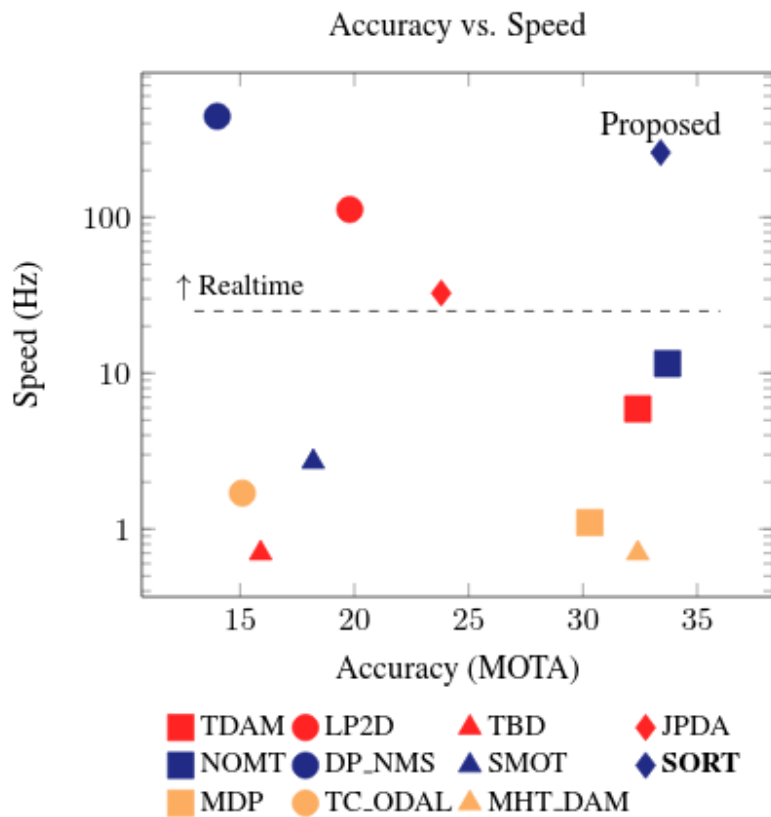
    # create and initialise new trackers for unmatched detections
    for i in unmatched_dets:
        trk = KalmanBoxTracker(dets[i,:])
        self.trackers.append(trk)
    i = len(self.trackers)
    for trk in reversed(self.trackers):
        d = trk.get_state()[0]
        if (trk.time_since_update < 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)) # +1
as MOT benchmark requires positive
            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))

```

3.4.5 Ưu điểm



Hình 22: Biểu đồ so sánh tốc độ và độ chính xác của các thuật toán theo dõi đối tượng.

Trên Hình 22, có thể quan sát rằng tốc độ và độ chính xác trong việc theo dõi đối tượng của thuật toán SORT vượt trội hơn hẳn so với các thuật toán khác. Sự khác biệt này rất rõ ràng, đặc biệt là khi chúng ta xem xét tốc độ cập nhật của SORT. Với thiết kế đơn giản nhưng hiệu quả, thuật toán SORT có khả năng theo dõi và cập nhật trạng thái của các đối tượng với tốc độ cao đáng kinh ngạc, nhanh hơn nhiều lần so với các thuật toán theo dõi đối tượng tiên tiến khác hiện có trên thị trường tại thời điểm đó. Điều này cho thấy SORT không chỉ mang lại độ chính xác cao mà còn cải

thiện đáng kể tốc độ xử lý, tạo ra một sự khác biệt lớn trong hiệu suất tổng thể so với các phương pháp lúc bấy giờ.

3.4.6 Nhược điểm:

Vấn đề lớn nhất của SORT hiện nay là việc quản lý ID Switches, tức là việc các đối tượng bị đổi ID một cách ngẫu nhiên khi chúng bị che khuất, có quỹ đạo trùng lặp, hoặc trong các tình huống khó xác định. SORT hiện chỉ sử dụng độ đo IoU để liên kết giữa nhận diện và theo dõi, mà không xem xét nhiều thông tin hơn về hình dạng hay các đặc trưng khác của đối tượng. Điều này dẫn đến số lượng ID Switches lớn cho một đối tượng khiến cho độ chính xác của thuật toán bị ảnh hưởng đáng kể.



Hình 23: Ví dụ minh họa về nhược điểm của SORT.

Ví dụ trên minh họa về nhược điểm của SORT khi đối tượng di chuyển chông chéo. Trong tình huống này, khả năng các đối tượng bị thay đổi ID là khá cao.

3.5 SFSORT

3.5.1 Giới thiệu:

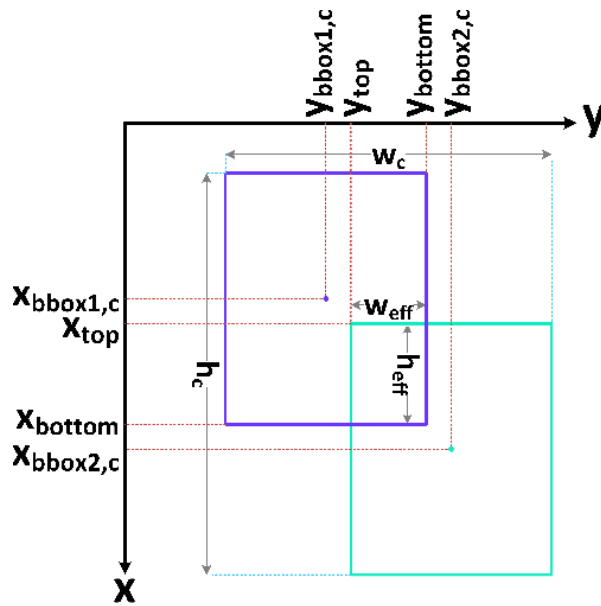
SFSORT [18] là một hệ thống theo dõi đa đối tượng dựa trên SORT được thiết kế để hoạt động với tính toán nhanh và độ chính xác cao trong thời gian thực. Được giới thiệu vào năm 2024, SFSORT giới thiệu những cải tiến đột phá trong việc liên kết các đối tượng qua các khung hình video mà không cần dự đoán vị trí tương lai. Thuật toán SFSORT sử dụng phương pháp theo dõi thời gian trực tuyến và giới thiệu

chỉ số tương đồng hộp giới hạn (BBSI) để loại bỏ sự phụ thuộc vào Kalman Filter, đồng thời giảm bớt yêu cầu tính toán so với các phương pháp trước đó.

3.5.2 Chỉ số tương đồng hộp giới hạn:

Chỉ số tương đồng hộp giới hạn là một công cụ đánh giá mới nhằm đo lường sự tương đồng giữa các hộp giới hạn. Chỉ số này được thiết kế để tính toán chi phí liên kết cho cả các khung giới hạn trùng nhau và không trùng nhau. BBSI xem xét ba yếu tố chính: sự tương đồng về hình dạng của các hộp giới hạn, khoảng cách giữa chúng, và diện tích giao nhau nếu chúng chồng chéo lên nhau. Bằng cách kết hợp các yếu tố này, BBSI cung cấp một chỉ số tổng quát về mức độ tương đồng giữa các hộp giới hạn, giúp cải thiện việc nhận diện và theo dõi đối tượng trong các ứng dụng thị giác máy tính.

Dưới đây là một số chi tiết được trực quan hóa để tính toán chỉ số BBSI:



Hình 24: Trực quan hóa các chi tiết tính chỉ số BBSI.

Công thức tính chỉ số BBSI:

- Bước 1: x_{bottom} được tính bằng giá trị nhỏ nhất giữa tọa độ x ở góc dưới phải của hai hộp giới hạn.

$$x_{bottom} = \min(x_{bbox1,rb}, x_{bbox2,rb}) \quad (15)$$

- Bước 2: x_{top} được tính bằng giá trị lớn nhất giữa tọa độ x ở góc trên trái của hai hộp giới hạn.

$$x_{top} = \max(x_{bbox1,tl}, x_{bbox2,tl}) \quad (16)$$

- Bước 3: y_{bottom} được tính bằng giá trị nhỏ nhất giữa tọa độ y ở góc dưới phải của hai hộp giới hạn.

$$y_{bottom} = \min(y_{bbox1,rb}, y_{bbox2,rb}) \quad (17)$$

- Bước 4: y_{top} được tính bằng giá trị lớn nhất giữa tọa độ y ở góc trên trái của hai hộp giới hạn.

$$y_{top} = \max(y_{bbox1,tl}, y_{bbox2,tl}) \quad (18)$$

- Bước 5: h_{eff} (chiều cao vùng giao nhau của hai hộp giới hạn) được tính bằng hiệu của x_{bottom} và x_{top} , nếu nhỏ hơn 0 thì lấy giá trị 0.

$$h_{eff} = \max(0, x_{bottom} - x_{top}) \quad (19)$$

- Bước 6: w_{eff} (chiều rộng vùng giao nhau của hai hộp giới hạn) được tính bằng hiệu của y_{bottom} và y_{top} , nếu nhỏ hơn 0 thì lấy giá trị 0.

$$w_{eff} = \max(0, y_{bottom} - y_{top}) \quad (20)$$

- Bước 7: Tính S_h . S_h là một thước đo sự tương đồng về chiều cao, nó tiến gần đến 1 đối với các hộp giới hạn tương đồng và tiến gần về 0 đối với các hộp giới hạn không tương đồng.

$$S_h = \frac{h_{eff}}{h_{eff} + |h_{bbox2} - h_{bbox1}| + \epsilon} \quad (21)$$

- Bước 8: Tính S_w . S_w là một thước đo sự tương đồng về chiều rộng, nó tiến gần đến 1 đối với các hộp giới hạn tương đồng và tiến gần về 0 đối với các hộp giới hạn không tương đồng.

$$S_w = \frac{w_{eff}}{w_{eff} + |w_{bbox2} - w_{bbox1}| + \epsilon} \quad (22)$$

- Bước 9: Tính S_c . S_c biểu thị sự phù hợp của các tâm hộp giới hạn.

$$S_c = \frac{|x_{bbox1,c} - x_{bbox2,c}| + |y_{bbox1,c} - y_{bbox2,c}|}{h_c + w_c} \quad (23)$$

- Bước 10: Tính ADIoU.

$$ADIoU = IoU - S_c \quad (24)$$

- Bước 11: Tính BBSI.

$$BBSI = ADIoU + S_h + S_w \quad (25)$$

3.5.3 Các thành phần chính của SFSORT:

SFSORT bao gồm ba thành phần chính:

- Module liên kết thứ nhất: Liên kết các phát hiện có độ tin cậy cao.
- Module liên kết thứ hai: Liên kết các phát hiện có độ tin cậy trung bình.
- Module quản lý theo vết đối tượng: Quản lý trạng thái của các dấu vết đối tượng.

3.5.3.1 Module liên kết thứ nhất:

Module liên kết thứ nhất tạo ra một bộ theo dõi chứa các dấu vết đã mất và còn hoạt động từ khung hình T-1. Sau đó, dựa trên chỉ số tương đồng hộp giới hạn, nó so sánh các hộp giới hạn của các dấu vết trong bộ theo dõi với các phát hiện có điểm số cao từ bộ phát hiện đối tượng. Tiếp theo, nó gán ID cho các phát hiện khớp với các dấu vết, đảm bảo rằng ID của mỗi đối tượng từ khung hình T giống với dấu vết tương ứng của nó từ khung hình T-1. Các dấu vết đã khớp sau đó được gửi đến module quản lý theo vết đối tượng để cập nhật trạng thái. Các phát hiện có điểm số đủ cao nhưng không khớp với dấu vết hiện có được coi là dấu vết mới và được chuyển tiếp đến module quản lý theo vết đối tượng để khởi tạo.

Hàm chi phí trong mô đun liên kết thứ nhất:

$$cost_{1st.association} = 1 - \frac{BBSI}{3} \quad (26)$$

3.5.3.2 Module liên kết thứ hai:

Module liên kết thứ hai có nhiệm vụ liên kết các dấu vết không khớp từ liên kết đầu tiên với các phát hiện có điểm số trung bình. Sự giảm điểm số phát hiện thường xảy ra do sự che khuất một phần của một số bộ phận đối tượng, dẫn đến sự thay đổi kích thước của hộp giới hạn phát hiện. Do đó, bỏ qua sự tương đồng về kích thước của hộp giới hạn, module liên kết thứ hai chỉ dựa vào chỉ số IoU. Các khớp thành công trong module liên kết thứ hai sau đó được chuyển tiếp đến module quản lý theo vết đối tượng để cập nhật trạng thái.

Hàm chi phí trong mô đun liên kết thứ hai:

$$cost_{2nd.association} = 1 - IoU \quad (27)$$

3.5.3.3 Module quản lý theo vết đối tượng:

Sau quá trình liên kết, module quản lý theo vết đối tượng xử lý các dấu vết đã khớp, các dấu vết mới và các dấu vết không khớp. Đối với các dấu vết đã khớp, module quản lý theo vết đối tượng cập nhật trạng thái của dấu vết thành đang hoạt động, ghi lại tọa độ hộp giới hạn và số khung hình chứa dấu vết đó. Đối với các dấu vết không khớp, module quản lý theo vết đối tượng sẽ kiểm tra tọa độ hộp giới hạn cuối cùng được ghi lại và tùy thuộc vào vị trí dấu vết bị mất, nó sẽ thay đổi trạng thái của dấu vết thành mất ở trung tâm hoặc mất ở biên. Hệ thống theo dõi giữ lại các dấu vết đã mất trong dự đoán về khả năng quay lại của chúng. Tuy nhiên, module quản lý theo vết đối tượng sẽ loại bỏ các dấu vết đã mất trong khoảng thời gian vượt quá giới hạn thời gian quy định khỏi danh sách các dấu vết đủ điều kiện để liên kết đối tượng - dấu vết. Đáng chú ý, giới hạn thời gian cho các dấu vết mất ở biên ngắn hơn so với các dấu vết mất ở trung tâm. Sự phân biệt này xuất phát từ việc hiểu rằng khi một dấu vết bị mất ở rìa của khung hình, có nhiều khả năng là do đối tượng di chuyển ra khỏi tầm nhìn của camera.

3.5.3.4 Các siêu tham số chính của SFSORT:

Các siêu tham số chính của SFSORT được định nghĩa như bảng sau:

Ký hiệu	Mô tả
HTH	Điểm số tối thiểu cho các phát hiện có điểm số cao
LTH	Điểm số tối thiểu cho các phát hiện có điểm số trung bình
MTH1	Chi phí tối đa cho phép trong mô đun liên kết đầu tiên
MTH2	Chi phí tối đa cho phép trong mô đun liên kết thứ hai
NTH	Điểm số tối thiểu cho các phát hiện được xác định là các đối tượng mới
HMargin	Biên để xác định ranh giới ngang của các khu vực trung tâm
VMargin	Biên để xác định ranh giới dọc của các khu vực trung tâm
Ctime	Thời gian chờ cho các đối tượng bị mất dấu ở các khu vực trung tâm
Mtime	Thời gian chờ cho các đối tượng bị mất dấu ở các khu vực biên

Bảng 9: Các siêu tham số chính của SFSORT.

Số lượng đối tượng trong mỗi khung hình video ảnh hưởng đến HTH, NTH và MTH1. Khi khung hình trở nên đông đúc hơn, điểm tin cậy phát hiện thường giảm do che khuất. Do đó, cả HTH và MTH1 cần được giảm để đạt được bộ theo dõi thích ứng. Hơn nữa, khi khung hình trở nên đông đúc hơn, số lượng các dấu vết bị mất thường tăng do che khuất. Do đó, NTH cần được tăng để ngăn chặn việc chuyển đổi ID. SFSORT sử dụng mối quan hệ tuyến tính giữa logarit của số lượng đối tượng trong một khung hình và các siêu tham số HTH, NTH và MTH1.

$$count = \log_{10}(|\{x|score(x) > CTH\}|) \quad (28)$$

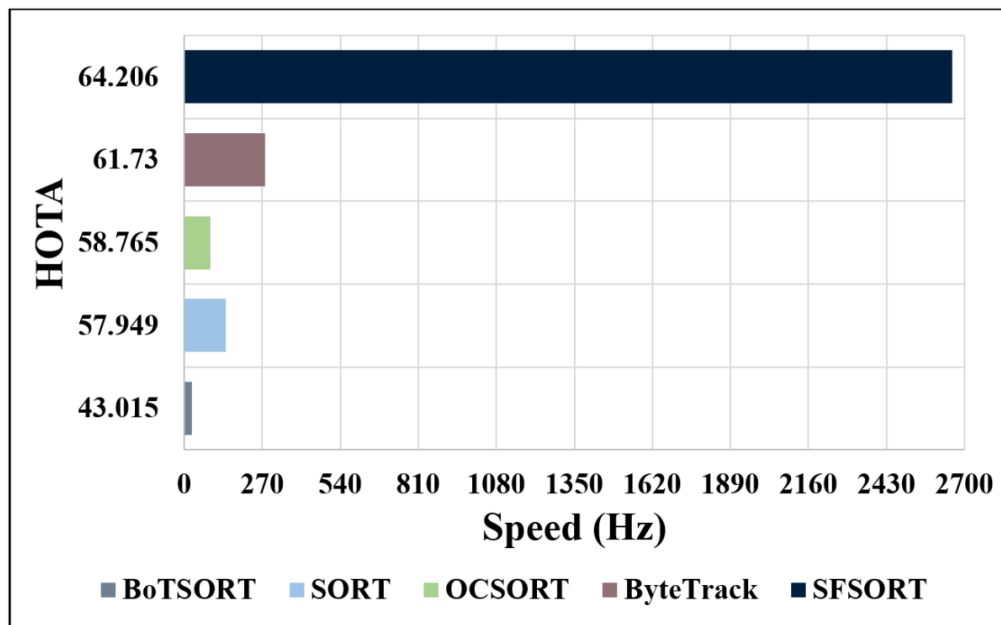
$$HTH = HTH_0 - (HTH_m * count) \quad (29)$$

$$NTH = NTH_0 + (NTH_m * count) \quad (30)$$

$$MTH1 = MTH_0 - (MTH_m * count) \quad (31)$$

Số lượng đối tượng trong khung hình được xác định bởi số lượng phát hiện trong tập hợp có điểm phát hiện vượt quá một ngưỡng nhất định. Điều này giới thiệu ngưỡng, ký hiệu là CTH, như một siêu tham số nhỏ trong vấn đề theo dõi.

3.5.4 Ưu điểm của SFSORT:



Hình 25: So sánh độ chính xác và tốc độ của các thuật toán theo vết đối tượng dựa trên SORT.

Quan sát từ hình 25 có thể thấy SFSORT có ưu điểm vượt trội về tốc độ xử lý, được biết đến là một trong những thuật toán theo vết đối tượng nhanh nhất thế giới. Khác với thuật toán SORT truyền thống, SFSORT giảm thiểu hiện tượng ID switch bằng cách không chỉ dựa vào chỉ số IoU mà còn áp dụng chỉ số BBSI để tối ưu hóa quá trình liên kết giữa các nhận diện và theo dõi. Điều này giúp cải thiện độ chính xác và hiệu suất của thuật toán trong các tình huống phức tạp hơn, nơi mà các đối tượng có thể chồng lấn hoặc có quỹ đạo di chuyển gần nhau.



Hình 26: Ví dụ minh họa ưu điểm của SFSORT.

Ví dụ trên minh họa ưu điểm của thuật toán SFSORT khi các đối tượng di chuyển chòng chéo. Thuật toán này giúp tối ưu hóa và làm cho việc quản lý ID trở nên ổn định hơn.

Chương 4. Giải pháp đề xuất

4.1 Phân tích dữ liệu

Tập dữ liệu được xây dựng bằng cách chuyển đổi video thu thập từ UAV thành các khung hình riêng lẻ và sau đó gán nhãn thủ công cho từng khung hình.

- Số lượng mẫu:
 - Tổng số lượng ảnh: 540.
 - Tổng số lượng tập nhãn tương ứng: 540.
- Kích thước hình ảnh:
 - Kích thước mỗi ảnh: 1280 x 720 pixel.
- Số lượng đối tượng: 2
 - car: 3110.
 - truck: 741.
- Kích thước tệp dữ liệu:
 - Tổng kích thước của 540 ảnh: 144MB.
 - Tổng kích thước của 540 tập nhãn: 154KB.
- Ví dụ: Dưới đây là minh họa về hình ảnh và tập nhãn tương ứng.



Hình 27: Ví dụ về hình ảnh trong tập dữ liệu.

```
0 0.378125 0.71328125 0.14609375 0.18203125
0 0.5734375 0.578125 0.10234375 0.14453125
0 0.103125 0.88125 0.16796875 0.209375
0 0.87421875 0.14765625 0.0859375 0.159375
0 0.1625 0.7046875 0.1421875 0.17890625
1 0.75390625 0.3140625 0.1578125 0.2859375
```

Hình 28: Ví dụ về nhãn trong tập dữ liệu.

Cấu trúc tổng quát của một tập tin nhãn YOLOv8:

`<class> <x_center> <y_center> <width> <height>`

Trong đó:

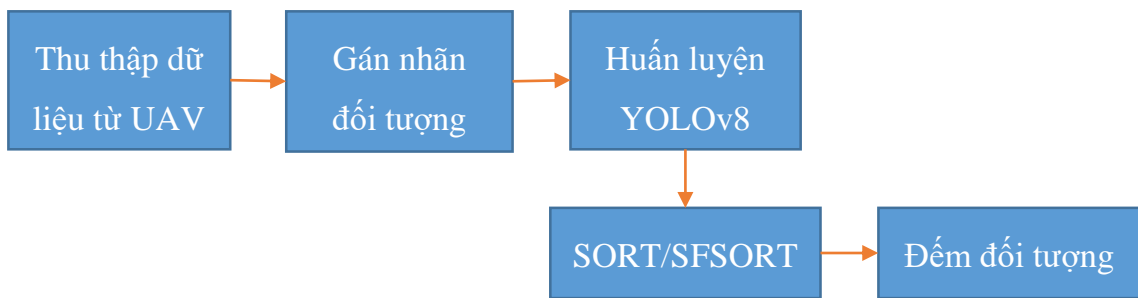
- `<class>`: Nhãn lớp của đối tượng.
- `<x_center>`: Tọa độ x được chuẩn hóa của tâm hộp giới hạn.
- `<y_center>`: Tọa độ y được chuẩn hóa của tâm hộp giới hạn.
- `<width>`: Chiều rộng được chuẩn hóa của hộp giới hạn.
- `<height>`: Chiều cao được chuẩn hóa của hộp giới hạn.

- Đánh giá:

Tập dữ liệu trên khá tốt với việc chuyển đổi video từ UAV thành 540 khung hình chất lượng cao, mỗi ảnh có kích thước 1280 x 720 pixel, cho phép mô hình học được nhiều chi tiết tinh vi. Số lượng nhãn cũng rất đáng kể với 3110 car (xe hơi) và 741 truck (xe tải), điều này cung cấp một lượng dữ liệu phong phú để đào tạo mô hình. Tổng kích thước của ảnh là 144MB và nhãn chỉ 154KB, cho thấy sự tổ chức dữ liệu hiệu quả và dễ quản lý. Dù số lượng mẫu có thể là thách thức đối với các dự án lớn hơn, tập dữ liệu này đã được xây dựng cẩn thận với chất lượng nhãn cao và sự chú ý đến chi tiết, là nền tảng vững chắc cho các nghiên cứu và phát triển trong lĩnh vực theo dõi đối tượng.

4.2 Mô hình

Giải pháp đề xuất cho bài toán “Đếm số đối tượng từ máy bay không người lái”:



Để giải quyết bài toán đếm số đối tượng từ máy bay không người lái, khóa luận đề xuất một quy trình gồm các bước sau:

- Thu thập dữ liệu từ UAV:

Sử dụng UAV để quay video và chụp ảnh, thu thập dữ liệu hình ảnh chứa các đối tượng cần đếm trong các môi trường và điều kiện khác nhau.

- Gán nhãn đối tượng:

Tiến hành gán nhãn bằng makesense.ai để xác định và đánh dấu các đối tượng trong các hình ảnh thu thập được. Các nhãn này sẽ chứa thông tin về vị trí và loại đối tượng, chuẩn bị cho việc huấn luyện mô hình.

- Huấn luyện mô hình YOLOv8 trên tập dữ liệu mới:

Sử dụng tập dữ liệu đã gán nhãn để huấn luyện mô hình YOLOv8, một trong những mô hình tiên tiến trong việc nhận diện và phát hiện đối tượng. Mô hình này sẽ được tối ưu hóa để có thể nhận diện chính xác các đối tượng từ dữ liệu UAV.

- Kết hợp YOLOv8 với SORT/SFSORT để theo vết đối tượng và gán ID cho đối tượng:

Sau khi huấn luyện xong mô hình YOLOv8, các phát hiện đối tượng thu được từ mô hình sẽ được kết hợp với thuật toán SORT và SFSORT để theo dõi các đối tượng trong video.

- Đếm đối tượng đi qua ROI:

Xác định vùng quan tâm trong video nơi cần đếm số lượng đối tượng đi qua. Sử dụng thông tin từ việc theo dõi đối tượng để đếm số lượng đối tượng đi qua ROI

Kiến trúc YOLOv8 được chia thành ba phần chính:

- Backbone: Backbone là phần trích xuất đặc trưng của mô hình. Nó chịu trách nhiệm trích xuất các đặc trưng từ hình ảnh đầu vào. YOLOv8 sử dụng CSPDarknet làm backbone. Backbone của YOLOv8 bao gồm 5 giai đoạn. Mỗi giai đoạn bao gồm một số khối CSPDarknet. Khối CSPDarknet là một khối mạng nơ-ron được thiết kế để trích xuất các đặc trưng từ hình ảnh một cách hiệu quả. Khối CSPDarknet bao gồm hai phần:
 - C2f: C2f là phần trích xuất đặc trưng chính của khối CSPDarknet. Nó giúp cải thiện khả năng học tập của mô hình. Cấu trúc của C2f:
 - Convolutional Layer đầu vào.
 - Split Layer: Đầu vào từ lớp convolutional đầu tiên được chia thành hai nhánh.
 - Nhánh 1: Đi qua n lớp Bottleneck, mỗi lớp có thể có hoặc không có kết nối tắt.
 - Concatenate Layer: Kết hợp đầu ra của các lớp Bottleneck với nhánh thứ hai của Split Layer.
 - Convolutional Layer cuối cùng: để kết hợp các đặc trưng đã hợp nhất.
 - DownSample: DownSample là phần giảm kích thước của đặc trưng. Nó được sử dụng để giảm kích thước của đặc trưng sau khi trích xuất bởi C2f, nhằm giữ được thông tin quan trọng trong các kích thước nhỏ hơn.
- Neck: Neck là phần kết nối backbone với head. Nó chịu trách nhiệm kết hợp các đặc trưng từ các giai đoạn khác nhau của backbone để cải thiện hiệu suất phát hiện đối tượng. YOLOv8 sử dụng Path Aggregation Network (PANet) làm neck. PANet trong YOLOv8 được thiết kế để giải quyết vấn đề của các mô hình phát hiện đối tượng khi phải xử lý các đối tượng có kích thước khác nhau trong hình ảnh bằng cách sử dụng các kết nối ngược và các kết nối bổ sung để kết hợp thông tin từ các lớp khác nhau. Bằng cách kết hợp thông tin

từ các tầng khác nhau, PANet giúp cải thiện chất lượng và độ chính xác của dự đoán. PANet gồm:

- Bottom-up: Đây là các đường đi từ các lớp thấp đến cao trong mạng nơ-ron, giúp chiết xuất các đặc trưng ở các tỉ lệ thấp đến cao.
 - Top-down: Ngược lại với bottom-up, các đường đi này từ các lớp cao xuống thấp để tổng hợp và kết nối các thông tin từ các tỉ lệ khác nhau của hình ảnh.
- Head: Head là phần dự đoán các đối tượng trong hình ảnh. Nó bao gồm nhiều khối mạng nơ-ron, thực hiện các nhiệm vụ như:
- Dự đoán các hộp giới hạn cho các đối tượng.
 - Dự đoán độ tin cậy cho mỗi hộp giới hạn, thể hiện mức độ chắc chắn rằng hộp giới hạn đó chứa đối tượng.
 - Phân loại đối tượng, xác định lớp đối tượng được hiển thị trong hộp giới hạn.

Hai thành phần quan trọng trong head là ClsLoss và BboxLoss, có vai trò tính toán tổn thất cho nhiệm vụ phân loại và dự đoán hộp giới hạn, góp phần điều chỉnh mô hình trong quá trình huấn luyện.

- Trong lớp đầu ra của YOLOv8, họ sử dụng hàm sigmoid làm hàm kích hoạt cho điểm số độ chính xác của đối tượng, đại diện cho xác suất mà hộp giới hạn chứa một đối tượng. Mô hình sử dụng hàm softmax cho xác suất lớp, đại diện cho xác suất các đối tượng thuộc về từng lớp có thể có.
- YOLOv8 sử dụng các hàm mất mát CIoU và DFL cho mất mát hộp giới hạn và hàm mất mát binary cross-entropy cho phân loại.
- Huấn luyện YOLOv8:
 - Cài đặt và import các thư viện cần thiết:

```
!pip install ultralytics
import cv2
import numpy as np
from ultralytics import YOLO
```

- Khởi tạo mô hình YOLOv8: Chúng ta sẽ khởi tạo mô hình YOLOv8 với phiên bản nano(n) từ trọng số đã được huấn luyện trên bộ dữ liệu COCO.

```
model=YOLO ("yolov8n.pt")
```

- Huấn luyện mô hình:

```
model.train(data = "/content/drive/MyDrive/Colab  
Notebooks/data.yaml", epochs=50, imgsz=640)
```

Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size			
1/50	2.67G	0.625	1.747	0.88	114	640: 100%	34/34	[00:24<00:00, 1.41it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	17/17	[00:14<00:00, 1.16it/s]
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size			
2/50	2.2G	0.4993	0.6974	0.8441	142	640: 100%	34/34	[00:19<00:00, 1.78it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	17/17	[00:12<00:00, 1.34it/s]
	all	540	3851	0.945	0.879	0.97	0.889		
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size			
3/50	2.2G	0.4794	0.5949	0.8408	154	640: 100%	34/34	[00:17<00:00, 1.99it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	17/17	[00:10<00:00, 1.60it/s]
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size			
4/50	2.2G	0.4571	0.5696	0.8356	166	640: 100%	34/34	[00:19<00:00, 1.74it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	17/17	[00:14<00:00, 1.18it/s]
	all	540	3851	0.964	0.967	0.983	0.932		
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size			
5/50	2.22G	0.4229	0.5123	0.8285	137	640: 100%	34/34	[00:19<00:00, 1.72it/s]	
	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100%	17/17	[00:14<00:00, 1.16it/s]
	all	540	3851	0.967	0.968	0.987	0.954		

Hình 30: Quá trình huấn luyện mô hình.

Hình trên minh họa cho quá trình huấn luyện mô hình YOLOv8 trên tập dữ liệu mới.

4.2.2 Kết hợp YOLOv8 với SORT

- Chuẩn bị đầu vào:

Nhận dòng video hoặc một chuỗi các khung hình.

- Khởi tạo:

Tải mô hình YOLOv8 đã được huấn luyện.

Tạo đối tượng SORT để quản lý thông tin theo dõi.

- Vòng lặp qua từng khung hình:

Bước 1: Phát hiện đối tượng với YOLOv8:

- Sử dụng mô hình YOLOv8 để phát hiện các đối tượng trong khung hình hiện tại.

Bước 2: Tiền xử lý các phát hiện:

- Định dạng lại các phát hiện để phù hợp với đầu vào của SORT (bao gồm tọa độ của hộp giới hạn và độ tin cậy).

Bước 3: Cập nhật theo dõi với SORT:

- Đưa các phát hiện đã định dạng vào SORT.
- SORT sử dụng Kalman Filter để dự đoán vị trí tiếp theo của các đối tượng và giải thuật Hungary để ghép nối các phát hiện mới với các đối tượng đang được theo dõi.

Bước 4: Lấy thông tin đối tượng đã theo dõi:

- SORT trả về danh sách các đối tượng đang được theo dõi cùng với các ID duy nhất cho mỗi đối tượng.
- Cập nhật thông tin vị trí và ID cho mỗi đối tượng trong khung hình hiện tại.

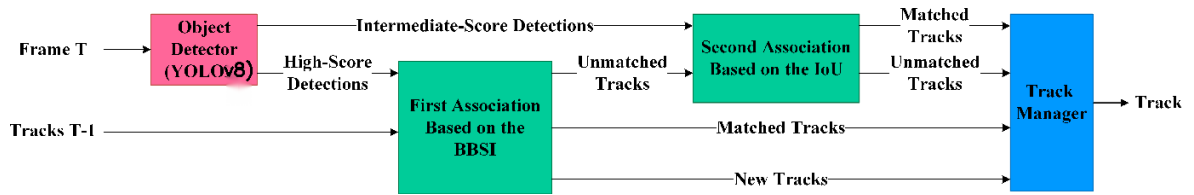
Bước 5: Hiển thị hoặc lưu trữ kết quả:

- Vẽ các hộp giới hạn và ID của đối tượng lên khung hình.
- Lưu khung hình đã xử lý hoặc hiển thị nó trên màn hình.

Lặp lại quá trình trên cho từng khung hình trong dòng video. [19]

4.2.3 Kết hợp YOLOv8 với SFSORT

Cách kết hợp YOLOv8 với SFSORT được mô tả như hình sau:



Hình 31: Kết hợp YOLOv8 với SFSORT.

Trước hết YOLOv8 được dùng để phát hiện các đối tượng trong từng khung hình video. Sau đó, SFSORT nhận các hộp giới hạn đối tượng và điểm phát hiện từ bộ phát hiện đối tượng YOLOv8. Sau khi xử lý từng khung hình video, thuật toán cập nhật hai danh sách: Các dấu vết đang hoạt động và các dấu vết bị mất. Thuật toán loại bỏ các dấu vết bị mất đã vượt quá thời gian cho phép từ danh sách các dấu vết bị mất. Các dấu vết bị mất và các dấu vết đang hoạt động được kết hợp để tạo thành một tập hợp theo dõi. Khi tập hợp theo dõi trống, thường là trường hợp lúc bắt đầu theo dõi đối tượng, thuật toán xác định tất cả các phát hiện có điểm cao là các dấu vết mới, khởi tạo chúng và thêm vào danh sách các dấu vết đang hoạt động. Thuật toán phân chia các phát hiện thành hai loại: Phát hiện chắc chắn và phát hiện chưa chắc chắn, dựa trên điểm phát hiện của chúng.

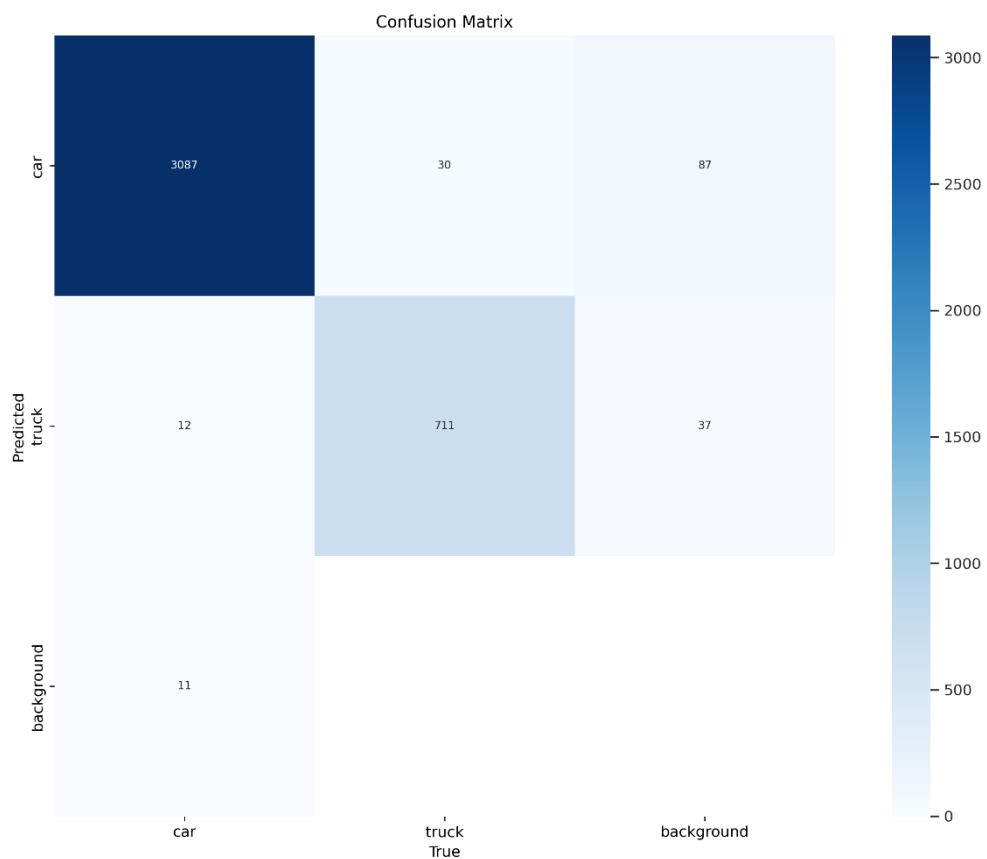
Module liên kết đối tượng-dấu vết đầu tiên xảy ra liên quan đến các phát hiện chắc chắn và các dấu vết từ tập hợp theo dõi. Ở module này chỉ số BBSI được sử dụng làm thước đo liên kết. Trong sự liên kết đầu tiên này, thuật toán cập nhật trạng thái của các dấu vết được khớp thành trạng thái hoạt động, và ghi nhận hộp giới hạn của chúng cùng với số khung hiện tại. Ngoài ra, thuật toán loại bỏ các dấu vết đã khớp có mặt trong danh sách các dấu vết bị mất ra khỏi danh sách này. Các phát hiện có điểm cao nhưng không khớp được xác định là các dấu vết mới. Thuật toán thêm tất cả các dấu vết được liên kết trong module liên kết đầu tiên vào danh sách các dấu vết đang hoạt động.

Module liên kết thứ hai liên kết các phát hiện chưa chắc chắn và các dấu vết không được khớp từ module liên kết đầu tiên. Ở module này chỉ số IoU được sử dụng làm thước đo liên kết. Các dấu vết được liên kết trong module liên kết thứ hai được kích hoạt và thêm vào danh sách các dấu vết đang hoạt động. Các dấu vết được khớp,

nếu có mặt trong danh sách các dấu vết bị mất, sẽ được loại bỏ khỏi danh sách này. Sau sự cố gắng liên kết thứ hai, thuật toán thêm các dấu vết còn lại không được khớp vào danh sách các dấu vết bị mất. Đối với các dấu vết bị mất, thuật toán cập nhật trạng thái của chúng, ghi nhận hộp giới hạn được quan sát lần cuối và ghi lại số khung hình gần nhất của dấu vết. Trạng thái được xác định dựa trên nơi dấu vết bị mất, gồm trạng thái bị mất tại trung tâm hoặc bị mất tại biên. Các dấu vết đang hoạt động được thêm vào danh sách Tracks, một tập hợp các dấu vết từ tất cả các khung hình video. Cuối cùng, thuật toán trả về danh sách Tracks.

4.3 Thử nghiệm

- YOLOv8:
- + Ma trận nhầm lẫn: Hình dưới là ma trận nhầm lẫn thu được khi huấn luyện mô hình YOLOv8 trên tập dữ liệu mới.



Hình 32: Ma trận nhầm lẫn của mô hình YOLOv8.

+ Đánh giá:

- Dự đoán car khá chính xác với 3087 lần đúng và chỉ có 117 lần nhầm lẫn (30 lần với truck và 87 lần với background).
- Dự đoán truck cũng khá chính xác với 711 lần đúng và chỉ có 49 lần nhầm lẫn (12 lần với car và 37 lần với background).
- Mô hình hoạt động khá tốt trong việc phân loại các đối tượng, nhưng vẫn có một số nhầm lẫn cần được cải thiện.

+ Kết quả huấn luyện: Hình sau mô tả kết quả huấn luyện của mô hình YOLOv8 trên tập dữ liệu mới.

Class	Images	Instances	Box(P	R	mAP50	mAP50-95):
all	540	3851	0.978	0.98	0.993	0.987
car	540	3110	0.98	0.992	0.994	0.986
truck	492	741	0.976	0.969	0.992	0.987

Hình 33: Kết quả huấn luyện mô hình YOLOv8.

+ Đánh giá:

- Độ chính xác và độ nhạy đều rất cao cho cả hai lớp "car" và "truck". Điều này cho thấy mô hình YOLOv8 hoạt động rất tốt trong việc nhận diện và phát hiện các đối tượng trong tập dữ liệu.
- mAP tại 50% IoU (mAP50) cho tất cả các lớp là 0.993, rất gần với 1.0, cho thấy mô hình có khả năng phân loại chính xác cao.
- mAP50-95, một chỉ số khắt khe hơn vì nó xem xét một dải các ngưỡng IoU từ 50% đến 95%, cũng rất cao (0.987 cho tất cả các lớp), cho thấy mô hình duy trì hiệu suất tốt trên nhiều ngưỡng IoU khác nhau.
- Mô hình được huấn luyện trên 540 ảnh và tổng cộng 3851 instance cho tất cả các lớp, điều này cho thấy tập dữ liệu đủ lớn để mô hình học và tổng quát hóa tốt.
- Riêng lớp "truck" có số lượng ảnh thấp hơn (492) nhưng vẫn đạt được kết quả rất tốt, cho thấy khả năng học của mô hình mạnh mẽ ngay cả với số lượng dữ liệu ít hơn.

- Kết quả thực nghiệm theo vết đối tượng và đếm đối tượng:

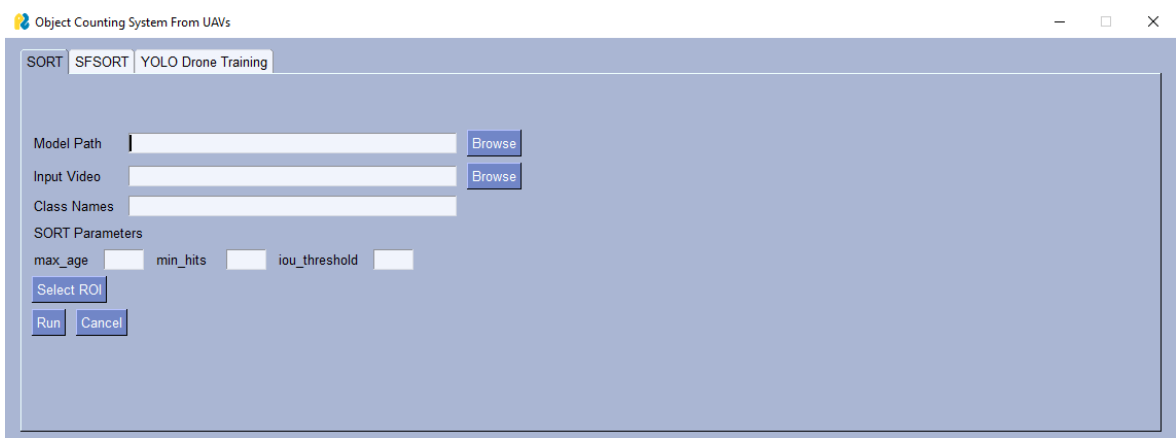
Giả sử các phương pháp theo vết đối tượng SORT và SFSORT được sử dụng trên cùng một video và cùng ROI.

Phương pháp	Số ID được gán	Số lượng đếm			Tốc độ theo vết (Hz)
		car	truck	Tổng	
SORT	49	16	2	18	207.77
SFSORT	30	16	2	18	596.50

Bảng 10: Kết quả thực nghiệm theo vết đối tượng và đếm đối tượng của các phương pháp.

Dựa trên bảng kết quả thực nghiệm, có thể thấy rằng việc quản lý ID của SFSORT ổn định hơn so với SORT và tốc độ theo dõi cũng rất nhanh.

- Thiết kế giao diện ứng dụng:
- + Giao diện đếm đối tượng kết hợp YOLOv8 và SORT:



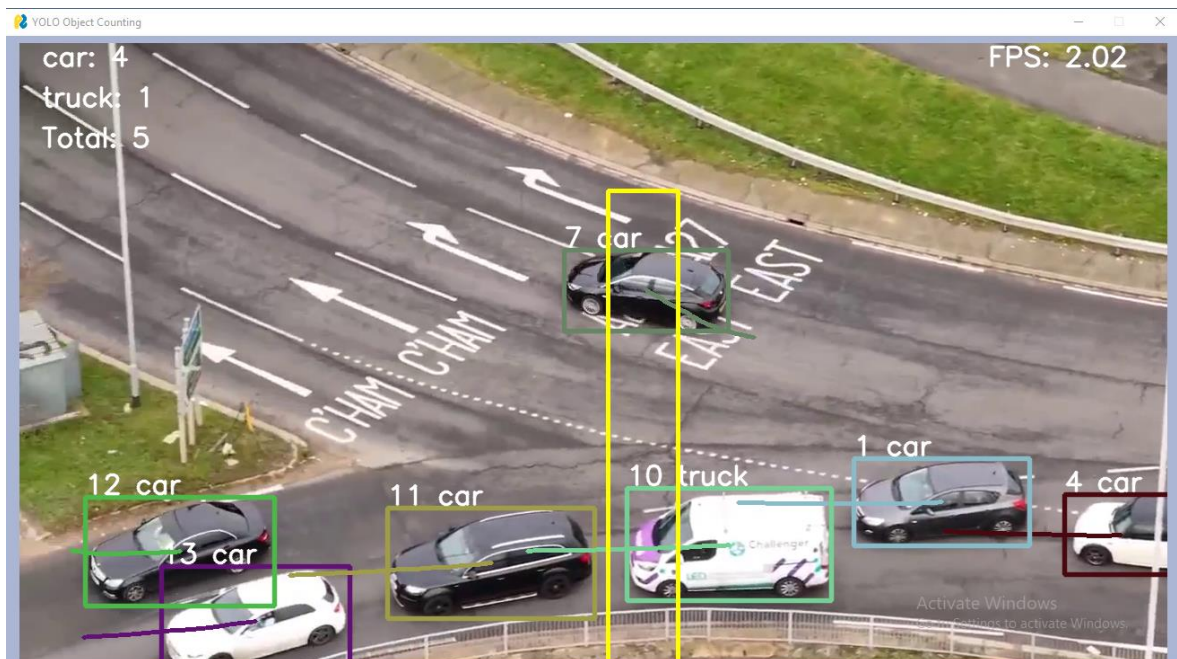
Hình 34: Giao diện đếm đối tượng kết hợp YOLOv8 và SORT.

- Chọn đường dẫn đến mô hình YOLOv8 đã được huấn luyện.
- Chọn đường dẫn đến video cần đếm đối tượng.
- Nhập tên của các lớp đối tượng, ngăn cách nhau bởi dấu “,”.
- Nhập các tham số của SORT.
- Nhấn vào nút “Select ROI” sẽ hiện ra khung hình đầu tiên của video để người dùng vẽ ROI.



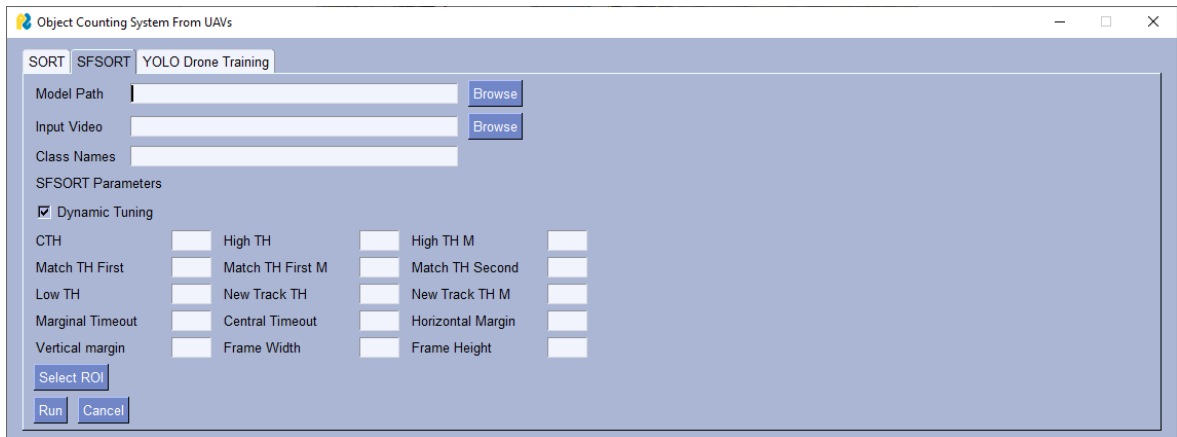
Hình 35: Giao diện vẽ ROI khi kết hợp YOLO với SORT.

- Nhấn vào nút “Run” sẽ hiển thị kết quả đếm cùng với FPS.



Hình 36: Giao diện kết quả khi kết hợp YOLO với SORT.

- + Giao diện đếm đối tượng kết hợp YOLOv8 và SFSORT:



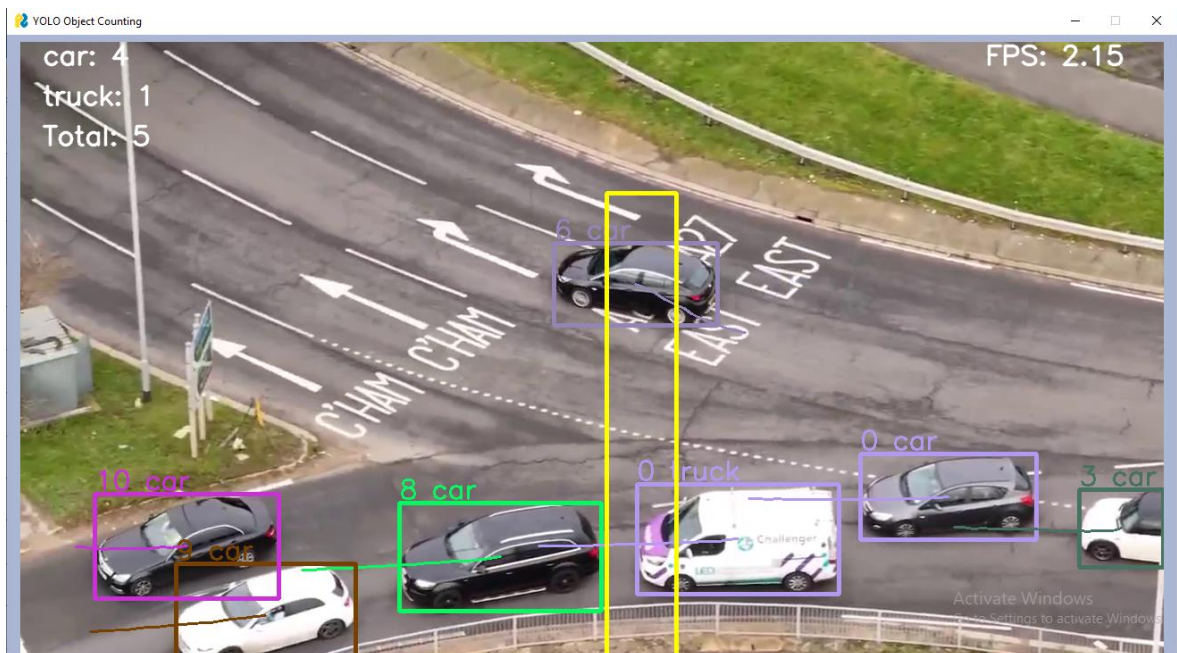
Hình 37: Giao diện đếm đối tượng kết hợp YOLOv8 và SFSORT.

- Chọn đường dẫn đến mô hình YOLOv8 đã được huấn luyện.
- Chọn đường dẫn đến video cần đếm đối tượng.
- Nhập tên của các lớp đối tượng, ngăn cách nhau bởi dấu “,”.
- Nhập các tham số của SFSORT.
- Nhấn vào nút “Select ROI” sẽ hiện ra khung hình đầu tiên của video để người dùng vẽ ROI.



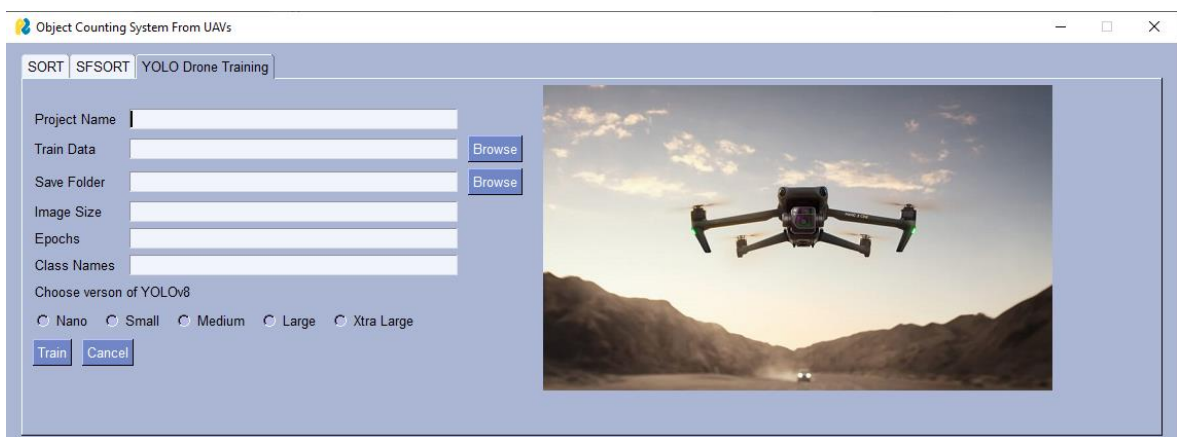
Hình 38: Giao diện vẽ ROI khi kết hợp YOLO với SFSORT.

- Nhấn vào nút “Run” sẽ hiển thị kết quả đếm cùng với FPS.



Hình 39: Giao diện kết quả khi kết hợp YOLO với SFSORT.

+ Giao diện huấn luyện mô hình YOLOv8:



Hình 40: Giao diện huấn luyện mô hình YOLOv8.

- Nhập tên dự án huấn luyện.
- Chọn đường dẫn đến tập dữ liệu dùng để huấn luyện mô hình.
- Chọn đường dẫn đến thư mục lưu kết quả huấn luyện mô hình.
- Nhập kích thước ảnh huấn luyện mong muốn.
- Nhập số lượng epochs huấn luyện mong muốn.
- Nhập tên của các lớp đối tượng, ngăn cách nhau bởi dấu “,”.

- Chọn loại YOLOv8 muốn huấn luyện. Gồm: Nano (n), Small (s), Medium (m), Large (l), Xtra Large (x).
- Nhấn vào nút “Train” để bắt đầu quá trình huấn luyện mô hình.

KẾT LUẬN

Những điều đã làm được

Khóa luận này đã đóng góp trong việc ứng dụng các phương pháp học sâu và thuật toán theo dõi đối tượng để giải quyết bài toán đếm số lượng đối tượng từ video ghi lại bằng máy bay không người lái. Cụ thể, chúng tôi đã thành công trong việc huấn luyện mô hình YOLOv8, một trong những mô hình nhận diện đối tượng tiên tiến và hiệu quả nhất hiện nay. Mô hình này đã được tối ưu hóa để nhận diện và đếm số lượng đối tượng trong từng khung hình của video với độ chính xác cao và khả năng xử lý nhanh chóng trong thời gian thực.

Ngoài ra, chúng tôi đã tích hợp thành công YOLOv8 với các thuật toán theo dõi đối tượng như SORT và SFSORT. SORT và SFSORT là những thuật toán mạnh mẽ trong việc theo dõi các đối tượng qua các khung hình liên tiếp, giúp loại bỏ sự trùng lặp trong việc đếm và cải thiện độ chính xác của kết quả đếm. Sự kết hợp này không chỉ nâng cao tính chính xác trong việc đếm đối tượng mà còn mang lại tính linh hoạt và hiệu quả cao trong việc ứng dụng vào các tình huống thực tiễn như giám sát giao thông, quản lý an ninh và nghiên cứu môi trường.

Kết quả thực nghiệm đã cho thấy phương pháp đề xuất có khả năng đếm đối tượng một cách tương đối chính xác, đáp ứng được các yêu cầu của bài toán và mở ra những triển vọng mới trong việc áp dụng các công nghệ hiện đại vào thực tế. Điều này khẳng định giá trị của nghiên cứu và đóng góp quan trọng vào lĩnh vực thị giác máy tính và học sâu.

Những điều chưa làm được

Mặc dù đã đóng góp đáng kể, khóa luận này vẫn còn một số hạn chế cần khắc phục để nâng cao hiệu quả và độ chính xác của hệ thống. Đầu tiên, khả năng nhận

diện của mô hình YOLOv8 vẫn có thể được cải thiện hơn nữa. Mặc dù đã huấn luyện và tối ưu hóa mô hình, nhưng trong một số tình huống phức tạp, như khi đối tượng có kích thước nhỏ hoặc khi có sự che khuất giữa các đối tượng, độ chính xác nhận diện vẫn chưa đạt mức tối ưu. Việc tinh chỉnh các siêu tham số của mô hình và thử nghiệm với tập dữ liệu phong phú hơn có thể giúp nâng cao khả năng nhận diện và giảm thiểu lỗi trong các tình huống khó khăn này.

Thêm vào đó, thuật toán SFSORT, mặc dù đã được tích hợp và sử dụng trong nghiên cứu, vẫn còn một số điểm cần cải thiện. Các tham số của thuật toán SFSORT cần được tinh chỉnh thêm để tối ưu hóa hiệu quả theo vết đối tượng. Thực hiện các thử nghiệm với các giá trị tham số khác nhau và đánh giá tác động của chúng đến hiệu suất theo dõi sẽ giúp nâng cao khả năng theo dõi đối tượng trong nhiều điều kiện khác nhau, từ đó cải thiện độ chính xác và độ ổn định của hệ thống.

Hướng phát triển

Để khắc phục những hạn chế và nâng cao hiệu quả của hệ thống trong tương lai, hướng phát triển tiếp theo của nghiên cứu cần tập trung vào hai vấn đề chính: tăng cường dữ liệu huấn luyện và tinh chỉnh các tham số của thuật toán SFSORT.

Đầu tiên, việc tăng cường dữ liệu huấn luyện là bước quan trọng nhằm nâng cao khả năng nhận diện của mô hình YOLOv8. Bằng cách bổ sung thêm các tập dữ liệu đa dạng và phong phú hơn, bao gồm các tình huống phức tạp như đối tượng bị che khuất, ánh sáng yếu, hoặc các điều kiện thời tiết khắc nghiệt, mô hình có thể học được các đặc trưng phức tạp và nâng cao độ chính xác trong việc nhận diện đối tượng.

Thứ hai, việc tinh chỉnh các tham số của thuật toán SFSORT để cải thiện khả năng theo vết đối tượng là một hướng nghiên cứu quan trọng. Các tham số cần được nghiên cứu kỹ lưỡng để tối ưu hóa hiệu suất theo dõi.

Những cải tiến này không chỉ giúp nâng cao độ chính xác và hiệu quả của hệ thống mà còn mở rộng khả năng ứng dụng trong các lĩnh vực thực tế như giám sát an ninh, quản lý giao thông thông minh, và nghiên cứu môi trường. Việc nghiên cứu và phát triển thêm các phương pháp này sẽ góp phần quan trọng vào việc xây dựng các hệ thống giám sát thông minh, hiện đại và hiệu quả hơn trong tương lai.

TÀI LIỆU THAM KHẢO

- [1] P. Sharma, «A Practical Guide to Object Detection using the Popular YOLO Framework – Part III (with Python codes),» 15 2 2014. [En ligne]. Available: <https://www.analyticsvidhya.com/blog/2018/12/practical-guide-object-detection-yolo-framework-python/>.
- [2] F. Jacob Solawetz, «What is YOLOv8? The Ultimate Guide. [2024],» 11 1 2023. [En ligne]. Available: <https://blog.roboflow.com/whats-new-in-yolov8/>.
- [3] D. P. Hidayatullah, «YOLOv8 Architecture Detailed Explanation - A Complete Breakdown,» Youtube, 2023.
- [4] D.-M. C.-E. Juan Terven, «A COMPREHENSIVE REVIEW OF YOLO ARCHITECTURES,» *ArXiv*, vol. abs/2304.00501, 2024.
- [5] soumyadip, «Mastering All YOLO Models from YOLOv1 to YOLOv9: Papers Explained (2024),» 26 12 2023. [En ligne]. Available: <https://learnopencv.com/mastering-all-yolo-models/>.
- [6] ultralytics. [En ligne]. Available: <https://github.com/ultralytics/ultralytics>.
- [7] T. KELDENICH, «YOLOv8 – How to use it? Best Tutorial,» 18 1 2023. [En ligne]. Available: <https://inside-machinelearning.com/en/yolov8-how-to-use/#boxzilla-13443>.
- [8] T. KELDENICH, «How to make Bounding Boxes in Python – Best function,» 18 1 2023. [En ligne]. Available: <https://inside-machinelearning.com/en/bounding-boxes-python-function/>.

- [9] B. Esme, «Kalman Filter For Dummies,» 3 2009. [En ligne]. Available: <http://bilgin.esme.org/BitsAndBytes/KalmanFilterforDummies#>.
- [10] O. K. Alessandro Minisini, «Hungarian algorithm for solving the assignment problem,» 13 12 2023. [En ligne]. Available: <https://cp-algorithms.com/graph/hungarian-algorithm.html>.
- [11] Z. G. L. O. F. R. B. U. Alex Bewley, «Simple Online and Realtime Tracking,» *ArXiv*, vol. abs/1602.00763, 2017.
- [12] «SORT: Simple Online and Realtime Tracking,» 8 4 2023. [En ligne]. Available: <https://www.luffca.com/2023/04/multiple-object-tracking-sort/>.
- [13] B. T. Tung, «SORT - Deep SORT : Một góc nhìn về Object Tracking (phần 1),» 16 12 2020. [En ligne]. Available: <https://viblo.asia/p/sort-deep-sort-mot-goc-nhin-ve-object-tracking-phan-1-Az45bPooZxY>.
- [14] abewley. [En ligne]. Available: <https://github.com/abewley/sort/blob/master/sort.py>.
- [15] abewley. [En ligne]. Available: <https://github.com/abewley/sort>.
- [16] gitmehrdad. [En ligne]. Available: <https://github.com/gitmehrdad/SFSORT>.
- [17] gitmehrdad. [En ligne]. Available: https://github.com/gitmehrdad/SFSORT/blob/main/SFSORT_YOLOv8.ipynb.
- [18] Z. S. F. F. S. H. H. M. S. B. S. M. M. Morsali, «SFSORT: Scene Features-based Simple Online Real-Time Tracker,» *ArXiv*, vol. abs/2404.07553, 2024.
- [19] Mosesdaudu, «Object Detection & Tracking With YOLOv8 and Sort Algorithm,» 28 3 2023. [En ligne]. Available:

<https://medium.com/@mosesdaudu001/object-detection-tracking-with-yolov8-and-sort-algorithm-363be8bc0806>.

