BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

ĐỒ ÁN TỐT NGHIỆP

NGÀNH : CÔNG NGHỆ THÔNG TIN

Sinh viên : Hoàng Hữu Cường

Giáo viên hướng dẫn : TS. Hồ Thị Hương Thơm

HẢI PHÒNG – 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

XÂY DỰNG MÔ HÌNH PHÁT HIỆN VẬT CẢN VÀ BIỂN BÁO HỖ TRỢ RA QUYẾT ĐỊNH CHO XE TỰ LÁI

ĐỒ ÁN TỐT NGHIỆP ĐẠI HỌC HỆ CHÍNH QUY

NGÀNH: Công nghệ thông tin

Sinh viên : Hoàng Hữu Cường

Giáo viên hướng dẫn : TS. Hồ Thị Hương Thơm

HẢI PHÒNG – 2025

BỘ GIÁO DỤC VÀ ĐÀO TẠO

TRƯỜNG ĐẠI HỌC QUẢN LÝ VÀ CÔNG NGHỆ HẢI PHÒNG

NHIỆM VỤ ĐỀ TÀI TỐT NGHIỆP

Tên đề tài: Xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái.

Sinh viên làm đề tài tốt ngiệp

Sinh viên : Hoàng Hữu Cường **Mã SV**: **2112111020**

Lớp : CT2501C

Ngành : Công nghệ thông tin

Tên đề tài : Xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái

Nội dung và các yêu cầu cần giải quyết trong nhiệm vụ đề tài tốt nghiệp

a) Mô tả tóm tắt đề tài

Tìm hiểu về mô hình phát hiện đối tượng bằng YOLO11 cho bài toán phát hiện loại hàng hóa , thiết kế xây dựng mô hình phát hiện vật cản và biển báo.

b) Nội dung hướng dẫn

- Tìm hiểu về bài toán phát hiện đối tượng nói chung và phát hiện phát hiện đối tượng bằng YOLO11.
- Phân tích, thu thập dữ liệu hình ảnh hàng hóa để huấn luyện phát hiện đối tượng.
- Cài đặt mô hình phát hiện đối tượng bằng YOLO11
- Thử nghiệm mô hình trên hình ảnh và video để phát hiện vật cản và biển báo.
- Nhận xét đánh giá, kết luận và đưa ra khuyến nghị khi ứng dụng

c) Kết quả cần đạt được

- Đã xây dựng được mô hình phát hiện đối tượng vật vản và biển báo bằng YOLO11.
- Xây dựng được ứng dụng phát hiện vật cản và biển báo trên hình ảnh và video.
- Viết báo cáo đồ án tốt nghiệp.

LỜI CẢM ƠN

Trong quá trình làm đồ án vừa qua vì được sự chỉ dẫn nhiệt tình của cô TS. Hồ Thị Hương Thơm – Trường Đại học Hàng hải Việt Nam, em đã hoàn thành đồ án của mình. Mặc dù em đã cố gắng với sự tận tâm của cô, nhưng vì thời gian và khả năng nên đồ án của em vẫn còn không tránh được những điều thiếu sót.

Em xin chân thành và bày tỏ lòng biết ơn sâu sắc đến cô vì đã tận tình chỉ bảo, hướng dẫn và giành thời gian quý báu của mình cho em trong thời gian qua để em có thể hoàn thành đồ án của mình đúng thời hạn.

Em xin cảm ơn tất cả thầy cô giáo trong khoa Công nghệ thông tin vì đã truyền đạt cho em rất nhiều các kiến thức nền tảng, chuyên ngành, chuyên môn và chuyên sâu cực kì vững chắc trong những năm qua để em có thể hoàn thành được đồ án này.

Em xin cảm ơn Trường Đại học Quản lý và Công nghệ Hải Phòng vì không ngừng hỗ trợ và đào tạo những điều kiện tốt nhất trong những năm vừa qua để em có thể học và thực hiện tốt đồ án.

Em xin chân thành cảm ơn !

LỜI CAM ĐOAN

Sinh viên xin cam đoan đề tài "Xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái" là công trình nghiên cứu độc lập dưới sự hướng dẫn của TS. Hồ Thị Hương Thơm – Giảng viên Khoa Công nghệ thông tin Trường Đại Học Hàng Hải Việt Nam. Đề tài là sự cố gắng, quyết tâm của cá nhân sinh viên trong việc tiếp cận kiến thức chuyên môn mới để hoàn thành đạt mục tiêu của đề tài. Các dữ liệu đề nghiên cứu thực nghiệm là trung thực có nguồn gốc trích dẫn rõ ràng. Sinh viên xin hoàn toàn chịu trách nghiệm với lời cam đoan này

Người cam đoan

Hoàng Hữu Cường

MỤC LỤC

DANH MỤC HÌNH

DANH MỤC BẢNG

BẢNG CỤM TỪ VIẾT TẮT

Từ viết tắt	Tiếng Anh	Tiếng Việt
AI	Artificial Intelligence	Trí tuệ nhân tạo
ML	Machine Learning	Học máy
DL	Deep Learning	Học sâu
CV	Computer Vision	Thị giác máy tính
CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
YOLO	You Only Look Once	Phát hiện đối tượng thời gian thực
YOLOv4/v5/v8/v11	YOLO versions	Các phiên bản YOLO
mAP	Mean Average Precision	Độ chính xác trung bình
GPU	Graphics Processing Unit	Bộ xử lý đồ họa
TPU	Tensor Processing Unit	Bộ xử lý Tensor
CPU	Central Processing Unit	Bộ xử lý trung tâm
API	Application Programming Interface	Giao diện lập trình ứng dụng
UI	User Interface	Giao diện người dùng
VS Code	Visual Studio Code	Trình soạn thảo mã nguồn VS Code
SDK	Software Development Kit	Bộ công cụ phát triển phần mềm

AVD	Android Virtual Device	Thiết bị ảo Android
APK	Android Package Kit	Tệp cài đặt Android
ADAS	Advanced Driver Assistance Systems	Hệ thống hỗ trợ lái xe nâng cao
RGB	Red – Green – Blue	Mô hình màu 3 kênh
TF Lite	TensorFlow Lite	Phiên bản TensorFlow nhẹ
KITTI	KITTI Dataset	Bộ dữ liệu giao thông KITTI
BDD100K	Berkeley DeepDrive 100K	Bộ dữ liệu giao thông 100.000 ảnh
REST API	Representational State Transfer API	API phong cách REST
IDE	Integrated Development Environment	Môi trường phát triển tích hợp

CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN VỀ ĐỀ TÀI

Chương 1 cho ta thấy cái nhìn tổng quan về đề tài "Xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái". Chương mở đầu bằng việc đặt vấn đề, làm rõ tầm quan trọng của việc phát hiện chính xác các vật cản và biển báo giao thông đối với sự an toàn và hiệu quả của xe tự lái. Tiếp theo, chương trình bày các phương pháp phát hiện đối tượng, đặc biệt là việc ứng dụng các mô hình học sâu, để nhận diện vật cản và biển báo trong môi trường giao thông**.** Phạm vi đề tài được xác định rõ, giới hạn nghiên cứu trong việc phát hiện vật cản và biển báo giao thông, đồng thời trình bày các mục tiêu và ý nghĩa của đề tài, nhấn mạnh đóng góp của nghiên cứu đối với sự phát triển của công nghệ xe tự lái.

1.1. Đặt vấn đề

Trong những năm gần đây, ngành công nghiệp ô tô đang chứng kiến một cuộc cách mạng nhờ sự phát triển mạnh mẽ của trí tuệ nhân tạo (AI), đặc biệt là học sâu (Deep Learning) và thị giác máy tính (Computer Vision). Các công nghệ này đã và đang được ứng dụng rộng rãi trong nhiều lĩnh vực:

Y tế: hỗ trợ chẩn đoán hình ảnh y khoa (X-quang, MRI, CT-Scan), phát hiện sớm ung thư, phân tích bệnh lý.

An ninh giám sát: nhận diện khuôn mặt, phát hiện hành vi bất thường, theo dõi đối tượng trong thời gian thực.

Thương mại điện tử: gợi ý sản phẩm, phân tích hành vi người dùng, tối ưu quảng cáo cá nhân hóa.

Nông nghiệp thông minh: giám sát cây trồng, phát hiện sâu bệnh, tự động phân loại và thu hoạch nông sản.

Robot và tự động hóa công nghiệp: điều khiển robot, kiểm tra chất lượng sản phẩm, giám sát dây chuyền sản xuất.

Trong số các ứng dụng đó, hệ thống xe tự lái (Autonomous Vehicles – AV) nổi lên như một trong những lĩnh vực tiêu biểu và đầy tiềm năng, hứa hẹn tạo ra cuộc cách mạng trong ngành giao thông vận tải toàn cầu. Xe tự lái không chỉ áp dụng AI để "nhìn thấy" và "hiểu" môi trường xung quanh, mà còn có khả năng ra quyết định lái thay con người, nhằm mang lại an toàn, hiệu quả và tiện ích vượt trội.

Hình 1.1.Hình ảnh về tự lái

Xe tự lái không chỉ là xu hướng công nghệ mà còn là giải pháp tiềm năng cho các vấn đề toàn cầu: giảm thiểu tai nạn, cải thiện hiệu quả giao thông, bảo vệ môi trường và nâng cao chất lượng cuộc sống. Các tập đoàn công nghệ hàng đầu như Tesla, Waymo (Alphabet), Baidu, Nvidia, cùng các hãng xe truyền thống như Mercedes-Benz, BMW, Toyota, Hyundai… đã đầu tư hàng tỷ đô la Mỹ vào nghiên cứu, phát triển và thử nghiệm công nghệ này trên quy mô toàn cầu.

Một trong những yếu tố then chốt giúp xe tự lái hoạt động an toàn và hiệu quả là khả năng phát hiện và nhận diện vật cản cũng như biển báo giao thông. Hệ thống phải đảm bảo phát hiện chính xác các phương tiện khác, người đi bộ, xe đạp… và hiểu được ngữ cảnh giao thông thông qua việc nhận diện biển báo để đưa ra quyết định điều khiển phù hợp.

Tuy nhiên, hệ thống phát hiện và nhận dạng đối tượng cho xe tự lái vẫn đối mặt với nhiều thách thức:

- Các thách thức đặt ra: (Sensor Fusion) đặt ra thách thức lớn về tính toán và thuật toán. Điều kiện môi trường thay đổi phức tạp.

- Ánh sáng yếu (ban đêm, hầm tối), chói sáng (ánh nắng trực tiếp).

- Thời tiết bất lợi như mưa lớn, sương mù, tuyết hoặc bụi mịn.

- Các yếu tố che khuất một phần đối tượng (xe bị chắn, người đi bộ bị che) Những yếu tố này khiến hệ thống thị giác máy tính khó duy trì độ chính xác ổn định.

a) Yêu cầu xử lý thời gian thực (Real-time Processing)

Xe tự lái phải phản ứng trong vài phần nghìn giây để kịp thời xử lý tình huống nguy hiểm (ví dụ: người đi bộ băng qua đường, xe phía trước phanh gấp). Do đó, mô hình AI cần:

- Nhẹ và tối ưu hóa để chạy được trên các phần cứng giới hạn (GPU trên xe, Edge AI).
- Nhanh nhưng vẫn chính xác, tránh tình trạng độ trễ gây nguy hiểm.

Độ chính xác và độ tin cậy cao:

- Sai sót trong nhận diện dù chỉ 1% cũng có thể dẫn đến hậu quả nghiêm trọng. Ví dụ: nhầm biển “Stop” thành biển “Speed Limit 50” hoặc không nhận diện được người đi bộ.
- Mô hình cần đảm bảo khả năng tổng quát hóa (generalization), hoạt động tốt không chỉ trong môi trường được huấn luyện mà còn trong những tình huống thực tế đa dạng.

Tính đa dạng của đối tượng và ngữ cảnh giao thông:

- Các loại phương tiện: ô tô, xe máy, xe đạp, xe buýt, xe tải… với hình dạng, kích thước và màu sắc khác nhau.
- Biển báo giao thông: có hàng trăm loại, thay đổi theo từng quốc gia, đôi khi bị hỏng, xoay lệch hoặc bị cản trở tầm nhìn.
- Người đi bộ với nhiều dáng đi, tư thế, thậm chí là mang đồ hoặc dắt thú nuôi.

b) Yêu cầu tích hợp đa cảm biến

- Camera RGB, Camera hồng ngoại, Radar, LiDAR… đều cần kết hợp để bù trừ điểm yếu cho nhau. Ví dụ: LiDAR hoạt động tốt trong sương mù, trong khi camera RGB hữu ích để nhận diện màu sắc biển báo.
- Việc xử lý và đồng bộ dữ liệu đa cảm biến

1.2 Phát hiện đối tượng

Phát hiện đối tượng (Object Detection) là một lĩnh vực quan trọng trong thị giác máy tính (Computer Vision, có nhiệm vụ xác định:

- Sự hiện diện (Existence): trong ảnh hoặc video có xuất hiện đối tượng quan tâm hay không.
- Vị trí (Localization): đối tượng nằm ở đâu trong khung hình, thông qua việc dự đoán các hộp giới hạn (Bounding Boxes).
- Loại đối tượng (Classification): đối tượng thuộc lớp nào (ví dụ: người, ô tô, xe máy, biển báo giao thông…).

Điểm khác biệt cơ bản giữa phân loại hình ảnh (Image Classification) và phát hiện đối tượng (Object Detection) nằm ở khả năng định vị:

- Phân loại hình ảnh: chỉ cho biết trong ảnh có chứa một đối tượng hay không (ví dụ: “có người đi bộ” hoặc “không có người đi bộ”).
- Phát hiện đối tượng: không chỉ cho biết có người đi bộ mà còn chỉ ra người đi bộ ở đâu, bao nhiêu người, và thậm chí nhiều loại đối tượng khác cùng lúc (người, xe, đèn giao thông, biển báo…).

1.3.Phương pháp phát hiện đối tượng

1.3.1. Phương pháp phát hiện truyền thống.

Trước năm 2014, phát hiện đối tượng dựa trên các phương pháp truyền thống:

- HOG (Histogram of Oriented Gradients), SIFT, SURF: trích xuất đặc trưng thủ công từ ảnh.
- Bộ phân loại (Classifier) như SVM, Adaboost: dùng để nhận dạng đối tượng dựa trên đặc trưng.
- Sliding Window: quét toàn bộ ảnh ở nhiều tỉ lệ để tìm đối tượng dẫn đến rất chậm, kém hiệu quả.

Những phương pháp này có hạn chế:

- Độ chính xác không cao, dễ bị ảnh hưởng bởi ánh sáng, tư thế, kích thước khác nhau của đối tượng.
- Không đáp ứng được yêu cầu xử lý thời gian thực trong các hệ thống phức tạp như xe tự lái.

Từ năm 2014 trở đi, sự xuất hiện của học sâu (Deep Learning), đặc biệt là mạng nơ-ron tích chập (Convolutional Neural Networks – CNNs), đã tạo ra bước ngoặt trong phát hiện đối tượng. Các nghiên cứu như R-CNN, Fast R-CNN, Faster R-CNN, SSD, YOLO đã đưa hiệu năng phát hiện đối tượng lên một tầm cao mới, đạt cả:

- Độ chính xác cao (High Accuracy).
- Khả năng xử lý thời gian thực (Real-time Processing).

1.3.2 Phương pháp phát hiện dựa trên học sâu.

Hiện nay, hầu hết các nghiên cứu và ứng dụng trong phát hiện đối tượng đều dựa vào học sâu (Deep Learning), cụ thể là các mô hình mạng nơ-ron tích chập (Convolutional Neural Networks – CNNs). Các phương pháp phát hiện đối tượng hiện đại được chia thành hai nhóm chính:

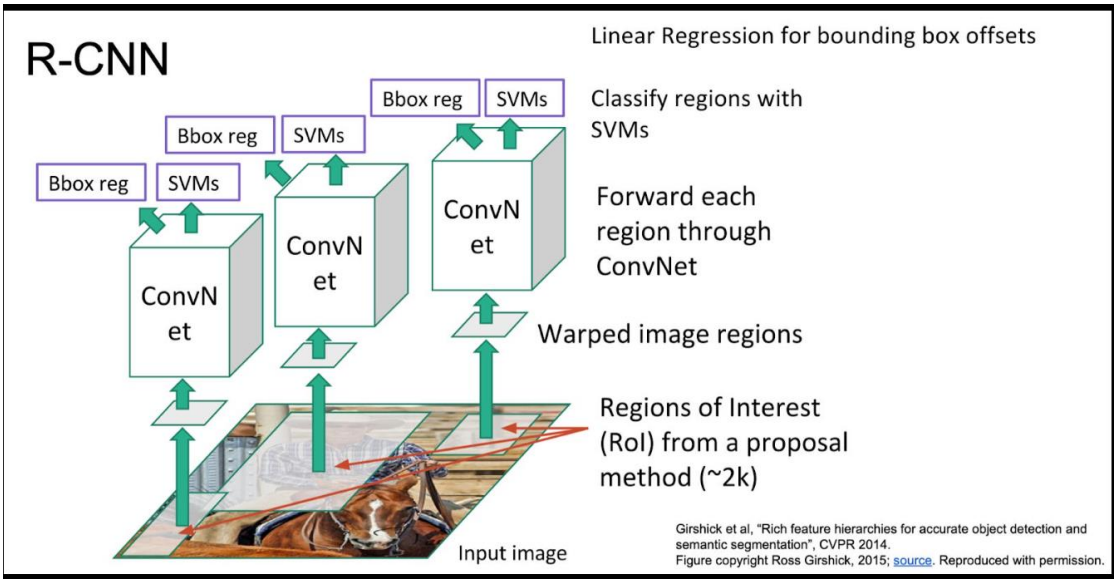
a) Phương pháp hai giai đoạn

Các mô hình hai giai đoạn hoạt động theo cơ chế "đề xuất – phân loại":

- Giai đoạn 1: Tạo ra các vùng đề xuất (Region Proposals) có khả năng chứa đối tượng.
- Giai đoạn 2: Thực hiện phân loại (object classification) và tinh chỉnh (bounding box regression) trên các vùng đã được đề xuất.

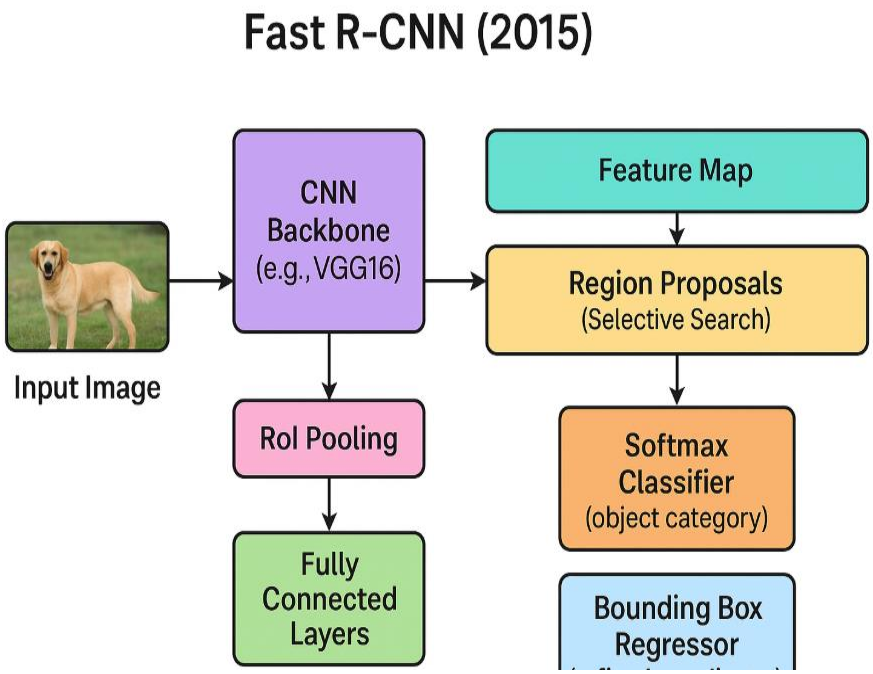
Các mô hình tiêu biểu:

- R- CNN là một trong những phương pháp tiên phong ứng dụng CNN cho phát hiện đối tượng. R-CNN sử dụng thuật toán Selective Search để tạo vùng đề xuất, sau đó áp dụng CNN để trích xuất đặc trưng và phân loại. Tuy nhiên, nhược điểm lớn là tốc độ chậm (mất vài chục giây để xử lý một ảnh).



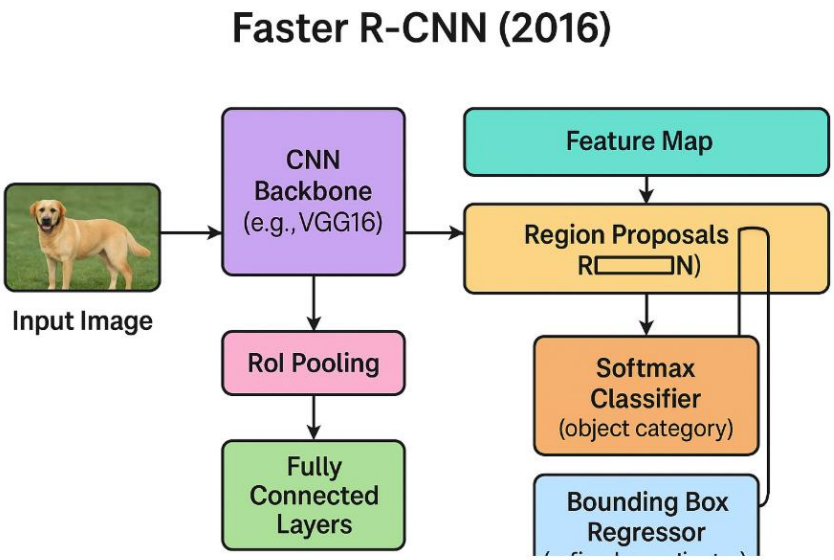
Hình 1.2. Cấu trúc R-CNN

- Fast R-CNN (2015) - Cải tiến từ R-CNN bằng cách sử dụng RoI Pooling, cho phép trích xuất đặc trưng một lần trên toàn ảnh thay vì cho từng vùng đề xuất. Điều này giúp giảm đáng kể thời gian tính toán, tuy nhiên vẫn phụ thuộc vào bước Selective Search nên tốc độ chưa đạt mức thời gian thực.



Hình 1.3. Fast R-CNN (2015)

- Faster R-CNN(2016) là thay thế Selective Search bằng Region Proposal Network (RPN) – một mạng nơ-ron học sâu để sinh vùng đề xuất. Đây là bước đột phá giúp tăng cả độ chính xác và tốc độ xử lý, trở thành một trong những mô hình hai giai đoạn phổ biến nhất trong nghiên cứu học thuật.



Hình 1.4. Cấu trúc Faster R-CNN

Đặc điểm chung:

- Ưu điểm: Độ chính xác cao, khả năng nhận diện tốt ngay cả với đối tượng nhỏ hoặc bị che khuất một phần.
- Nhược điểm: Tốc độ xử lý chậm hơn so với các mô hình một giai đoạn, do đó chưa phù hợp với yêu cầu thời gian thực khắt khe của xe tự lái.

b) Phương pháp một giai đoạn

Khác với mô hình hai giai đoạn, các mô hình một giai đoạn thực hiện phát hiện trực tiếp trên toàn ảnh, tức là đồng thời dự đoán cả bounding box và nhãn lớp mà không cần bước sinh vùng đề xuất riêng biệt.

Các mô hình tiêu biểu:

- - SSD (Single Shot MultiBox Detector, 2016):

Sử dụng nhiều tầng đặc trưng (feature maps) ở các độ phân giải khác nhau để phát hiện đối tượng với nhiều kích thước. SSD cân bằng tốt giữa tốc độ và độ chính xác, thường đạt được kết quả thời gian thực (~22 fps trên GPU).

- YOLO (You Only Look Once):

Đây là họ mô hình phát hiện đối tượng nổi tiếng và được ứng dụng rộng rãi nhất trong xe tự lái.

YOLOv1 – YOLOv3: tập trung vào tốc độ vượt trội, có thể xử lý hàng chục khung hình mỗi giây.

YOLOv4 – YOLOv7: cải thiện đáng kể về độ chính xác, đồng thời vẫn giữ tốc độ cao.

YOLOv8, YOLO11 (mới): được phát triển mạnh mẽ nhờ tối ưu kiến trúc và khả năng triển khai trên phần cứng hạn chế (Edge Devices).

Đặc điểm chung:

- Ưu điểm: Tốc độ xử lý nhanh, phù hợp cho ứng dụng thời gian thực như giám sát video, robot, và đặc biệt là xe tự lái.
- Nhược điểm: Độ chính xác có thể thấp hơn so với mô hình hai giai đoạn, đặc biệt là với các đối tượng nhỏ hoặc chồng lấn. Tuy nhiên, các phiên bản mới của YOLO đã thu hẹp đáng kể khoảng cách này.

c) Nhận xét chung

Các mô hình hai giai đoạn (Faster R-CNN, Mask R-CNN) vẫn thường được dùng trong nghiên cứu hoặc ứng dụng đòi hỏi độ chính xác tuyệt đối (ví dụ: phân tích ảnh y khoa, giám sát an ninh chất lượng cao).

Các mô hình một giai đoạn (SSD, YOLO) được ưu tiên trong các ứng dụng thời gian thực, nơi tốc độ xử lý là yếu tố sống còn, điển hình là xe tự lái.

Do đó, trong bối cảnh đề tài này – xây dựng mô hình phát hiện vật cản và biển báo hiệu hỗ trợ ra quyết định cho xe tự lái – việc lựa chọn các mô hình một giai đoạn (đặc biệt là YOLO) là hướng tiếp cận hợp lý, vừa đảm bảo tốc độ xử lý, vừa duy trì độ chính xác cần thiết.

1.4. Mô tả phạm vi của đề tài

- Giới thiệu bài toán

Trong những năm gần đây, sự phát triển nhanh chóng của công nghệ xe tự lái đã tạo ra một cuộc cách mạng trong ngành công nghiệp ô tô và giao thông vận tải. Các hệ thống xe tự hành ngày nay không còn đơn thuần là một ý tưởng viễn tưởng, mà đã và đang được nghiên cứu, phát triển và thử nghiệm bởi nhiều tập đoàn công nghệ lớn như Tesla, Waymo (Google), Baidu, và nhiều công ty khởi nghiệp khác trên toàn thế giới.

Một trong những thách thức lớn nhất trong việc phát triển xe tự lái là làm sao để xe có thể "nhìn thấy" và hiểu được thế giới xung quanh nó giống như con người. Để thực hiện được điều này, xe tự lái cần được trang bị các cảm biến như

camera, Lidar, Radar, GPS... Tuy nhiên, chỉ có dữ liệu từ cảm biến là chưa đủ. Xe cần có khả năng xử lý và diễn giải dữ liệu này để nhận diện được các yếu tố quan trọng trong môi trường như người đi bộ, xe cộ, chướng ngại vật, biển báo giao thông, tín hiệu đèn đường, v.v.

Trong số các chức năng cốt lõi của một hệ thống xe tự lái, khả năng phát hiện vật cản và biển báo giao thông là một trong những thành phần quan trọng nhất của hệ thống nhận thức (Perception System). Đây là cơ sở đầu tiên để xe hiểu được môi trường xung quanh, nhận biết những nguy hiểm tiềm ẩn và đưa ra các quyết định phù hợp trong quá trình di chuyển. Nếu hệ thống không phát hiện được vật cản hoặc biển báo, xe có thể gặp nguy hiểm, gây ra tai nạn hoặc vi phạm luật giao thông.

Đặc biệt, trong các môi trường đô thị phức tạp, mật độ giao thông cao, có nhiều phương tiện và người đi bộ di chuyển hỗn hợp, việc phát hiện nhanh chóng và chính xác các đối tượng là yếu tố then chốt để đảm bảo an toàn. Một hệ thống phát hiện kém hiệu quả không chỉ ảnh hưởng đến trải nghiệm người dùng mà còn có thể gây nguy hiểm đến tính mạng.

Chính vì vậy, bài toán đặt ra là: làm thế nào để xây dựng một hệ thống có thể phát hiện vật cản và biển báo với độ chính xác cao, tốc độ xử lý nhanh, ổn định trong các điều kiện môi trường khác nhau, từ đó tạo nền tảng cho việc ra quyết định lái xe một cách thông minh và an toàn.

1.5. Mục tiêu của đề tài

Đề tài này tập trung vào việc nghiên cứu, thiết kế và triển khai một hệ thống phát hiện vật cản và nhận diện biển báo giao thông dựa trên học sâu (Deep Learning), nhằm phục vụ cho ứng dụng xe tự lái và các hệ thống hỗ trợ lái nâng cao (ADAS). Mục tiêu vừa mang tính học thuật, nhằm chứng minh hiệu quả của các mô hình phát hiện đối tượng hiện đại, vừa mang tính ứng dụng thực tiễn,

hướng đến việc tạo ra một mô-đun có khả năng hoạt động trong điều kiện giao thông thực tế tại Việt Nam.

1.5.1. Ý nghĩa của mục tiêu

Thông qua việc đạt được các mục tiêu trên, đề tài sẽ:

- Góp phần nghiên cứu và phát triển công nghệ xe tự hành tại Việt Nam, nơi điều kiện giao thông có nhiều thách thức (lưu lượng xe máy lớn, đường phố đông đúc, hạ tầng chưa đồng bộ).

- Cung cấp một bộ mô hình và dữ liệu huấn luyện có thể tái sử dụng hoặc mở rộng cho các nghiên cứu tiếp theo.

- Tạo nền tảng quan trọng cho việc phát triển các hệ thống hỗ trợ lái xe nâng cao (ADAS – Advanced Driver Assistance Systems), hướng tới xe tự lái hoàn toàn trong tương lai gần.

1.6. Phạm vi của đề tài

Do tính chất phức tạp của hệ thống xe tự lái và giới hạn về thời gian, tài nguyên và kỹ thuật trong khuôn khổ một đồ án tốt nghiệp đại học, đề tài này tập trung vào một phạm vi nghiên cứu rõ ràng và cụ thể, bao gồm:

a) Phạm vi về chức năng:

Đề tài chỉ tập trung vào việc phát hiện và nhận diện đối tượng từ hình ảnh hoặc video thu được từ camera.

Không đi sâu vào các chức năng điều khiển xe như tự động rẽ, tự động phanh, bám làn, hay định vị – đây là những phần thuộc về hệ thống điều khiển và điều hướng (Control & Planning).

Việc ra quyết định điều khiển chỉ dừng lại ở mức đơn giản, dựa trên các luật logic định sẵn (rule-based) như: dừng khi gặp biển “STOP”, giảm tốc khi gặp vật cản trong khoảng gần.

b) Phạm vi về dữ liệu và cảm biến:

Đầu vào hệ thống là hình ảnh 2D từ camera RGB thông thường, không sử dụng các cảm biến phức tạp hơn như Lidar, Radar, hoặc dữ liệu 3D.

Dữ liệu sử dụng để huấn luyện và đánh giá mô hình chủ yếu là dữ liệu công khai, chẳng hạn như KITTI hoặc BDD100K, và/hoặc video thực tế được ghi lại trong môi trường giao thông.

c) Phạm vi về môi trường thử nghiệm:

Việc thử nghiệm và đánh giá hệ thống chủ yếu được thực hiện trong môi trường mô phỏng hoặc video offline, không triển khai trực tiếp trên xe thật hoặc môi trường ngoài trời thực tế.

Điều này nhằm đảm bảo tính khả thi trong phạm vi đồ án, đồng thời vẫn cho phép kiểm tra khả năng phát hiện và phản ứng của hệ thống trong các tình huống đa dạng.

d) Phạm vi kỹ thuật:

Đề tài sử dụng các thư viện mã nguồn mở như YOLO11, PyTorch, OpenCV để xây dựng và huấn luyện mô hình.

Không phát triển hệ thống từ đầu mà sử dụng kiến trúc mô hình có sẵn và thực hiện tinh chỉnh (fine-tuning) để phù hợp với mục tiêu.

1.7. Giải pháp xử lý bài toán

Để giải quyết bài toán, đề tài xây dựng giải pháp gồm các bước như sau:

Bước 1: Thu thập và xử lý dữ liệu

- Sử dụng các bộ dữ liệu công khai như KITTI, BDD100K, hoặc tự thu thập video thực tế.
- Tiền xử lý dữ liệu: resize, chuẩn hóa, augmentation để tăng độ đa dạng dữ liệu đầu vào.

Bước 2: Huấn luyện mô hình phát hiện đối tượng

- Lựa chọn mô hình YOLO11 để huấn luyện, do hiệu suất cao và hỗ trợ tốt cho các loại đối tượng giao thông.
- Tinh chỉnh (fine-tune) mô hình trên tập dữ liệu có gán nhãn cụ thể cho vật cản và biển báo.

Bước 3: Nhận diện và xử lý hậu kỳ

- Sau khi phát hiện, áp dụng thuật toán Non-Maximum Suppression (NMS) để loại bỏ hộp giới hạn trùng lặp.
- Hiển thị kết quả trực quan: vẽ bounding box, nhãn đối tượng lên màn hình video.

Bước 4: Hỗ trợ ra quyết định điều khiển

- Nhận diện được đối tượng ra quyết định né tránh hoặc dừng lại.

Bước 5: Đánh giá và thử nghiệm

- Đánh giá hiệu suất mô hình dựa trên các chỉ số: Accuracy, Precision, Recall, F1-Score.
- Kiểm thử hệ thống trong các video mô phỏng tình huống thực tế để kiểm tra khả năng phản ứng.

1.8.Giới thiệu về YOLO và các phiên bản của YOLO.

1.8.1 .Giới thiệu về YOLO và phiên bản YOLO11

YOLO (You Only Look Once) là một thuật toán tiên tiến trong lĩnh vực thị giác máy tính, nổi tiếng với tốc độ xử lý vượt trội. Đúng như tên gọi, YOLO xử lý toàn bộ hình ảnh chỉ trong một lần duy nhất. Thay vì quét ảnh nhiều lần để tìm đối tượng, YOLO chia hình ảnh thành các ô lưới và dự đoán đồng thời các hộp giới hạn, độ tin cậy và lớp đối tượng cho mỗi ô, từ đó cho phép phát hiện đối tượng theo thời gian thực.

1.8.2.Các phiên bản YOLO:

HÌnh 1.5. Quá trình phát triển YOLO

YOLOv3: Đây là phiên bản thứ ba của thuật toán phát hiện đối tượng You Only Look Once (YOLO). Được phát triển ban đầu bởi Joseph Redmon, YOLOv3 đã cải tiến so với các phiên bản tiền nhiệm bằng cách giới thiệu các tính năng như dự đoán đa tỷ lệ và ba kích thước kernel phát hiện khác nhau.

YOLOv4 là viết tắt của You Only Look Once phiên bản 4. Đây là một mô hình phát hiện đối tượng theo thời gian thực được phát triển để giải quyết những hạn chế của các phiên bản YOLO trước đó như YOLOv3 và các mô hình phát hiện đối tượng khác. Không giống như các công cụ phát hiện đối tượng dựa trên mạng nơ-ron tích chập (CNN) khác, YOLOv4 không chỉ áp dụng cho các hệ thống đề xuất mà còn cho quản lý quy trình độc lập và giảm đầu vào của con người. Hoạt động của nó trên các đơn vị xử lý đồ họa (GPU) thông thường cho phép sử dụng hàng loạt với giá cả phải chăng và nó được thiết kế để hoạt động trong thời gian thực trên một GPU thông thường trong khi chỉ yêu cầu một GPU như vậy để huấn luyện.

YOLOv5u thể hiện một bước tiến trong phương pháp luận phát hiện đối tượng. Bắt nguồn từ kiến trúc nền tảng của mô hình YOLOv5 do Ultralytics phát triển, YOLOv5u tích hợp split head không neo, không objectness, một tính năng đã được giới thiệu trước đó trong các mô hình YOLOv8. Sự điều chỉnh này tinh chỉnh kiến trúc của mô hình, dẫn đến sự cải thiện về sự đánh đổi giữa độ chính xác và tốc độ trong các tác vụ phát hiện đối tượng. Với các kết quả thực nghiệm và các tính năng có được, YOLOv5u cung cấp một giải pháp thay thế hiệu quả cho những người tìm kiếm các giải pháp mạnh mẽ trong cả nghiên cứu và ứng dụng thực tế.

YOLOv6 là một trình phát hiện đối tượng tiên tiến, mang lại sự cân bằng đáng kể giữa tốc độ và độ chính xác, khiến nó trở thành một lựa chọn phổ biến cho các ứng dụng thời gian thực. Mô hình này giới thiệu một số cải tiến đáng chú ý về kiến trúc và lược đồ huấn luyện, bao gồm việc triển khai mô-đun Bi-directional Concatenation (BiC), chiến lược anchor-aided training (AAT) và cải tiến backbone và thiết kế neck để đạt được độ chính xác hiện đại trên tập dữ liệu COCO.

YOLOv7 là một trình phát hiện đối tượng theo thời gian thực hiện đại, vượt trội hơn tất cả các trình phát hiện đối tượng đã biết về cả tốc độ và độ chính xác trong phạm vi từ 5 FPS đến 160 FPS. Nó có độ chính xác cao nhất (56,8% AP) trong số tất cả các trình phát hiện đối tượng theo thời gian thực đã biết với 30 FPS trở lên trên GPU V100. Hơn nữa, YOLOv7 vượt trội hơn các trình phát hiện đối tượng khác như YOLOR, YOLOX, Scaled-YOLOv4, YOLOv5 và nhiều trình khác về tốc độ và độ chính xác

YOLOv8 được Ultralytics phát hành vào ngày 10 tháng 1 năm 2023, mang lại hiệu suất vượt trội về độ chính xác và tốc độ. Dựa trên những tiến bộ của các phiên bản YOLO trước, YOLOv8 giới thiệu các tính năng và tối ưu hóa mới, khiến nó trở thành một lựa chọn lý tưởng cho nhiều tác vụ phát hiện đối tượng trong nhiều ứng dụng.

YOLOv9 đánh dấu một bước tiến đáng kể trong lĩnh vực phát hiện đối tượng theo thời gian thực, giới thiệu các kỹ thuật đột phá như Thông tin Gradient có thể lập trình (PGI) và Mạng tổng hợp lớp hiệu quả tổng quá (GELAN).. Dự án YOLOv9, mặc dù được phát triển bởi một nhóm mã nguồn mở riêng biệt, nhưng được xây dựng dựa trên nền tảng mã mạnh mẽ do Ultralytics YOLOv5 cung cấp, thể hiện tinh thần hợp tác của cộng đồng nghiên cứu AI.

YOLOv10, được xây dựng trên gói Python Ultralytics bởi các nhà nghiên cứu tại Đại học Thanh Hoa, giới thiệu một phương pháp mới để phát hiện đối tượng theo thời gian thực, giải quyết cả những thiếu sót trong quá trình xử lý hậu kỳ và kiến trúc mô hình được tìm thấy trong các phiên bản YOLO trước. Bằng cách loại bỏ non-maximum suppression (NMS) và tối ưu hóa các thành phần mô hình khác nhau, YOLOv10 đạt được hiệu suất hiện đại với chi phí tính toán giảm đáng kể.

CHƯƠNG 2. PHƯƠNG PHÁP PHÁT HIỆN ĐỐI TƯỢNG VẬT CẢN CHO XE TỰ LÁI

Chương 2 tập trung xây dựng và triển khai mô hình nhận diện dựa trên mạng nơ-ron sâu. Trước tiên, chương giới thiệu sơ đồ thuật toán và mô hình phát hiện đối tượng, mô tả cách hệ thống vận hành từ việc thu thập dữ liệu, xử lý hình ảnh đến nhận diện vật cản. Tiếp đó, phương pháp YOLO11 được trình bày chi tiết, bao gồm cấu trúc mạng, nguyên lý hoạt động, và khả năng nhận diện vật cản trong thời gian thực. Chương cũng mô tả cách sử dụng mô hình đã được huấn luyện để phát hiện đối tượng mới, đảm bảo độ chính xác và tốc độ xử lý phù hợp với xe tự lái. Cuối cùng, các cơ chế đưa ra cảnh báo và gửi dữ liệu hỗ trợ ra quyết định được trình bày, cho thấy cách hệ thống kết hợp kết quả nhận diện để giúp xe tự lái di chuyển an toàn.

2.1. Sơ đồ thuật toán và mô hình phát hiện đối tượng

2.1.1. Đối tượng cần nhận diện

a) Người đi bộ

Người đi bộ là đối tượng xuất hiện phổ biến trong môi trường giao thông. Việc nhận diện người đi bộ giúp hệ thống phát hiện sự hiện diện của con người trên đường và đưa ra các cảnh báo an toàn cần thiết.

Hình 2.1. Ảnh người đi bộ trên đường

b)Vật cản

Vật cản bao gồm những đối tượng có thể gây cản trở hoặc nguy hiểm khi phương tiện di chuyển, như xe khác, chướng ngại vật, đồ vật trên đường… Nhận diện vật cản giúp tránh va chạm và đảm bảo an toàn khi vận hành.

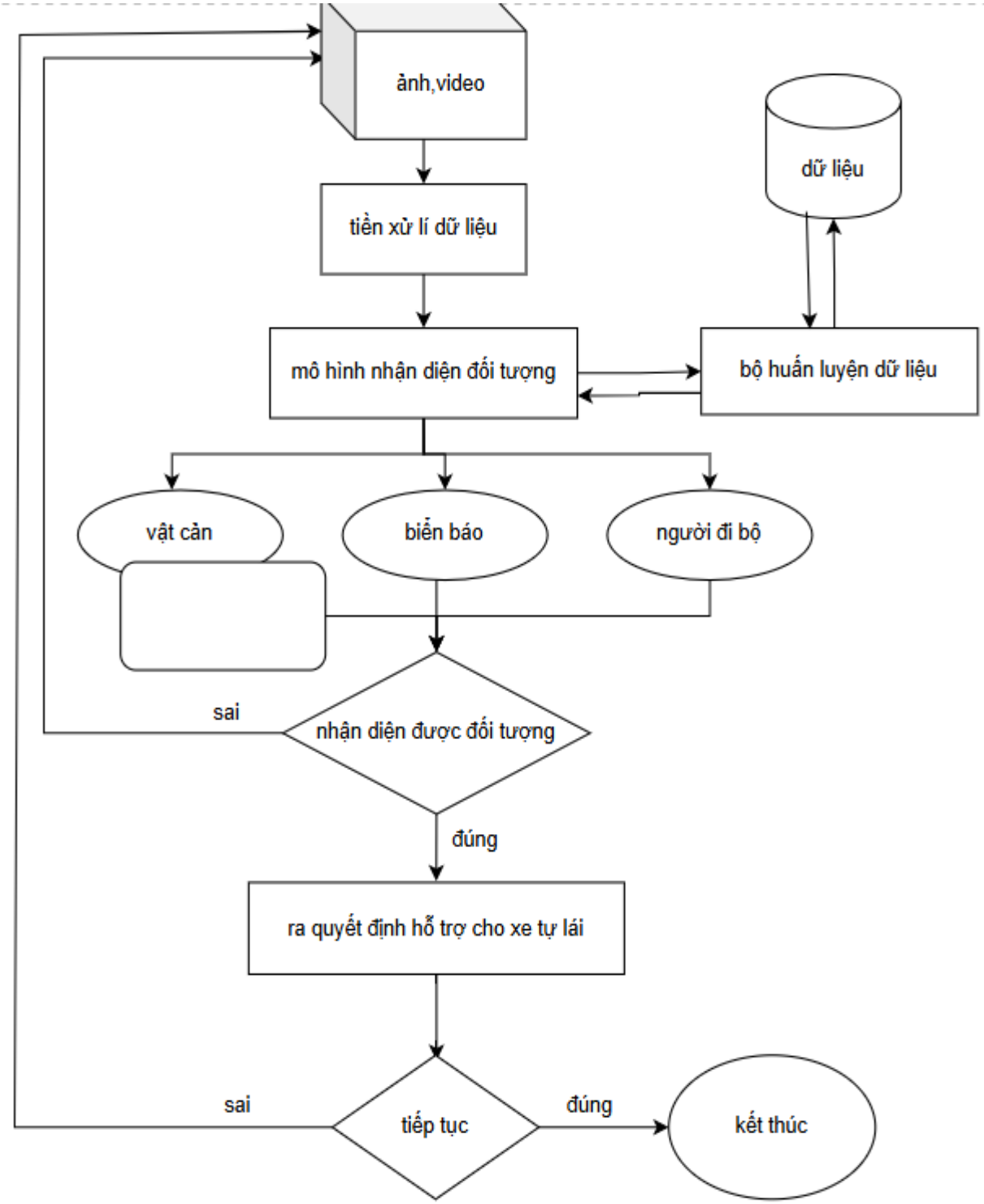
Hình 2.2. Ảnh vật cản trên đường

c)Biển báo

Biển báo giao thông cung cấp thông tin và quy định quan trọng trên đường. Nhận diện biển báo giúp hệ thống hiểu được các hướng dẫn như giới hạn tốc độ, cảnh báo nguy hiểm hoặc chỉ dẫn di chuyển.

Hình 2.3.Ảnh biển báo đèn giao thông

2.1.2Sơ đồ thuật toán bài toán



Hình 2.4.Hình sơ đồ thuật toán để giải quyết bài đề tài

Sơ đồ khối của hệ thống:

Đầu vào (Input): Hình ảnh được thu thập liên tục từ camera được gắn trên xe. Dữ liệu này có thể là các khung hình (frame) tĩnh từ video hoặc một luồng hình ảnh trực tiếp.

Tiền xử lý dữ liệu (Preprocessing): Các khung hình được chuẩn hóa về kích thước và định dạng phù hợp với yêu cầu của mô hình. Bước này đảm bảo rằng tất cả dữ liệu đầu vào đều tương thích, giúp mô hình hoạt động ổn định.

Mô hình phát hiện (Detection Model): Khung hình đã được tiền xử lý sẽ được đưa vào mô hình học sâu đã được huấn luyện (ví dụ: mô hình YOLO của bạn). Mô hình này sẽ xử lý hình ảnh và đưa ra các dự đoán.

Hậu xử lý kết quả (Post-processing): Mô hình có thể tạo ra nhiều hộp giới hạn chồng chéo nhau cho cùng một đối tượng. Bước này sử dụng thuật toán Non-Maximum Suppression (NMS) để loại bỏ các hộp giới hạn trùng lặp và chỉ giữ lại hộp có độ chính xác cao nhất, giúp kết quả rõ ràng và chính xác hơn.

Đầu ra (Output): Kết quả cuối cùng là các hộp giới hạn được vẽ xung quanh vật thể (ví dụ: người đi bộ, ô tô, biển báo), cùng với nhãn và độ tin cậy tương ứng.

Cảnh báo và ra quyết định (Alert and Decision Making): Dữ liệu về vị trí và loại vật cản sẽ được gửi đến hệ thống điều khiển của xe. Ví dụ, nếu phát hiện vật cản ở khoảng cách gần, hệ thống có thể kích hoạt cảnh báo phanh khẩn cấp.

2.2.Phương pháp phát hiện vật cản bằng YOLO11.

Về phiên bản YOLO11:là phiên bản mới nhất trong dòng YOLO của Ultralytics, một công cụ phát hiện đối tượng theo thời gian thực, định nghĩa lại những gì có thể với độ chính xác, tốc độ và hiệu quả vượt trội. Dựa trên những tiến bộ ấn tượng của các phiên bản YOLO trước, YOLO11 giới thiệu những cải tiến đáng kể trong kiến trúc và phương pháp huấn luyện, khiến nó trở thành một lựa chọn linh hoạt cho nhiều tác vụ thị giác máy tính.

YOLO11 là phiên bản mới nhất (2024-2025) do Ultralytics phát triển, kế thừa từ YOLOv8 và được tối ưu hơn về:

- Tốc độ inference: nhẹ hơn, triển khai được trên edge devices (NVIDIA Jetson, Raspberry Pi).
- Độ chính xác: cải thiện mAP (Mean Average Precision).
- Hỗ trợ đa nhiệm: detection, segmentation, pose estimation trong cùng framework.
- Tích hợp huấn luyện nhanh chóng: dễ dàng fine-tune với dữ liệu giao thông.

YOLO11 đánh dấu một chương mới cho gia đình YOLO, cung cấp một mô hình có khả năng và linh hoạt hơn, đưa tầm nhìn máy tính lên một tầm cao mới. Với kiến trúc tinh tế và khả năng nâng cao, mô hình hỗ trợ các tác vụ tầm nhìn máy tính như ước tính tư thế và phân đoạn trường hợp mà cộng đồng Vision AI yêu thích ở Ultralytics YOLOv8 , nhưng với hiệu suất và độ chính xác thậm chí còn cao hơn. Glenn Jocher , Nhà sáng lập kiêm Tổng giám đốc điều hành của Ultralytics, chia sẻ: “Với YOLO11, chúng tôi đặt mục tiêu phát triển một mô hình vừa mạnh mẽ vừa thiết thực cho các ứng dụng trong thế giới thực. Hiệu quả và độ chính xác được cải thiện của nó khiến nó trở thành một công cụ mạnh mẽ có thể thích ứng với những thách thức độc đáo mà nhiều ngành công nghiệp khác nhau phải đối mặt. Tôi rất mong chờ xem cộng đồng Vision AI sử dụng YOLO11 để tạo ra các giải pháp sáng tạo và đưa tầm nhìn máy tính lên một tầm cao mới như thế nào”.

Các tính năng chính của YOLO11 :

- Cải thiện trích xuất đặc trưng: YOLO11 sử dụng backbone và kiến trúc neck được cải tiến, giúp tăng cường khả năng trích xuất đặc trưng để phát hiện đối tượng chính xác hơn và thực hiện các tác vụ phức tạp.

- Tối Ưu Hóa cho Hiệu Quả và Tốc Độ: YOLO11 giới thiệu các thiết kế kiến trúc tinh tế và quy trình huấn luyện được tối ưu hóa, mang lại tốc độ xử lý nhanh hơn và duy trì sự cân bằng tối ưu giữa độ chính xác và hiệu suất.
- Độ Chính Xác Cao Hơn với Ít Tham Số Hơn: Với những tiến bộ trong thiết kế mô hình, YOLO11m đạt được độ chính xác trung bình (mAP) cao hơn trên bộ dữ liệu COCO trong khi sử dụng ít hơn 22% tham số so với YOLOv8m, giúp nó đạt hiệu quả tính toán mà không ảnh hưởng đến độ chính xác.
- Khả Năng Thích Ứng Cao Trong Mọi Môi Trường: YOLO11 có thể được triển khai liền mạch trên nhiều môi trường khác nhau, bao gồm các thiết bị biên, nền tảng đám mây và các hệ thống hỗ trợ NVIDIA GPU, đảm bảo tính linh hoạt tối đa.
- Hỗ Trợ Nhiều Tác Vụ Đa Dạng: Cho dù đó là phát hiện đối tượng, phân vùng thể hiện, phân loại hình ảnh, ước tính tư thế hoặc phát hiện đối tượng theo hướng (OBB), YOLO11 được thiết kế để đáp ứng một loạt các thách thức về thị giác máy tính.

2.2.1 Cấu trúc của YOLO:

Mô hình YOLO có cấu trúc tổng thể gồm ba phần chính :Backbone, Neck, và Head, mỗi phần đảm nhận một vai trò cụ thể trong việc xử lý hình ảnh và đưa ra dự đoán.

a). Backbone (Mạng xương sống)

Backbone là thành phần đầu tiên và quan trọng nhất trong cấu trúc của YOLO. Nó đóng vai trò như bộ não của mô hình, có nhiệm vụ chính là trích xuất các đặc trưng hình ảnh từ đầu vào. Hãy tưởng tượng Backbone như một chuỗi các "mắt xích" có khả năng quan sát và phân tích hình ảnh ở nhiều cấp độ khác nhau.

Chức năng chính:

- Trích xuất đặc trưng: Khi một hình ảnh được đưa vào, Backbone sẽ sử dụng một mạng nơ-ron tích chập (Convolutional Neural Network - CNN) để tạo ra các bản đồ đặc trưng (feature maps) có kích thước giảm dần. Các lớp đầu tiên sẽ thu thập các đặc trưng cơ bản như cạnh, góc, màu sắc, trong khi các lớp sâu hơn sẽ nhận diện các đặc trưng phức tạp hơn như hình dạng, kết cấu của đối tượng.
- Tạo ra nhiều bản đồ đặc trưng: Backbone không chỉ tạo ra một bản đồ đặc trưng duy nhất. Thay vào đó, nó tạo ra nhiều bản đồ đặc trưng ở các kích thước khác nhau. Các bản đồ có kích thước lớn chứa thông tin chi tiết về các đối tượng nhỏ, trong khi các bản đồ có kích thước nhỏ lại chứa thông tin ngữ cảnh tổng thể, rất hữu ích cho việc phát hiện các đối tượng lớn.

Cấu trúc chi tiết: Backbone của các phiên bản YOLO hiện đại (như YOLOv8) thường sử dụng các kiến trúc mạng hiệu quả như CSPDarknet hoặc các biến thể của nó. Kiến trúc này giúp giảm thiểu việc tính toán lặp lại và tăng cường hiệu suất của mô hình.

b) Neck (Phần cổ)

Neck nằm giữa Backbone và Head. Đây là "cầu nối" giúp kết hợp các thông tin đặc trưng từ Backbone, làm cho quá trình phát hiện đối tượng trở nên chính xác hơn, đặc biệt là với các đối tượng có kích thước đa dạng.

Chức năng chính:

- Tăng cường thông tin: Neck nhận các bản đồ đặc trưng ở nhiều kích thước khác nhau từ Backbone. Thay vì chỉ sử dụng một bản đồ đặc trưng, Neck kết hợp chúng lại để tạo ra các bản đồ đặc trưng mới, giàu thông tin hơn.
- Giải quyết bài toán đa tỷ lệ: Đây là vai trò quan trọng nhất của Neck. Bằng cách kết hợp thông tin từ các lớp sâu (đặc trưng ngữ cảnh) với các

lớp nông (đặc trưng chi tiết), Neck giúp mô hình phát hiện tốt các đối tượng có kích thước nhỏ (ví dụ: một chiếc xe ở xa) và cả các đối tượng lớn (một người đi bộ ở gần).

Cấu trúc chi tiết:

- Các kiến trúc phổ biến được sử dụng trong Neck bao gồm: FPN (Feature Pyramid Network) và PAN (Path Aggregation Network).
- FPN: Xây dựng một kim tự tháp đặc trưng từ trên xuống (top-down), truyền các đặc trưng ngữ cảnh từ các lớp sâu xuống các lớp nông.
- PAN: Bổ sung thêm một đường dẫn từ dưới lên (bottom-up), truyền các đặc trưng chi tiết từ các lớp nông lên các lớp sâu. Sự kết hợp giữa FPN và PAN giúp mô hình có khả năng tổng hợp thông tin một cách toàn diện hơn, cải thiện đáng kể hiệu suất phát hiện.

Lưu ý : FPN và PANet giúp YOLO xử lý được nhiều kích thước và kiểu vật thể khác nhau, từ đó nâng cao độ chính xác trong các bài toán nhận diện phức tạp.

c) Head (Phần đầu)

Head là thành phần cuối cùng của mô hình, chịu trách nhiệm đưa ra dự đoán cuối cùng dựa trên các đặc trưng đã được Backbone và Neck xử lý.

Chức năng chính dự đoán hộp giới hạn và nhãn:

- Head sẽ xử lý các bản đồ đặc trưng được truyền từ Neck và đưa ra các dự đoán về vị trí, kích thước của các hộp giới hạn (bounding boxes) và xác suất của từng lớp đối tượng (người đi bộ, ô tô, biển báo).
- Tạo ra các dự đoán: Đối với mỗi ô lưới trong các bản đồ đặc trưng, Head sẽ dự đoán một tập hợp các hộp giới hạn tiềm năng. Mỗi hộp giới hạn này sẽ đi kèm với:
  - Tọa độ: Vị trí (x, y) và kích thước (chiều rộng, chiều cao) của hộp giới hạn.

• Độ tin cậy của đối tượng: Một giá trị từ 0 đến 1, cho biết khả năng có một đối tượng nằm trong hộp này.

• Xác suất lớp: Một tập hợp các giá trị xác suất cho mỗi lớp đối tượng mà mô hình đã được huấn luyện.

- Cấu trúc chi tiết: Các phiên bản YOLO gần đây như YOLOv8 đã cải tiến Head để việc dự đoán trở nên hiệu quả hơn. Thay vì sử dụng một Head duy nhất, mô hình sử dụng nhiều Head để xử lý các đặc trưng ở các kích thước khác nhau, giúp việc phát hiện đối tượng chính xác hơn. Head của YOLOv8 cũng có sự phân tách giữa nhánh dự đoán vị trí và nhánh dự đoán lớp, làm cho mô hình hoạt động ổn định và chính xác hơn.

2.2.2. Phát hiện đối tượng bằng mô hình đã được huấn luyện (Inference)

Sau khi mô hình học sâu như YOLO đã hoàn thành quá trình huấn luyện chuyên sâu trên một bộ dữ liệu khổng lồ và đa dạng, nó trở thành một "chuyên gia" có khả năng nhận diện các đối tượng mà nó đã được đào tạo (ví dụ: người đi bộ, ô tô, biển báo giao thông). Giai đoạn này, thường được gọi là inference (suy luận), là lúc mô hình được đưa vào ứng dụng thực tế để xử lý dữ liệu mới.

Quy trình hoạt động chi tiết:

- Toàn bộ quá trình này diễn ra với tốc độ cực kỳ cao, cho phép hệ thống phản ứng kịp thời với môi trường thay đổi liên tục. Khi một luồng hình ảnh video thời gian thực từ camera của xe được đưa vào, quy trình diễn ra
- Tạo ra hàng loạt dự đoán thô (Raw Predictions): mô hình sẽ xử lý từng khung hình một (frame-by-frame). Mỗi khung hình được chia thành một mạng lưới ô vuông (grid cells). Với mỗi ô vuông, mô hình sẽ tạo ra một loạt các dự đoán tiềm năng.
- Một dự đoán bao gồm nhiều thông tin:

- Tọa độ và kích thước của hộp giới hạn: Vị trí (x, y) và kích thước (chiều rộng, chiều cao) của hộp bao quanh đối tượng.
- Điểm tin cậy (Objectness Score): Một giá trị từ 0 đến 1, cho biết mức độ tự tin của mô hình rằng có một đối tượng nằm bên trong hộp này.
- Xác suất lớp (Class Probabilities): Một tập hợp các giá trị xác suất cho mỗi lớp đối tượng đã được huấn luyện. Ví dụ, mô hình có thể dự đoán hộp này có 95% là "ô tô" và 3% là "người đi bộ".

Loại bỏ các dự đoán yếu (Filtering Low-Confidence Boxes): hàng nghìn dự đoán thô được tạo ra cho mỗi khung hình. Nhiều trong số đó có điểm tin cậy rất thấp và không chứa đối tượng thực sự.

Mô hình sẽ tự động sàng lọc và loại bỏ tất cả các hộp giới hạn có điểm tin cậy thấp hơn một ngưỡng đã được định trước. Bước này giúp giảm đáng kể số lượng hộp giới hạn cần xử lý tiếp theo, làm tăng hiệu quả tính toán.

Sử dụng thuật toán Non-Maximum Suppression để chọn ra dự đoán tốt nhất:

- Đây là một bước hậu xử lý quan trọng. Sau khi lọc các dự đoán yếu, vẫn có thể có nhiều hộp giới hạn chồng chéo nhau cùng chỉ vào một đối tượng duy nhất.
- Thuật toán NMS hoạt động theo các bước:
    - Bước 1: Chọn hộp giới hạn có điểm tin cậy cao nhất trong số tất cả các hộp còn lại.
    - Bước 2: So sánh hộp đã chọn với tất cả các hộp còn lại. Bất kỳ hộp nào có độ chồng lấn (Intersection over Union - IoU) lớn hơn một ngưỡng nhất định (ví dụ: 0.5) với hộp đã chọn sẽ bị loại bỏ.

- Bước 3: Lặp lại quy trình này cho đến khi không còn hộp giới hạn nào có độ chồng lấn cao, đảm bảo rằng mỗi đối tượng chỉ có một hộp giới hạn duy nhất.

Hiệu suất thời gian thực:

- Toàn bộ quy trình này, từ khi khung hình được đưa vào đến khi kết quả cuối cùng được xuất ra, chỉ mất vài mili giây.
- Tốc độ này được đo bằng FPS (Frames Per Second - Khung hình trên giây). Một mô hình có FPS cao cho thấy nó có khả năng xử lý nhanh, điều này là vô cùng quan trọng đối với xe tự lái để có thể phản ứng ngay lập tức với các tình huống giao thông nguy hiểm.

Đưa ra cảnh báo và gửi dữ liệu để hỗ trợ đưa ra quyết định: khi quá trình phát hiện đã hoàn tất, thông tin thu được sẽ không chỉ dừng lại ở màn hình hiển thị mà còn trở thành dữ liệu đầu vào then chốt cho các hệ thống điều khiển khác của xe.

Cảnh báo trực quan (Visual Alerts):

- Để người lái xe (hoặc hệ thống giám sát) dễ dàng nhận biết, kết quả phát hiện sẽ được hiển thị trực quan.
- Các hộp giới hạn (bounding boxes) sẽ được vẽ trực tiếp lên hình ảnh video, bao quanh các đối tượng đã được nhận diện.
- Mỗi loại đối tượng có thể được gán một màu sắc hoặc kiểu hiển thị riêng biệt để dễ phân biệt, ví dụ:
    - Hộp giới hạn màu đỏ cho người đi bộ.
    - Hộp giới hạn màu xanh lá cây cho ô tô.
    - Hộp giới hạn màu vàng cho biển báo.

Gửi dữ liệu và tích hợp hệ thống (Data Transmission and System Integration):

- Đây là phần quan trọng nhất, nơi hệ thống nhận thức (perception system) giao tiếp với các module điều khiển khác của xe tự lái. Dữ liệu được truyền đi không chỉ là hình ảnh, mà là thông tin số hóa có cấu trúc và ý nghĩa.

- Thông tin chi tiết như tọa độ của hộp giới hạn, loại đối tượng, và điểm tin cậy sẽ được đóng gói thành các gói dữ liệu và gửi đến các hệ thống khác của xe, bao gồm:

- Hệ thống điều khiển phanh/ga: Nếu mô hình phát hiện một người đi bộ đang băng qua đường ở khoảng cách gần, dữ liệu này sẽ được gửi đến hệ thống điều khiển phanh để xe tự động giảm tốc độ hoặc dừng lại, tránh va chạm.
- Hệ thống lái (Steering system): Nếu phát hiện vật cản ở phía trước và không thể phanh kịp, dữ liệu sẽ được sử dụng để tính toán và thực hiện chuyển hướng lái an toàn.
- Hệ thống cảnh báo âm thanh: Đưa ra cảnh báo bằng âm thanh cho người lái xe khi phát hiện một mối nguy hiểm tiềm tàng.

Tóm lại, quá trình này biến thông tin thị giác thành dữ liệu hành động, cho phép xe tự lái không chỉ "nhìn thấy" mà còn "hiểu" và "phản ứng" một cách tự động và an toàn với môi trường xung quanh.

2.2.3. Đưa ra cảnh báo và gửi dữ liệu để hỗ trợ đưa ra quyết định

Sau khi mô hình phát hiện đối tượng (YOLO11) đã xác định chính xác vị trí (bounding box) và loại đối tượng (class), thông tin đầu ra không chỉ dừng lại ở việc hiển thị cho người quan sát mà còn trở thành dữ liệu đầu vào cốt lõi cho toàn bộ hệ thống xe tự lái. Quá trình này có thể chia thành hai hướng chính:

1. Cảnh báo trực quan (Visual Alerts)

a) Hiển thị trên giao diện người dùng

- Kết quả phát hiện (hộp giới hạn + nhãn đối tượng) được truyền đến hệ thống hiển thị của xe, bao gồm:

+ Màn hình điều khiển trung tâm (Central Display).

+ Màn hình HUD (Head-Up Display) chiếu trực tiếp lên kính lái.

+ Màn hình phụ trợ cho kỹ sư/giám sát trong xe thử nghiệm.

b) Biểu diễn trực quan hóa theo màu sắc/kiểu hiển thị

- Người đi bộ (Pedestrian): Hộp giới hạn, ưu tiên cao nhất vì liên quan trực tiếp đến an toàn.
- Phương tiện (Car, Motorbike, Truck…): Hộp giới hạn, biểu thị đối tượng động nhưng có thể dự đoán quỹ đạo.
- Biển báo giao thông (Traffic Signs): Hộp giới hạn, giúp xe tuân thủ luật giao thông.

c) Thông tin chi tiết đi kèm

- Nhãn (Label): ví dụ "Pedestrian".
- Điểm tin cậy : hiển thị theo %, ví dụ "Pedestrian – 92%".
- Khoảng cách tương đối : có thể bổ sung nếu tích hợp với Lidar/Radar.

d) Cảnh báo bổ sung

Ngoài hình ảnh, hệ thống có thể kết hợp:

- Rung vô-lăng hoặc ghế lái: Nhắc nhở người khiển trong tình huống khẩn cấp.
- Màu sắc cảnh báo động: Bounding box có thể nhấp nháy khi độ nguy hiểm tăng cao.

Việc này đảm bảo rằng ngay cả trong tình huống tài xế mất tập trung, hệ thống vẫn có thể thu hút sự chú ý và kích hoạt phản xạ phòng vệ kịp thời.

2. Gửi dữ liệu và tích hợp hệ thống

Đây là phần then chốt, nơi kết quả từ YOLO11 không chỉ dừng ở mức hình ảnh mà được chuyển hóa thành dữ liệu có cấu trúc để giao tiếp với các hệ thống điều khiển khác trong xe.

a) Dữ liệu truyền đi bao gồm:

- Bounding box: tọa độ cho từng đối tượng.
- Loại đối tượng (Class): Pedestrian, Car, Truck, Sign, Animal…
- Confidence score: mức độ chắc chắn của dự đoán (0–100%).
- Khoảng cách và vận tốc tương đối: được tính toán khi tích hợp với cảm biến Radar/Lidar hoặc hệ thống tracking .
- Thời gian phát hiện (timestamp): đồng bộ với hệ thống điều khiển để xử lý trong thời gian thực.

b) Quy trình tích hợp hệ thống.

- Mô hình YOLO11 hỗ trợ trích xuất dữ liệu phát hiện.
- perception (nhận thức) hỗ trợ đóng gói dữ liệu dưới dạng thông điệp .
- Bus truyền thông trong xe hỗ trợ phân phối dữ liệu đến các module khác.
- Module điều khiển (control system) hỗ trợ ra quyết định hành động.

c) Ứng dụng trong các hệ thống điều khiển:

- Hệ thống phanh/ga (Brake/Throttle):
  - Nếu phát hiện người đi bộ băng qua đường ở khoảng cách gần, dữ liệu sẽ kích hoạt Emergency Brake Assist để dừng khẩn cấp.
  - Nếu đối tượng chỉ mới xuất hiện xa phía trước, hệ thống có thể giảm tốc từ từ để đảm bảo an toàn.

- Hệ thống lái (Steering System):

  - Nếu có vật cản phía trước (xe tải đỗ, xe máy đi chậm), dữ liệu phát hiện sẽ được gửi đến Path Planning để tính toán quỹ đạo thay thế.

- Xe có thể chuyển làn hoặc lái vòng tránh mà vẫn đảm bảo giữ khoảng cách an toàn.

- Hệ thống hỗ trợ ra quyết định tổng thể (Decision-Making System):

- Tích hợp dữ liệu từ YOLO11 với HD Map, GPS, Radar, Lidar.
- Kết hợp thuật toán Behavior Planning để đưa ra hành động tối ưu: tiếp tục di chuyển, dừng lại, đổi làn, hoặc quay đầu.
- Ưu tiên an toàn nhưng vẫn đảm hiệu quả khi vận hành.

d) Ý nghĩa

Nhờ việc truyền dữ liệu có cấu trúc thay vì chỉ hiển thị hình ảnh, hệ thống xe tự lái:

- Tính chủ động: Xe không chỉ "nhìn thấy" mà còn "hiểu" tình huống.

- Tính kịp thời: Phản ứng trong thời gian thực (tính bằng mili-giây).

- Tính an toàn: Giảm thiểu rủi ro va chạm, bảo vệ hành khách và người tham gia giao thông khác.

- Tính mở rộng: Dữ liệu có thể được lưu trữ và phân tích để huấn luyện, tối ưu mô hình trong tương lai.

CHƯƠNG 3: CÀI ĐẶT, THỬ NGHIỆM VÀ ĐÁNH GIÁ

Chương 3 tập trung vào việc cài đặt, thử nghiệm và đánh giá mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái. Chương mở đầu bằng việc giới thiệu môi trường thử nghiệm, bao gồm Python, Visual Studio Code, Google Colab và Android Studio, nhằm cung cấp nền tảng cho việc phát triển và huấn luyện mô hình. Tiếp theo, chương trình bày cách tạo bộ dữ liệu học để huấn luyện mô hình, cũng như các bước đào tạo mô hình với dữ liệu đã chuẩn bị. Chương cũng giới thiệu giao diện ứng dụng, kết quả thử nghiệm trên mô hình đã huấn luyện và các biểu đồ minh họa hiệu suất mô hình. Phần cuối cùng của chương tập trung vào đánh giá thực tế và đánh giá mô hình, bao gồm phân tích tỉ lệ lỗi, ưu nhược điểm của mô hình, cùng các đề xuất cải tiến để nâng cao độ chính xác và tính khả dụng trong ứng dụng xe tự lái. Nhìn chung, Chương 3 cung cấp cái nhìn tổng quan về triển khai thực nghiệm, từ việc cài đặt đến đánh giá, giúp người đọc hiểu rõ hiệu quả và giới hạn của mô hình được xây dựng.

3.1.Môi trường thử nghiệm

3.1.1. Python

a. Khái niệm

Python là một ngôn ngữ lập trình bậc cao cho các mục đích lập trình đa năng, do Guido van Rossum tạo ra và lần đầu ra mắt vào năm 1991. Python được thiết kế với ưu điểm mạnh là dễ đọc, dễ học và dễ nhớ. Python là ngôn ngữ có hình thức rất sáng sủa, cấu trúc rõ ràng, thuận tiện cho người mới học lập trình và là ngôn ngữ lập trình dễ học, được dùng rộng rãi trong phát triển trí tuệ nhân tạo. Cấu trúc của Python còn cho phép người sử dụng viết mã lệnh với số lần gõ phím tối thiểu. Vào tháng 7 năm 2018, Van Rossum đã từ chức lãnh đạo trong cộng đồng ngôn ngữ Python sau 30 năm làm việc.

Hình 3.1. Hình ảnh về Python

Python hoàn toàn tạo kiểu động và dùng cơ chế cấp phát bộ nhớ tự động; do vậy nó tương tự như Perl, Ruby, Scheme, Smalltalk, và Tcl. Python được phát triển trong một dự án mã nguồn mở, do tổ chức phi lợi nhuận Python Software Foundation quản lý.

Ban đầu, Python được phát triển để chạy trên nền Unix. Nhưng rồi theo thời gian, Python dần mở rộng sang mọi hệ điều hành từ MS-DOS đến Mac OS, OS/2, Windows, Linux và các hệ điều hành khác thuộc họ Unix. Mặc dù sự phát triển của Python có sự đóng góp của rất nhiều cá nhân, nhưng Guido van Rossum hiện nay vẫn là tác giả chủ yếu của Python. Ông giữ vai trò chủ chốt trong việc quyết định hướng phát triển của Python.

Python luôn được xếp hạng vào những ngôn ngữ lập trình phổ biến nhất.

b. Đặc tính.

Python đang trở nên phổ biến trong cộng đồng lập trình nhờ các đặc tính sau:

- Ngôn ngữ thông dịch: Python được xử lý trong thời gian chạy bởi trình thông dịch Python.

- Ngôn ngữ hướng đối tượng: Nó hỗ trợ các đặc trưng và kỹ thuật lập trình hướng đối tượng.

- Ngôn ngữ lập trình tương tác: Người dùng có thể tương tác trực tiếp với trình thông dịch python để viết chương trình.

- Ngôn ngữ dễ học: Python rất dễ học, đặc biệt là cho người mới bắt đầu.

- Cú pháp đơn giản: Việc hình thành cú pháp Python rất đơn giản và dễ hiểu, điều này cũng làm cho nó trở nên phổ biến.

- Mã nguồn mở: Python cho phép người dùng có thể sử dụng, chỉnh sửa và phân phối mã nguồn một cách tự do.

- Di động: Mã Python có thể chạy trên nhiều nền tảng phần cứng có cùng giao diện.

- Có thể mở rộng: Người dùng có thể thêm các mô-đun cấp thấp vào trình thông dịch Python.

- Có thể cải tiến: Python cung cấp một cấu trúc cải tiến để hỗ trợ các chương trình lớn sau đó là shell-script.

- Hỗ trợ cộng đồng: python có một cộng đồng lớn với nhiều người dùng và nhà phát triển trên khắp thế giới đóng góp vào việc phát triển cà cải tiến Python.

Cộng đồng này cũng cung cấp hỗ trợ và tài liệu để giúp người dùng python giải quyết cá vấn đề một cách nhanh chóng và hiệu quả

c. Các phiên bản của Python

- Python đã được Guido van Rossum tạo ra vào những năm 1980 tại Trung tâm Toán học – Tin học (Centrum Wiskunde & Informatica, CWI) ở Hà Lan như là một ngôn ngữ kế tục ngôn ngữ ABC – một ngôn ngữ được lấy cảm hứng từ SETL (ngôn ngữ được phát triển bởi Jacob T. Schwartz và các đồng nghiệp), có khả năng xử lý ngoại lệ và giao tiếp với hệ điều hành Amoeba. Nó bắt đầu được triển khai vào tháng 12 năm 1989.

- Python 2.0 được ra mắt vào ngày 16 tháng 10 năm 2000, với nhiều đặc trưng mới mẻ, bao gồm một bộ dọn rác phát hiện theo chu kỳ và khả năng hỗ trợ Unicode.

- Python 3.0 được ra mắt vào ngày mùng 3 tháng 12 năm 2008. Đây là một phiên bản lớn của Python không tương thích ngược hoàn toàn. Nhiều đặc trưng

lớn của nó đã được chuyển mã ngược (backport) về loạt phiên bản Python 2.6.x và 2.7.x. Các bản phát hành của Python 3 có đi kèm với công cụ 2to3, có tác dụng tự động hoá việc dịch mã Python 2 sang Python 3.

- Python 3.9.2 và 3.8.8 được xúc tiến vì tất cả các phiên bản trước của Python (bao gồm cả 2.7) gặp một số vấn đề bảo mật, có thể dẫn đến thực thị mã từ xa và "đầu độc" bộ nhớ đệm.

- Trong năm 2022, Python 3.10.4 và 3.9.12 được xúc tiến cùng với 3.8.13 và 3.7.13, nguyên nhân là do một vài vấn đề về bảo mật. Khi Python 3.9.13 được phát hành vào tháng Năm năm 2022, loạt phiên bản 3.9 (cùng với loạt 3.8 và 3.7) được thông báo rằng sẽ chỉ nhận được các bản vá bảo mật trong tương lai. Vào ngày 7 tháng Chín năm 2022, bốn bản cập nhật mới được phát hành do có khả năng xảy ra một cuộc tấn công từ chối dịch vụ: 3.10.7, 3.9.14, 3.8.14 và 3.7.14.

- Tính đến tháng 10 năm 2023, Python 3.12 là bản phát hành ổn định mới nhất. Một số thay đổi đáng chú ý từ bản 3.11 bao gồm các thay đổi về ngôn ngữ và thư viện chuẩn.

3.1.2. Visual Studio Code

a. Khái niệm

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn miễn phí, nhẹ và mạnh mẽ, phát triển bởi Microsoft, hoạt động trên đa nền tảng (Windows, macOS, Linux). Nó hỗ trợ nhiều ngôn ngữ lập trình và cung cấp các tính năng hữu ích như gỡ lỗi (debug), tô sáng cú pháp (syntax highlighting), tự động hoàn thành mã, và tích hợp Git. VS Code được ưa chuộng vì khả năng tùy biến cao và kho tiện ích mở rộng phong phú, phù hợp cho cả người mới bắt đầu và lập trình viên chuyên nghiệp. Nó là đa nền tảng, hoạt động trên Microsoft Windows, macOS và Linux.

Hình 3.2.hình ảnh về VS Code

b. Các đặc trưng

Visual Studio Code là gì được rất nhiều người tìm hiểu. Đây cũng là một trong các ứng dụng được dân IT "săn đón" và tải về và sử dụng rất nhiều. Visual Studio Code cũng luôn có những cải tiến và tạo ra đa dạng các tiện ích đi kèm từ đó giúp cho các lập trình viên sử dụng dễ dàng hơn. Trong đó có thể kể đến những ưu điểm sau:

Đa dạng ngôn ngữ lập trình giúp người dùng thỏa sức sáng tạo và sử dụng như HTML, CSS, JavaScript, C++,…

Ngôn ngữ, giao diện tối giản, thân thiện, giúp các lập trình viên dễ dàng định hình nội dung.

Các tiện ích mở rộng rất đa dạng và phong phú.

Không phải ngẫu nhiên mà Visual Studio Code được các lập trình viên ưa chuộng sử dụng. Visual Studio Code mang rất nhiều ưu điểm vượt trội so với bất kỳ IDE nào khác:

- Hỗ trợ đa nền tảng: Linux, Mac, Windows,..
- Hỗ trợ đa ngôn ngữ: C/C++, C#, F#, JavaScript, JSON,…
- Ít dung lượng
- Tính năng mạnh mẽ
- Intellisense chuyên nghiệp
- Giao diện thân thiện

- Kiến trúc mạnh mẽ và người dùng có thể khai thác mở rộng
- Số lượng người sử dụng lớn tạo nên ộng đồng hỗ trợ rộng rãi

3.1.3. Google Collab

a. Khái niệm

Colaboratory hay còn gọi là Google Colab, là một sản phẩm từ Google Re search, nó cho phép chạy các dòng code python thông qua trình duyệt, đặc biệt phù hợp với Data analysis, machine learning và giáo dục. Colab không cần yêu cầu cài đặt hay cấu hình máy tính, mọi thứ có thể chạy thông qua trình duyệt, bạn có thể sử dụng tài nguyên máy tính từ CPU tốc độ cao và cả GPUs và cả TPUs đều được cung cấp cho bạn.

Hình 3.3.hình ảnh về GG Colab

Colab cung cấp nhiều loại GPU, thường là Nvidia K80s, T4s, P4s and P100s, tuy nhiên người dùng không thể chọn loại GPU trong Colab, GPU trong Colab thay đổi theo thời gian. Vì là dịch vụ miễn phí, nên Colab sẽ có những thứ tự ưu tiên trong việc sử dụng tài nguyên hệ thống, cũng như giới hạn thời gian sử dụng, thời gian sử dụng tối đa lên tới 12 giờ.

b. Những đặc trưng của Google Colab

- Sử dụng Jupyter Notebooks trực tuyến

Google Colab cho phép tạo và chạy các Jupyter Notebooks trực tuyến mà không cần cài đặt môi trường phát triển phức tạp trên máy tính cá nhân.

Giao diện sử dụng tương tự như Jupyter Notebook truyền thống với các cell cho phép thực thi mã Python hoặc viết markdown để tạo nội dung hướng dẫn.

- Khả năng chia sẻ và cộng tác

Người dùng có thể chia sẻ notebook với những người khác để cùng làm việc trên cùng một notebook, tạo điều kiện thuận lợi cho việc học tập và làm việc nhóm.

Các đặc trưng như bình luận và chế độ chỉnh sửa đồng thời giúp tăng tính tương tác và hiệu quả của quá trình cộng tác.

- Dùng GPU và TPU miễn phí

Google Colab cung cấp truy cập miễn phí đến GPU và TPU, đặc biệt hữu ích cho các tác vụ tính toán nặng về mặt số học, đặc biệt là trong lĩnh vực học máy và deep learning.

Việc sử dụng các card GPU hoặc TPU có sẵn giúp tăng tốc độ xử lý và huấn luyện mô hình so với việc sử dụng CPU thông thường.

- Lưu trữ dữ liệu trên Google Drive và tích hợp Google Cloud

Người dùng có thể truy cập và lưu trữ dữ liệu trực tiếp từ Google Drive, tạo điều kiện thuận lợi cho việc làm việc với tập tin dữ liệu lớn.

Tích hợp với điện toán đám mây của Google cũng cho phép sử dụng các dịch vụ như BigQuery, Cloud Storage, và các API khác từ dịch vụ điện toán đám mây trong quá trình làm việc.

3.1.4 Android Studio

a) Khái niệm

Android Studio là môi trường phát triển tích hợp chính thức do Google phát triển, dùng để xây dựng các ứng dụng chạy trên hệ điều hành Android. Android Studio được xây dựng dựa trên IntelliJ IDEA của JetBrains, hỗ trợ mạnh mẽ cho việc lập trình với các ngôn ngữ như Java, Kotlin và C++.

Nó cung cấp đầy đủ các công cụ phục vụ cho quá trình phát triển ứng dụng, từ việc thiết kế giao diện người dùng (UI), viết mã, biên dịch, mô phỏng thiết bị ảo (Android Emulator) cho đến đóng gói và triển khai (APK) lên điện thoại thực tế.

b)Các thành phần chính của Android Studio.

- Code Editor: Cung cấp môi trường viết mã thông minh với gợi ý cú pháp kiểm tra lỗi
- Layout Editor: Giao diện kéo thả giúp thiết kế màn hình ứng dụng trực quan bằng XML hoặc trực tiếp trên màn mô phỏng
- AVD Manager: Cho phép tạo và chạy các thiết bị ảo (giả lập điện thoại, máy tính bảng, xe hơi...) để kiểm thử ứng dụng mà không cần thiết bị thật.
- Logcat & Debugger: Hiển thị log hệ thống, giúp lập trình viên kiểm tra lỗi và theo dõi hoạt động của ứng dụng trong thời gian chạy.
- Gradle Build System: Hệ thống quản lý build mạnh mẽ, tự động hóa việc biên dịch, kiểm thử và đóng gói ứng dụng.
- SDK Manager: Dùng để tải về các phiên bản Android SDK, thư viện , API cần thiết cho từng thiết bị hoặc dự án.

c) Ứng dụng Android Studio trong đề tài.

Trong đề tài "Nhận diện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái", Android Studio được sử dụng để:

- Xây dựng ứng dụng điều khiển và hiển thị kết quả nhận diện (hiển thị vật cản, loại biển báo, cảnh báo âm thanh hoặc thông tin ra quyết định).

- Kết nối với mô hình AI chạy trên Python/YOLO thông qua giao thức socket / REST API.

- Hiển thị giao diện người dùng (GUI) trực quan, thân thiện, giúp người lái hoặc hệ thống theo dõi trạng thái của xe.

3.2. Tạo bộ dữ liệu học để huấn luyện

Roboflow là một framework dành cho nhà phát triển Thị giác Máy tính để thu thập dữ liệu tốt hơn để xử lý trước và các kỹ thuật đào tạo mô hình. Roboflow có sẵn các tập dữ liệu công khai cho người dùng và cũng có quyền truy cập để người dùng tải lên dữ liệu tùy chỉnh của riêng họ. Roboflow chấp nhận các định dạng chú thích khác nhau. Trong quá trình xử lý trước dữ liệu, có các bước liên quan như định hướng hình ảnh, thay đổi kích thước, độ tương phản và tăng cường dữ liệu.

Toàn bộ quy trình làm việc có thể được điều phối với các nhóm trong khuôn khổ. Đối với đào tạo mô hình, có một loạt thư viện mô hình đã có mặt như EfficientNet, MobileNet, Yolo, TensorFlow, PyTorch, v.v. Sau đó, các tùy chọn trực quan và triển khai mô hình cũng có sẵn do đó bao gồm toàn bộ hiện đại.

Roboflow được sử dụng trong các ngành công nghiệp thị giác máy tính khác nhau cho các trường hợp sử dụng như phát hiện rò rỉ khí đốt, phát hiện thực vật và cỏ dại, bảo trì máy bay, ước tính thiệt hại mái nhà, hình ảnh vệ tinh, ô tô tự lái, bộ đếm giao thông, dọn rác và nhiều hơn nữa.

Dưới đây là số lượng ảnh em đã thu thập được trong thời gian thực tập để tạo bộ huấn luyện cho đề tài

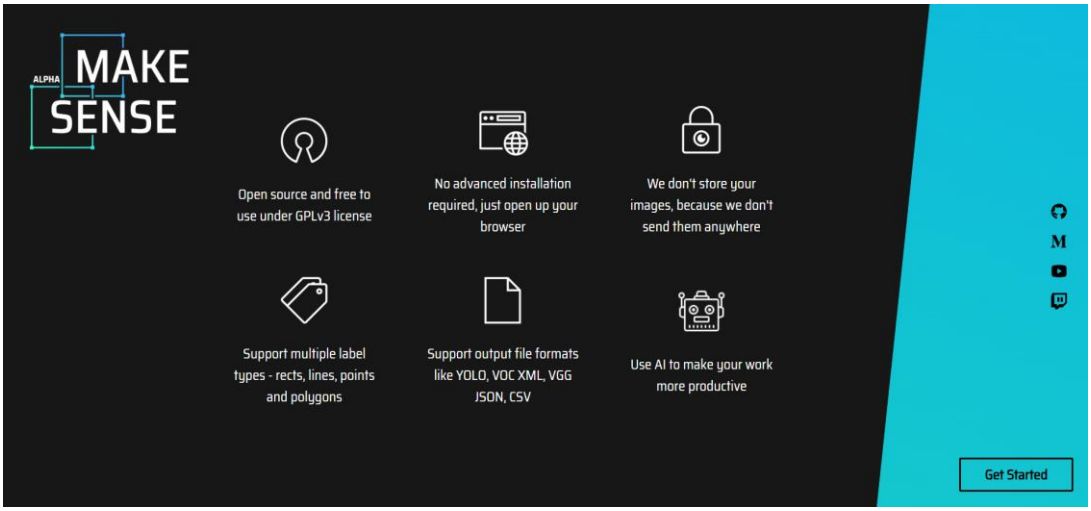
Tổng số lượng ảnh: 147 ảnh

Đối tượng	Số ảnh từ Roboflow	Số ảnh thu thập thực tế
Người đi bộ	51	19
Vật cản	37	10
Biển báo	13	10

Bảng 3.1.Số Lượng Ảnh của từng đối tượng

Với tổng số 1 ảnh trong đó ảnh được thu thập thủ công trên các trang mạng xã hộivà một số ảnh được thu thập từ việc chụp trực tiếp thực tế.

Sau khi có đủ hình ảnh về tất cả các đối tượng , bước tiếp theo sử dụng công cụ có sẵn trên mạng là makesense để gán nhãn cho đối tượng, trong bài này đối tượng sẽ là người đi bộ, vật cản , biển báo.



Hình 3.4.Giao diện makesensi dùng để gán nhán đối tượng

3.3. Đào tạo mô hình "Xây dụng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái"

Để huấn luyện mô hình "Xây dụng mô hình phát hiện vật cản và biển báo" bằng phương pháp học sâu thì em sử dụng một công cụ hỗ trợ ở đây là Google Colab. Google Colab là một dịch vụ máy tính đám mây của Google cho phép người dùng tạo ra các tệp note book Jupyter để thực thi mã Python. Nó cung cấp một môi trường tính toán đám mây miễn phí, với các đặc trưng như GPU miễn phí, RAM miễn phí đến 12GB có thể mở rộng đến 25.5GB nếu có trả phí, lưu trữ đám mây và nhiều đặc trưng khác.

Ngoài ta Google Colab còn liên kết với driver để có thể hộ trợ người dùng trong việc lưu trữ các dữ liệu quan trọng. Google Colab được thiết kế để hỗ trợ việc phát triển và huấn luyện các mô hình học máy và các ứng dụng AI. Nó cung cấp một môi trường tính toán đám mây miễn phí, mạnh mẽ và linh hoạt cho các nhu cầu học tập và nghiên cứu.

```
%pip install ultralytics
import ultralytics
ultralytics.checks()
```

```
Ultralytics YOLOv8.2.2 🚀 Python-3.10.12 torch-2.2.1+cu121 CUDA:0 (Tesla T4, 15102MiB)
Setup complete ✅ (2 CPUs, 12.7 GB RAM, 28.8/78.2 GB disk)
```

```
!pip install Pillow
```

```
Requirement already satisfied: Pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
```

```
from PIL import Image
```

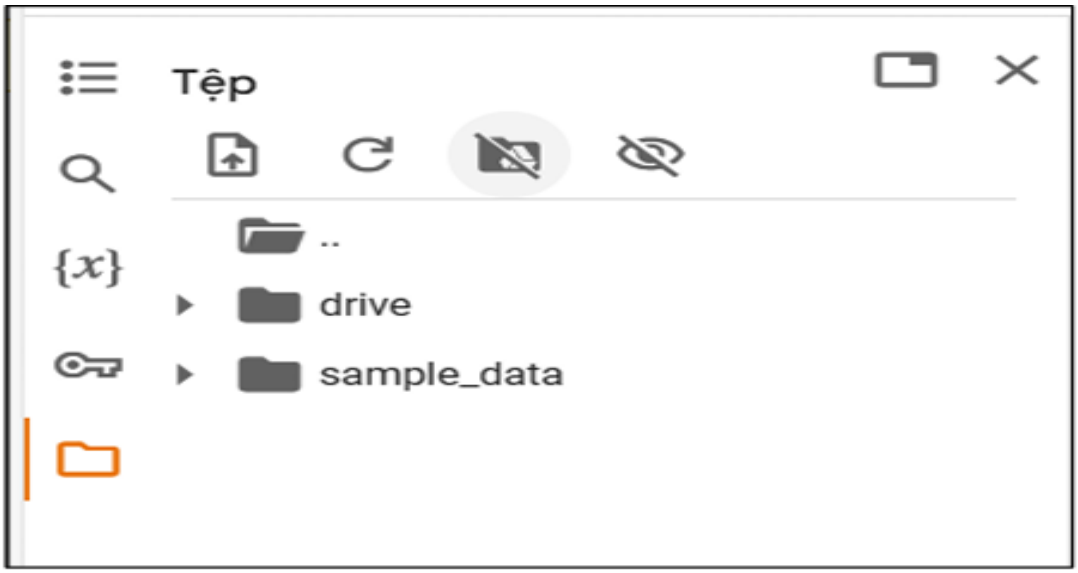
```
from IPython.display import Image
```

```
!apt install unzip
```

Hình 3.5: cài các thư viện hỗ trợ

Vì Google Colab bản miến phí chỉ cho giới hạn về thời gian chạy nên ta có thể liên kết với drive để lưu trữ để tránh mất dữ liệu sau khi hết phiên

Ta có thể dễ dàng kết nối với drive với 2 cách:



Hình 3.6: Ấn chọn biểu tượng drive trong phần tệp

```
from google.colab import drive
drive.mount('/content/drive')
```

Hình 3.7: kết nối qua đoạn mã

Sau khi cài xong một vài đoạn mã ta có thể bắt đầu train với đoạn mã sau

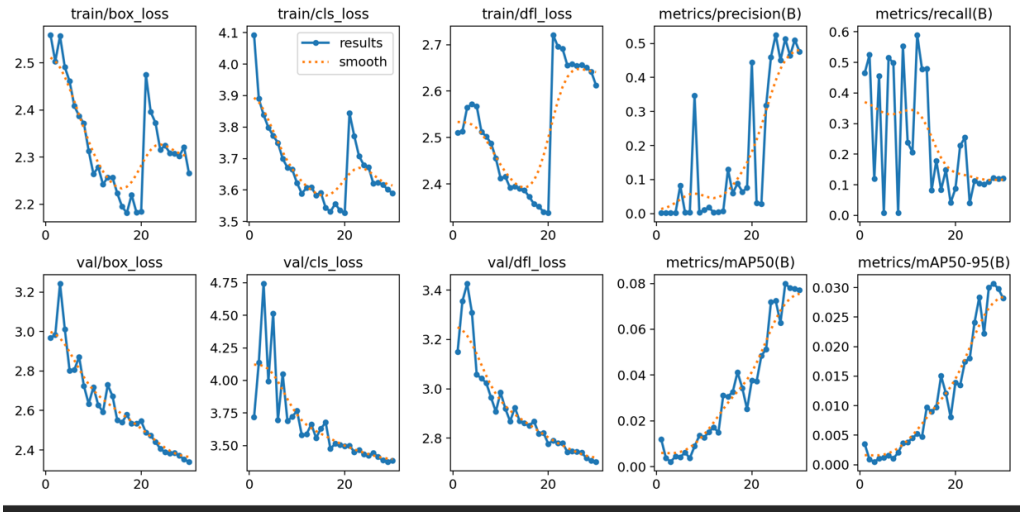
```
%cd {HOME}

yolo task=detect mode=train model=yolov8n.pt data={dataset.location}/data.yaml
epochs=100 imgsz=640
```

Hình 3.8: đoạn mã để bắt đầu train

Trong đó:

- %cd: là di chuyển đến tệp ta sẽ lưu trữ các dữ liệu khi đang train
- Epochs: Số lần ảnh được train (số càng lớn dữ liệu train sẽ được cải thiện nhưng đổi lại thì sẽ tốn thời gian hơn)
- Imgsz: size ảnh (đối với YOLOv8 nhà phát triển có yêu cầu size ảnh là 640x640).
- Model: mô hình được Ultralytics cúng cấp để đào tạo.
- Data: đường dẫn đến tệp cấu hình tập dữ liệu (ví dụ: coco8.yaml). Tệp này chứa các tham số dành riêng cho tập dữ liệu, bao gồm đường dẫn đến dữ liệu đào tạo và xác thực, tên lớp và số lượng lớp.

3.3.1. Giá trị các biểu đồ.



Hình 3.9.Biểu đồ thống kế sau khi đào tạo

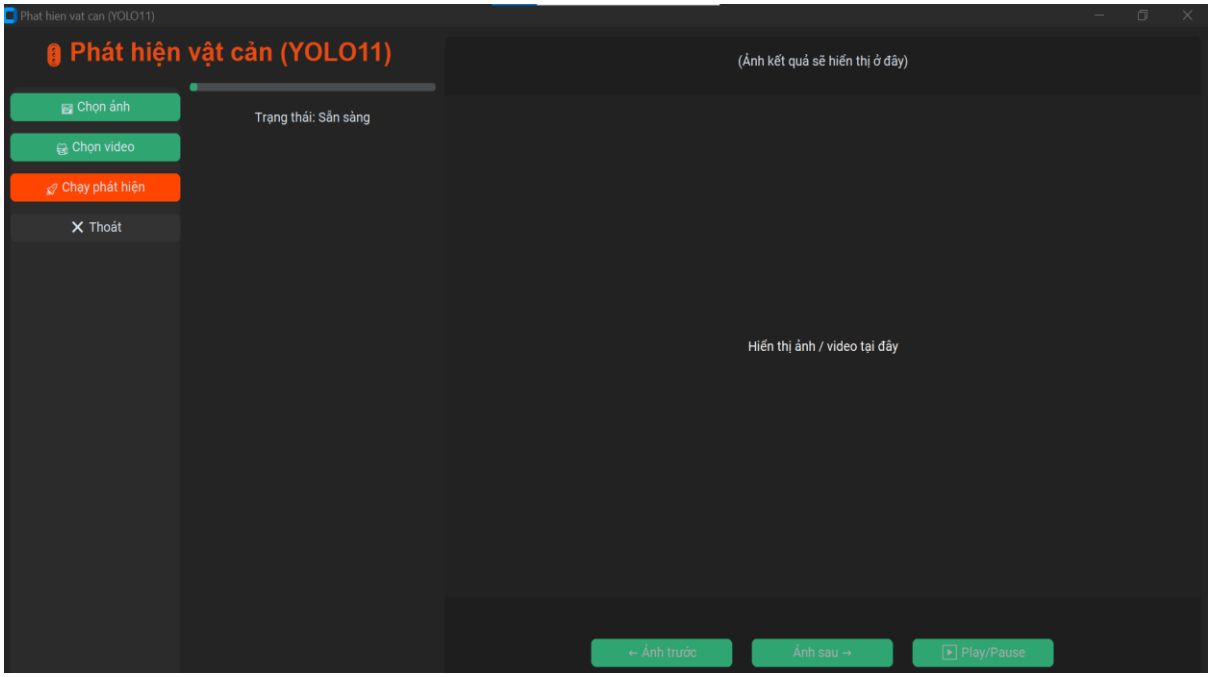
Theo như hình ta có thể thấy:

- Các biểu đồ mất mát đang giảm dần cho biết được mô hình đang được học tốt hơn.
- Biểu đồ độ chính xác: thì lại tăng dần lên

```
from ultralytics import YOLO

# Load mô hình PyTorch
model = YOLO("best.pt")

# Xuất sang định dạng TensorFlow Lite
model.export(format="tflite")
```

Hình 3.10. Đoạn code chuyển mô hình sang TF Lite

3.3.2. Giao diện ứng dụng phát hiện vật cản và biển báo



Hình 3.11: giao diện chương trình chạy trên VS code

Giao diện có các chức năng cơ bản gồm:

- Chọn ảnh: Dùng model đã train trên Google Colab để xử lý nhận diện ảnh và lưu ảnh về định dạng .JPG và lưu vào 1 tệp chỉ định.

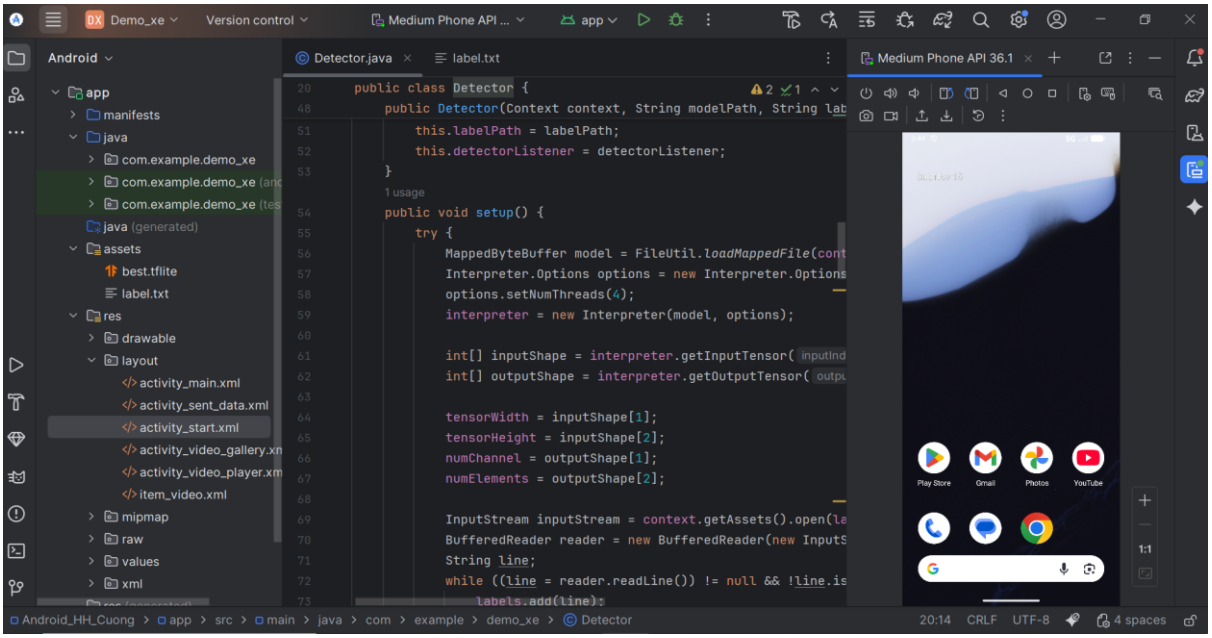
- Chọn Video: Dùng model đã train trên Google Colab để xử lý nhận diện đối tượng trong đoạn video

- Ảnh trước: dùng để xem ảnh trước đó nếu click nhiều ảnh để chạy

- Ảnh sau: dùng để xem ảnh sau đó nếu click nhiều ảnh để chạy

- Play/Pause: dùng để dừng đoạn video khi đang chạy

- Thoát: thoát chương trình



Hình 3.12.Giao diện trên Android Studio

3.3.3. Kết quả thử nghiệm trên mô hình đã huấn luyện

Thực hiện thủ công trên loại dữ liệu đầu vào:1 ảnh và 1 video đầy đủ các đối tượng: người đi bộ, vật cản, biển báo

Đối với ảnh tĩnh về 1 đối tượng độ nhận diện của chương trình chính xác khoảng 90%. Bảng sau đây cho ta thấy rõ về việc nhận diện đối tượng thông qua hình ảnh tĩnh trong thực tế

STT	Tên đối tượng	Số lượng ảnh	Nhận đúng	Nhận sai	Độ chính xác
1	Người đi bộ	100	95	5	95%
2	Vật cản	80	70	10	87%
3	Biển báo	56	50	19	89%

Bảng 3.2. Độ nhận diện đúng sai của từng đối tượng

Theo như bảng trên ra có thể thấy việc nhận diện hình ảnh tĩnh có thể nhận diện 1 cách tốt hơn

Hình 3.12. ảnh nhận diện người đi bộ qua đường trên vs code

Hình 3.13. ảnh nhận diện vật cản trên đường trên vs code

Hình 3.14: ảnh nhận diện biển báo giao thông

Khi sử dụng video để nhận diện chương trình cũng đã thực hoàn thành tốt việc nhận diện những là video có nhiều góc độ khác nhau nên độ chắc chắn của mô hình sẽ có lúc không được quá cao**.**

3.4. Đánh giá thực tế

Bảng dưới đây cho ta biết các lỗi sai mà mô hình nhận diện về đối tượng

Đối tượng	Lỗi mắc phải
Người đi bộ	- Nhận diện 1 đối tượng 2 lần - khó nhận diện người đi bộ ở xa
Biển báo	- nhận diện biển báo khó , cần phải có bộ dữ liệu lớn về biển báo
Vật cản	- nhận diện vật cản xe máy ở xa thành người

Bảng 3.3.bảng về các lỗi còn mắc phải của chương trình

Bảng dưới đây sẽ cho ta thấy được thực tế về việc nhận diện đối tượng phụ thuộc nhiều quan trọng vào thời điểm, thời tiết tốt hay xấu.

Môi trường	Quan sát , đánh giá	Đối tượng nhận diện
Ban ngày	Ánh sáng tốt, ảnh thu được đối tượng	Dễ nhận diện
Ban đêm	Ánh sáng yếu, ảnh thu được không nhận diện được hết các đối tượng	Khó nhận diện
Thời tiết xấu(mưa)	YOLO bị nhiễu	Khó nhận diện

Bảng 3.4.bảng đánh giá thực tế sự nhận diện đối tượng đối với môi trường xung quanh

3.5.Đánh giá mô hình

Mô hình YOLO11n đã được sử dụng để nhận diện người đi bộ, vật cản, biển báo trên Google Colab, với bộ dữ liệu gần 200 hình ảnh của 3 loại đối tượng khác nhau. Bài báo cáo này sẽ đánh giá các kết quả đạt được, cũng như nêu rõ các ưu điểm và nhược điểm của mô hình.

3.5.1. Ưu điểm của mô hình.

Hiệu suất và độ chính xác: YOLO11n đã cho thấy hiệu suất khá tốt với độ chính xác cao đối với các hình ảnh tĩnh chứa đối tượng.

Khả năng xử lý: Hệ thống xử lý nhanh và chính xác đối với các hình ảnh tĩnh và những video có một đối tượng.

3.5.2. Nhược điểm của mô hình.

Độ chính xác với hình ảnh phức tạp: Với các hình ảnh chứa nhiều đối tượng khác nhau, độ chính xác chỉ đạt mức khá.

Sai sót trong nhận diện: Mô hình còn gặp khó khăn khi nhận diện các đối tượng có nhiều trong một ảnh hoặc các đối tượng ở quá xa.

Biển báo, vật cản cần bộ dữ liệu khá đa dạng và lớn, để có thể tạo bộ huấn luyện hoàn chỉnh hơn.

3.5.3. Đề xuất cải tiến.

- Thêm dữ liệu đào tạo ,bổ sung thêm dữ liệu đa dạng để cải thiện khả năng nhận diện các đối tượng tốt hơn.

- Nâng cấp mô hình: Sử dụng các phiên bản cao cấp hơn của YOLO sắp tới ra mắt.

- Tối ưu hóa tài nguyên: Sử dụng các dịch vụ đám mây khác hoặc phiên bản trả phí của Google Colab để có tài nguyên huấn luyện mạnh mẽ hơn.

- Mô hình YOLO11n đã phần nào đáp ứng được các yêu cầu của đề tài, đặc biệt là đối với các hình ảnh tĩnh. Tuy nhiên, vẫn cần cải thiện thêm để tăng độ chính xác và giảm sai sót trong quá trình nhận diện đối tượng. Việc nâng cấp mô hình và bổ sung dữ liệu sẽ là các bước quan trọng để đạt được kết quả tốt hơn trong tương lai.

KẾT LUẬN

Sau thời gian gần 5 tháng nghiên cứu và thực hiện đồ án tốt nghiệp với sự hướng dẫn của cô TS. Hồ Thị Hương Thơm – Khoa Công Nghệ Thông Tin trường Đại Học Hàng Hải Việt Nam để hoàn thiện được đồ án với đề tài: xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái.

Một số mục tiêu mà đồ án đã đạt được:

Hoàn thành đề tài: ” Xây dựng mô hình phát hiện vật cản và biển báo hỗ trợ ra quyết định cho xe tự lái”

Đào tạo thành công mô hình nhận diện với 3 loại đối tượng khác nhau: người đi bộ, vật cản, biển báo bằng mô hìn YOLO11n.

Mô hình phân loại được các loại trái cây với độ chính xác trung bình với những ảnh tĩnh độ chính xác nhận diện đối tượng cao hơn và vẫn còn nhiều sai sót trong việc nhận diện video. Tuy nhiên khi đưa vào thực tiễn có thể đáp ứng được phần nào yêu cầu đặt ra nhưng có thể sẽ không được chính xác. Muốn cải thiện độ chính xác thì ảnh trước khi đưa vào phân loại cần phải được xử lý nâng cao chất lượng ảnh, hoặc ảnh được chụp/ quay từ các thiết bị chuyên dụng có chất lượng cao.

Chúng ta có thể phát triển chương trình này thành một mô-đun tiền xử lý hình ảnh cho các các đối tượng, để tích hợp vào hệ thống xe tự lái. Việc tự động hóa này sẽ góp phần hiện đại hóa ngành công nghiệ xe tự lái, hướng tới 1 quy trình công nghệ hóa hiện đại hóa. Điều này sẽ hỗ trợ hiệu quả cho nền sản xuất xe tự lái áp dụng vào thực tế với môi trường phát triển như hiện nay.

Trong quá trình học tập và nghiên cứu, tìm hiểu một lĩnh vực rất mới và nhiều thử thách đối với bản thân sinh viên cho nên đồ án của sinh viên sẽ còn có những thiếu sót và hạn chế. Sinh viên rất cầu thị quý thầy cô trong khoa Công nghệ thông tin và Viện đào tạo sau đại học có cái nhìn cảm thông và chia sẻ đóng

góp những lời nhận xét để sinh viên nâng cao sự hiểu biết về lĩnh vực AI vốn là lĩnh vực mới và hoàn thiện hơn đồ án của mình.

Em xin cảm ơn sự hướng dẫn chỉ bảo của Cô TS. Hồ Thị Hương Thơm

TÀI LIỆU THAM KHẢO

1. Hình ảnh về biển báo. Có tại: https://lythuyet.onthigplx.vn/cac-loai-bien-bao-giao-thong-duong-bo/
2. Hình ảnh vật cản(xe máy, ô tô,…). Có tại: https://universe.roboflow.com/my-xinh-z1bpm/phat-hien-vat-can
3. Hình ảnh về người đi bộ. Có tại: https://www.istockphoto.com/vi/search/2/image
4. Python Software Foundation, "Python Documentation," 2024. Có tại: https://docs.python.org/3/
7. Google Research, "Hướng dẫn sử dụng Google Colab," Google Research Documentation, 2024. Có tại: https://colab.research.google.com/
8. Microsoft, "Visual Studio Code Documentation," 2024. Có tại: https://code.visualstudio.com/docs/
9. Ultralytics, "YOLOv11 Model Release and Benchmark Report," 2025. Có tại: https://github.com/ultralytics/YOLOv11
10. Roboflow, "Computer Vision Dataset Management," 2024. Có tại: https://roboflow.com/
11. Goodfellow, Y. Bengio, and A. Courville, Deep Learning. Cambridge, MA: MIT Press, 2016. Có tại: https://www.deeplearningbook.org/