BỘ GIÁO DỤC VÀ ĐÀO TẠO **BỘ NÔNG NGHIỆP VÀ PTNT**
TRƯỜNG ĐẠI HỌC THỦY LỢI

NGUYỄN GIA BẢO

XÂY DỰNG HỆ THỐNG CẢNH BÁO CÁC TRƯỜNG HỢP NGUY HIỂM TẠI KHÚC CUA SỬ DỤNG MÔ HÌNH HỌC SÂU TRONG THỊ GIÁC MÁY TÍNH

ĐỒ ÁN TỐT NGHIỆP

HÀ NỘI, NĂM 2024

BỘ GIÁO DỤC VÀ ĐÀO TẠO BỘ NÔNG NGHIỆP VÀ PTNT
TRƯỜNG ĐẠI HỌC THỦY LỢI

NGUYỄN GIA BẢO

XÂY DỰNG HỆ THỐNG CẢNH BÁO CÁC TRƯỜNG HỢP NGUY HIỂM TẠI KHÚC CUA SỬ DỤNG MÔ HÌNH HỌC SÂU TRONG THỊ GIÁC MÁY TÍNH

Ngành: Công nghệ thông tin

Mã số: 7480201

NGƯỜI HƯỚNG DẪN: 1. ThS. Trương Xuân Nam

HÀ NỘI, NĂM 2024

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

----------★----------

NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Nguyễn Gia Bảo Hệ đào tạo : Chính quy

Lớp: 62TH-VA

Ngành: Công nghệ thông tin

Khoa: Công nghệ thông tin

1. TÊN ĐỀ TÀI:

Xây dựng hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua sử dụng mô hình học sâu trong thị giác máy tính

2. CÁC TÀI LIỆU CƠ BẢN:

3. CÁC TÀI LIỆU THAM KHẢO

4. NỘI DUNG CÁC PHẦN THUYẾT MINH VÀ TÍNH TOÁN:

Nội dung thuyết minh và tính toán	Tỷ lệ %
Chương 1: Giới thiệu bài toán, giới thiệu tổng quan	15%

Chương 2: Các kĩ thuật sử dụng trong bài toán về Object Detection.	25%
Chương 3: Ứng dụng cho "Hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua".	25%
Chương 4: Kết quả và định hướng phát triển	25%
Kết luận	10%

5 - GIÁO VIÊN HƯỚNG DẪN TỪNG PHẦN:

Giáo viên hướng dẫn toàn bộ quá trình thực hiện đồ án: **ThS. Trương Xuân Nam**

6. NGÀY GIAO NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Ngày ..... tháng ..... năm 2024

Trưởng Bộ môn

(Ký và ghi rõ Họ tên)

Giáo viên hướng dẫn chính

(Ký và ghi rõ Họ tên)

Nhiệm vụ Đồ án tốt nghiệp đã được Hội đồng thi tốt nghiệp của Khoa thông qua

Ngày ..... tháng ..... năm 2024

Chủ tịch Hội đồng

(Ký và ghi rõ Họ tên)

Sinh viên đã hoàn thành và nộp bản Đồ án tốt nghiệp cho Hội đồng thi ngày... tháng... năm 2024

Sinh viên làm Đồ án tốt nghiệp

(Ký và ghi rõ Họ tên)

TRƯỜNG ĐẠI HỌC THỦY LỢI

KHOA CÔNG NGHỆ THÔNG TIN

BẢN TÓM TẮT ĐỀ CƯƠNG ĐỒ ÁN TỐT NGHIỆP

TÊN ĐỀ TÀI: Xây dựng hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua sử dụng mô hình học sâu trong thị giác máy tính

Sinh viên thực hiện: Nguyễn Gia Bảo

Lớp : 62THVA

Email: kevinbao15072002@gmail.com

Điện thoại : 0378203533

Giáo viên hướng dẫn : Thầy Trương Xuân Nam

TÓM TẮT ĐỀ TÀI

Hiện nay, vấn đề an toàn giao thông là một vấn đề đáng lo ngại mà không chỉ ở Việt Nam mà còn trên toàn thế giới. Mỗi năm, hàng triệu người trên thế giới mất mạng hoặc bị thương do các tai nạn giao thông. Ở Việt Nam, trong 10 năm qua, con số 190.000 vụ tai nạn giao thông là một con số đáng báo động, và đặc biệt là các vụ tai nạn đường bộ chiếm tỷ lệ lớn.

Theo thống kê cho thấy, các vụ tai nạn đường bộ xảy ra chủ yếu tại các khúc cua ngoằn ngoèo, hiểm trở. Nguyên nhân chính dẫn đến các vụ tai nạn thương tâm là do khả năng và quan sát của các tài xế khi tham gia giao thông. Đại đa số các tài xế cho rằng họ không thể phát hiện sớm các phương tiện, các sự vật hiện tượng ở phía sau khúc cua. Điều này chứng tỏ rằng các tài xế có rất ít thời gian giữa việc phát hiện và phản ứng trước các sự kiện đang diễn ra, dẫn đến các vụ tai nạn không mong muốn.

Ngoài ra, cấu trúc đường và thiết kế của các khúc cua cũng đóng một vai trò quan trọng trong việc tạo ra các điểm mù, làm cho việc quan sát trở nên khó khăn hơn. Sự kết hợp giữa tốc độ di chuyển của phương tiện và khả năng nhận thức của tài xế có thể dẫn đến các hậu quả không lường trước được.

Vì vậy, để giảm thiểu rủi ro tai nạn tại các khúc cua, em quyết định xây dựng "**Hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua**" với mong muốn đóng góp một chút công sức để cải thiện chất lượng cuộc sống người dân Việt Nam.

Công nghệ sử dụng:

- Ngôn ngữ python.
- Học sâu cho thị giác máy tính
- Fine-turning pretrained model

CÁC MỤC TIÊU CHÍNH

Mục tiêu 1. Tìm hiểu các giải pháp hiện nay

Mục tiêu 2. Tìm hiểu về các thiết bị máy tính nhúng, các thiết bị ngoại vi.

Mục tiêu 3. Tìm hiểu về các mô hình học sâu trong thị giác máy tính.

Mục tiêu 4. Tìm hiểu về Object Detection và Object Tracking.

Mục tiêu 5. Tìm hiểu về mô hình đã được huấn luyện.

Mục tiêu 5. Cải thiện tốc độ xử lý thông tin và độ chính xác của hệ thống.

Mục tiêu 6. Xây dựng "Hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua".

Mục tiêu 7. Đánh giá mô hình

KẾT QUẢ DỰ KIẾN

- Hoàn thành các mục tiêu đã đề ra
- Xây dựng "**Hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua**"
- Báo cáo tổng kết Đồ Án Tốt Nghiệp

TÀI LIỆU THAM KHẢO

https://github.com/tensorflow/tensorflow/tree/v2.11.0

https://www.tensorflow.org/api_docs/python/tf

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

https://arxiv.org/abs/1512.02325

https://medium.com/visionwizard/object-detection-4bf3edadf07f

https://arxiv.org/pdf/1807.05511

LỜI CAM ĐOAN

Em xin cam đoan đây là Đồ án tốt nghiệp của cá nhân em. Các kết quả trong Đồ án tốt nghiệp này là trung thực và thực hiện dưới sự hướng dẫn của thầy ThS.Trương Xuân Nam. Việc tuân thủ quy định về trích dẫn và ghi nguồn tài liệu tham khảo đã được thực hiện sau khi tham khảo các nguồn tài liệu (nếu có).

Tác giả ĐATN

Bảo

Nguyễn Gia Bảo

LỜI CẢM ƠN

Qua quãng thời gian học tập, rèn luyện và nghiên cứu tại Khoa Công Nghệ Thông Tin - Trường Đại Học Thủy Lợi, bản thân em đã trải qua một chặng đường học với một môi trường đào tạo chất lượng, được hướng dẫn bởi những thầy cô tận tâm và yêu quý.

Em xin gửi lời cảm ơn chân thành đến quý thầy cô tại Trường Đại học Thủy Lợi, người đã trang bị cho em những kiến thức vô cùng quý báu suốt bốn năm học tập tại đây. Đặc biệt, lòng nhiệt huyết và sự chuyên nghiệp của các thầy cô trong Khoa Công nghệ thông tin đã tận tình giảng dạy, chỉ bảo và trang bị cho em những kiến thức chuyên ngành vô cùng quý giá, giúp em có đủ nền tảng để hoàn thành Đồ án tốt nghiệp này. Em chân thành biết ơn sự đồng hành và sự hỗ trợ quý báu từ các thầy cô trong suốt hành trình học tập của mình.

Em muốn bày tỏ lòng biết ơn sâu sắc đặc biệt đến TS. Trương Xuân Nam, giảng viên tận tình của Trường Đại học Thủy Lợi. Từ những ngày đầu tiên xây dựng đồ án, thầy đã không ngừng hướng dẫn em, đưa ra những góp ý chi tiết và truyền đạt những kinh nghiệm quý báu. Những đóng góp này đã giúp em vượt qua những thách thức và hoàn thành Đồ án tốt nghiệp một cách xuất sắc. Em xin gửi lời cảm ơn sâu sắc nhất đến TS. Trương Xuân Nam vì sự tận tâm và sự hỗ trợ quý báu trong quá trình nghiên cứu và thực hiện đồ án của mình.

Qua quá trình hoàn thành đồ án, em nhận thức rõ rằng kiến thức của mình vẫn còn hạn chế và thiếu kinh nghiệm thực tế. Điều này là không tránh khỏi và em tự nhận ra những điểm chưa hoàn hảo. Vì lẽ đó, em chân thành kính mong nhận được những ý kiến đóng góp quý báu từ quý thầy cô để em có cơ hội hoàn thiện và nâng cao sâu rộng kiến thức của mình trong lĩnh vực này.

Em xin chân thành cảm ơn vì sự quan tâm và hỗ trợ từ phía quý thầy cô!

Mục Lục

DANH MỤC CHỮ VIẾT TẮT

1 AI: Artificial Intelligence - Trí tuệ nhân tạo

2 BDD: Berkeley DeepDrive – Bộ dữ liệu về các phương tiện của Đại học California, Berkeley

3 CNN: Convolutional Neural Networks – Mạng thần kinh tích chập

4 COCO: Common Objects in Context - Các đối tượng phổ biến trong bối cảnh

5 CV: Computer Vision - Thị giác máy tính

6 DL: Deep Learning - Học sâu

7 FN: False Negative - Số lượng các vùng là đối tượng mà mô hình không dự đoán được.

8 FP: False Positive - Số lượng các vùng không phải đối tượng mà mô hình không dự đoán được.

9 FPN: Feature Pyramid Network - Mạng Cấu Trúc Kim Tự Tháp

10 FPS: Frame Per Second – Số lượng khung hình trong một giây

11 ID: Identification - Định danh

12 ML: Machine learning - Học máy

13 MLP: Multi-layer Perceptron

14 PLA: Perceptron learning Algorithim

15 SSD: Single Shot MultiBox Detector

16 TN: True Negative - Số lượng các vùng không phải đối tượng được mô hình dự đoán đúng.

17 TP: True Positive - Số lượng các vùng là đối tượng được mô hình dự đoán đúng.

DANH MỤC HÌNH ẢNH

MỞ ĐẦU

Computer Vision là 1 trong những lĩnh vực có tầm ảnh hưởng lớn đối với ngành Trí tuệ nhân tạo trong 10 năm qua, góp phần đưa sự ảnh hưởng của trí tuệ nhân tạo vươn tầm thế giới. Computer Vision gồm các phương thức, thuật toán nhằm tạo sự thuận tiện trong các công việc như xử lý ảnh kỹ thuật số, phân tích và nhận dạng hình ảnh. Hơn cả Object Detection là một phần ứng dụng sâu sắc nhất của Computer Vision do được cộng đồng phản hồi tích cực về tính thực tế và hữu dụng.

Object Detection có thể được hiểu là khả năng phát hiện, nhận dạng được vị trí của đối tượng, vật thể trong ảnh của hệ thống máy tính và phần mềm. Object Detection được ứng dụng rộng rãi trong các tác vụ thực tế như: phát hiện khuôn mặt, phát hiện biển số xe, hệ thống bảo mật và hệ thống xe không người lái.

Trong xã hội ngày nay, vấn đề an toàn giao thông ở các khúc cua đang là một thách thức lớn đối với xã hội. Việc áp dụng các giải pháp Object Detection sẽ giúp giảm thiểu các vụ tai nạn tại các khúc cua, giúp cải thiện chất lượng an toàn giao thông và quản lý lưu lượng xe cộ. Việc đưa ra các cảnh báo sớm có thể hỗ trợ tài xế trong việc quan sát và phản ứng kịp thời trước các tình huống nguy hiểm.

Trong thực tế, nhiều thành phố trên thế giới đã áp dụng các giải pháp Object Detection và đã có được nhiều kết quả tích cực. Điều này cho thấy rằng, áp dụng công nghệ AI trong lĩnh vực giao thông không chỉ là biện pháp hiện đại hóa mà còn là một cách thông minh và hiệu quả để giải quyết các vấn đề liên quan đến giao thông.

Vì vậy em sẽ đi sâu vào thách thức và triển vọng của Object Detection, đặc biệt là áp dụng vào bài toán thực tế - Cảnh báo các trường hợp nguy hiểm tại các khúc cua.

Đặt vấn đề

Vấn đề về an toàn giao thông đường bộ luôn là chủ đề nóng được đông đảo người dân quan tâm tới. Những vụ tai nạn không mong muốn xảy ra tại các khúc cua luôn để những sự mất mát đau thương, sự thiệt hại trầm trọng về cả vật chất và tinh thần. Vì vậy việc ứng

dụng Object Detection vào việc cảnh báo các trường hợp nguy hiểm tại các khúc cua có tiềm năng rất lớn trong việc cải thiện vấn đề an toàn giao thông đường bộ. Công nghệ Object Detection góp phần giảm thiểu các vụ tai nạn giao thông và hỗ trợ quản lý và giám sát giao thông hiệu quả.

Tuy nhiên, bên cạnh tiềm năng lớn với nhiều lợi ích mà Object Detection mang lại, thì công nghệ này còn nhiều vấn đề ảnh hưởng tới hệ thống:

- Điều kiện thời tiết: Môi trường có điều kiện thời tiết xấu làm giảm chất lượng của ảnh đầu vào hệ thống.
- Khả năng phát hiện: Khó khăn trong việc phát hiện các đối tượng bị che khuất bởi vật cản hoặc đối tượng khác
- Cấu trúc khung đường: tại khúc cua gây ra hiện tượng mất thông tin nếu có vật cản đứng trước vật thể cần phát hiện.
- Tính thời gian thực: Độ trễ trong xử lý thông tin có thể ảnh hưởng đến khả năng phản ứng kịp thời của tài xế khi vào khúc cua.
- Khả năng phát hiện vật thể đi đúng làn đường.

Mục tiêu nghiên cứu

Việc ứng dụng công nghệ Object Detection vào vấn đề an toàn giao thông đem lại nhiều giá trị cho ngành giao thông vận tải và đời sống của người dân.

Hệ thống được thiết kế để có thể hoạt động được:

- Trong môi trường có điều kiện thời tiết tương đối xấu.
- Có thể phát hiện được vật thể bị cản.
- Cảnh báo trực tiếp cho các tài xế và người tham gia giao thông.

Hệ thống được xây dựng để cảnh báo các trường hợp sau:

- Các trường hợp xe đi đúng chiều.
- Các trường hợp xe đi ngược chiều.

- Các trường hợp xe đứng yên.

Hệ thống có thể cải thiện thêm về tốc độ xử lý và độ chính xác phù hợp và đánh giá kết quả đạt được.

Ý nghĩa của bài toán

Ứng dụng công nghệ Object Detection cho việc cảnh báo các trường hợp nguy hiểm tại khúc cua đem lại nhiều giá trị về thiết thực cho an toàn giao thông:

Nâng cao an toàn giao thông:

- Giảm thiểu tai nạn: Nhờ khả năng phát hiện và cảnh báo sớm các mối nguy hiểm tiềm ẩn như người đi bộ, xe cộ, chướng ngại vật,... tại các khúc cua, công nghệ Object Detection góp phần đáng kể vào việc giảm thiểu tai nạn giao thông, đặc biệt là trên những cung đường nguy hiểm.
- Hỗ trợ lái xe an toàn: Hệ thống cảnh báo dựa trên Object Detection giúp người lái xe nhận thức rõ ràng hơn về các sự vật, sự kiện phía bên kia khúc cua, từ đó điều chỉnh tốc độ và hành vi lái xe phù hợp, đảm bảo an toàn cho bản thân và những người tham gia giao thông khác.

Lợi ích kinh tế:

- Giảm thiểu thiệt hại do tai nạn giao thông: Tai nạn giao thông gây ra nhiều thiệt hại về người và tài sản. Việc ứng dụng Object Detection để cảnh báo nguy hiểm tại các khúc cua góp phần giảm thiểu tai nạn, qua đó tiết kiệm chi phí cho y tế, sửa chữa, bảo hiểm,...
- Nâng cao hiệu quả hoạt động giao thông: Giảm ùn tắc giao thông giúp tiết kiệm thời gian di chuyển, tăng năng suất làm việc và thúc đẩy phát triển kinh tế.

Phương pháp nghiên cứu

Xây dựng dữ liệu: Để có thể xây dựng được hệ thống, em đi thu thập dữ liệu về các ảnh phương tiện và vật thể trên đường bộ của nước ta và các phương tiện trên thế giới. Dữ liệu được làm sạch, tiền xử lý và gán nhãn để loại bỏ các dữ liệu nhiễu và đảm bảo tính nhất quán của toàn bộ dữ liệu.

Máy tính nhúng Raspberry PI 4: Việc sử dụng máy tính nhúng Raspberry PI 4 cùng công nghệ Object Detection trong việc cảnh báo các trường hợp nguy hiểm tại khúc cua là một giải pháp tiết kiệm chi phí khi sản xuất hàng loạt nhưng vẫn đảm bảo việc nâng cao chất lượng an toàn giao thông.

Kiến trúc mô hình: Trong hệ thống này em sử dụng mô hình SSD MobileNET để có được tốc độ phản hồi thông tin nhanh chóng với độ chính xác cao, phù hợp với cấu hình của Raspberry PI4 và nhu cầu cần thiết khi bố trí các khúc cua.

Huấn luyện và đánh giá: Mô hình được huấn luyện dựa trên bộ dữ liệu được chia thành ba bộ: bộ huấn luyện, bộ xác nhận và bộ kiểm thử. Quá trình huấn luyện sử dụng thuật toán tối ưu hóa và được theo dõi bằng các độ đo hiệu suất như độ chính xác và giá trị mất mát trên tập kiểm tra. Để đánh giá hiệu suất, chúng Em sử dụng các tập kiểm tra chứa hình ảnh các phương tiện giao thông đa dạng và đánh giá kết quả dự đoán của mô hình so với các nhãn thực tế. Các độ đo như độ chính xác Precision, recall, .. được sử dụng để đánh giá hiệu suất của mô hình.

Sử dụng kĩ thuật xử lý ảnh:

- Các hình ảnh trong bộ dữ liệu sẽ được xử lý để cải thiện chất lượng của ảnh trước khi được đưa vào mô hình để huấn luyện nhằm nâng cao chất lượng và mở rộng sự đa dạng của dữ liệu.
- Ảnh đầu vào của hệ thống sẽ được đưa qua các kỹ thuật xử lý ảnh có thể xác định được vị trí của các vật thể có thuộc phạm vi ảnh hưởng không.

Thử nghiệm và đánh giá: Em thực hiện một loạt các thử nghiệm để đánh giá hiệu suất của mô hình trên các hình ảnh thực tế và so sánh với các mô hình khác. Kết quả của mỗi thử nghiệm được thảo luận và đánh giá chi tiết trong phần Kết Quả và Đánh giá

Chương 1: Giới thiệu bài toán, giới thiệu tổng quan

1.1. Giới thiệu tổng quan

Computer Vision là một lĩnh vực trong trí tuệ nhân tạo và khoa học máy tính, nghiên cứu về cách máy tính có thể hiểu và xử lý hình ảnh và video để tự động nhận biết, phân loại và hiểu về thế giới xung quanh chúng.

1.1.1. Ngữ cảnh lịch sử

- Sự xuất hiện đầu tiên của Computer Vision:
  Nguồn gốc của thị giác máy tính có thể bắt nguồn từ những năm 1950 và 1960 khi các nhà nghiên cứu lần đầu tiên bắt đầu khám phá ý tưởng dạy máy tính hiểu và giải thích dữ liệu hình ảnh. Mục đích tạo ra Computer Vision là để mô phỏng lại hệ thống thị giác của con người, sau đó trang bị lên những con robot (Người máy).

- Những thành tựu quan trọng trong những năm 1970 - 1990:
  Các nhà nghiên cứu tập trung phát triển các thuật toán phát hiện cạnh và trích xuất đặc điểm từ hình ảnh. Những kỹ thuật này đặt nền tảng cho các nhiệm vụ thị giác máy tính nâng cao hơn.
  Sự ra đời của “Máy dò cạnh Canny” và “Biến đổi Hough" đã đóng góp công lao trong việc phát hiện cạnh và trích xuất đặc trưng trong ảnh.
  Đến năm 1990, các nhà nghiên cứu bắt đầu khám phá khả năng nhận dạng đối tượng và hiểu cảnh bằng kỹ thuật học máy.
  Mạng nơ-ron “Tương quan theo tầng” và sự phát triển của thuật toán Chuyển đổi tính năng bất biến quy mô là những cột mốc quan trọng. Chúng giúp cải thiện độ chính xác và hiệu quả của các hệ thông Computer Vision trong các nhiệm vụ khác nhau.
- Cuộc cách mạng Deep Learning

Sự đột phá trong học sâu, đặc biệt là Mạng thần kinh chuyển đổi (CNN), đã cách mạng hóa thị giác máy tính. CNN tỏ ra có hiệu quả cao trong nhiệm vụ phân loại hình ảnh.

Thử thách nhận dạng hình ảnh quy mô lớn của ImageNet (ILSVRC) đã đóng một vai trò quan trọng trong việc thúc đẩy học sâu trong thị giác máy tính.

Các kỹ thuật học chuyển giao, chẳng hạn như tinh chỉnh các mô hình được đào tạo trước, đã trở nên phổ biến trong thị giác máy tính.

Mạng đối thủ sáng tạo (GAN) đã được sử dụng để tạo ra hình ảnh và video thực tế.

Sự phát triển của phần cứng chuyên dụng, chẳng hạn như Bộ xử lý đồ họa (GPU) và Bộ xử lý Tensor (TPU), đã đẩy nhanh quá trình đào tạo mạng lưới thần kinh sâu cho thị giác máy tính.

1.1.2. Lợi ích của Computer Vision:

Tự động hóa và hiệu quả:

- Tự động hóa các nhiệm vụ tốn thời gian và nguy hiểm: Computer Vision có thể được sử dụng để tự động hóa các nhiệm vụ nguy hiểm hoặc tốn thời gian cho con người, chẳng hạn như kiểm tra sản phẩm, giám sát an ninh, lái xe tự động.
- Nâng cao hiệu quả hoạt động: Computer Vision có thể giúp các doanh nghiệp nâng cao hiệu quả hoạt động bằng cách tự động hóa các quy trình, tối ưu hóa quy trình và giảm thiểu lỗi.

Cải thiện an toàn và bảo mật:

- Nâng cao an ninh: Computer Vision có thể được sử dụng để giám sát các khu vực, phát hiện xâm nhập và nhận dạng các đối tượng nguy hiểm.
- Cải thiện an toàn giao thông: Computer Vision có thể được sử dụng để phát hiện các mối nguy hiểm trên đường, hỗ trợ người lái xe và ngăn ngừa tai nạn.

Phát triển khoa học và công nghệ:

- Thúc đẩy nghiên cứu khoa học: Computer Vision được sử dụng trong nhiều lĩnh vực khoa học như y học, thiên văn học, robot học để thu thập và phân tích dữ liệu hình ảnh.
- Phát triển các công nghệ mới: Computer Vision là nền tảng cho nhiều công nghệ mới như xe tự lái, thực tế ảo, thực tế tăng cường.

Nâng cao chất lượng cuộc sống:

- Cải thiện dịch vụ y tế: Computer Vision được sử dụng để chẩn đoán bệnh, hỗ trợ phẫu thuật và theo dõi sức khỏe bệnh nhân.
- Giáo dục và giải trí: Computer Vision được sử dụng để tạo ra các trải nghiệm giáo dục tương tác, trò chơi điện tử và các ứng dụng giải trí khác.

1.2. Lý do lựa chọn đề tài:

Vấn đề an toàn giao thông đường bộ luôn là chủ đề nóng trong những năm vừa qua. Có rất nhiều vụ tai nạn xảy ra trên các cung đường bộ, đặc biệt là tại các khúc cua. Nguyên nhân chủ yếu là do các tài xế, những người tham gia giao thông không thể phản ứng kịp thời trước các sự kiện phía bên kia khúc cua. Vì vậy đề tài này hướng tới việc phát triển hệ thông Computer Vision có khả năng nhận biết các tình huống có thể gây nguy hiểm và cảnh báo tới những người tham gia giao thông, làm giảm thiểu các vụ tai nạn có thể xảy ra tại khúc cua.

1.3. Sự cần thiết trong an toàn giao thông:

Hệ thống này không chỉ giúp giảm thiểu các vụ tai nạn hy hữu có thể xảy ra, mà còn giúp quản lý lưu thông đường bộ, cải thiện chất lượng an toàn giao thông đường bộ, thức đẩy chất lượng đời sống của nhân dân.

1.4. Mục tiêu nghiên cứu

Mục tiêu chính của đề tài này là xây dựng và đánh giá hệ thống Computer Vision với khả năng phát hiện các trường hợp gây nguy hiểm và cảnh báo tới người tham gia giao thông.

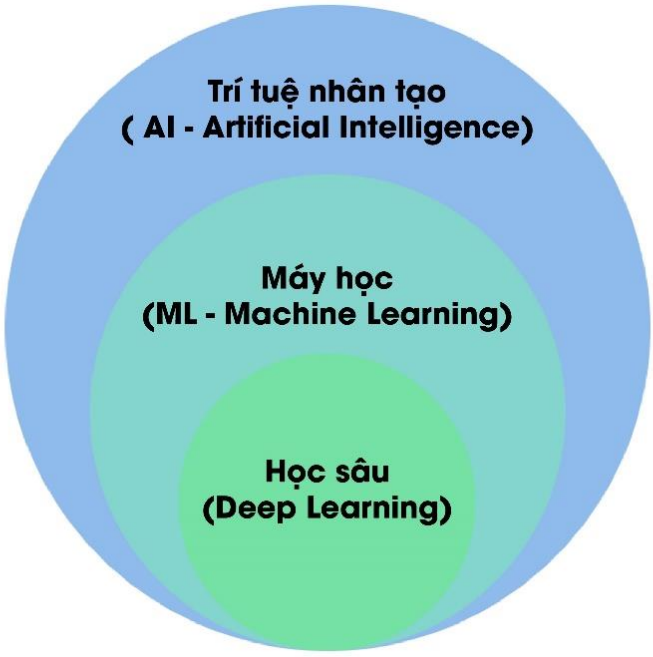
1.5. Phạm vi nghiên cứu

Nghiên cứu tập trung vào việc phát triển hệ thống Computer Vision xử lý các hình ảnh, video được thu thập từ Camera, phát hiện và theo dõi vật thể trong các hình ảnh và video đó.

Chương 2: Các kĩ thuật trong Computer Vision

Ở chương này sẽ tập trung thảo luận về các kỹ thuật quan trọng được sử dụng trong "Hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua" sử dụng Object Detection. Với các kỹ thuật này, chúng ta có thể xây dựng được hệ thống có khả năng cảnh báo nguy hiểm tới các tài xế, người tham gia giao thông để tránh được những vụ tai nạn không mong muốn. Cụ thể chương này sẽ chia sẻ gồm những phần sau:

2.1. Deep learning



Hình 1 Hình ảnh học sâu trong trí tuệ nhân tạo

Deep Learning (Học sâu) là một phần trong trí tuệ nhân tạo. Bằng cách xây dựng và huấn luyện mô hình có nhiều lớp nơ-ron, máy tính có thể học và phân tích các dữ liệu phức tạp. Trong quá trình huấn luyện mô hình mạng nơ-ron, các trọng số có thể thay đổi và nhiệm vụ chính của mô hình sẽ là tìm ra bộ trọng số phù hợp. Các hệ thống Deep Learing hiện nay thường có hàng trăm, hàng nghìn tham số, vì vậy mà các hệ thống này cần rất nhiều

thời gian và chi phí tính toán để huấn luyện mô hình, và kết quả có thể là khả năng hiểu sâu bộ dữ liệu và khả năng khái quát hóa dữ liệu.

Deep learning được xây dựng để mô phỏng khă năng tư duy của bộ não con người. Quy trình hoạt động cũng có một số đặc điểm sau:

Cấu trúc đa lớp: Mạng nơ-ron được chia thành làm nhiều lớp và phân vào ba tầng: tầng đầu vào, tầng đầu ra và tầng ẩn. Ở tầng ẩn là nơi chứa nhiều lớp nơ-ron, càng nhiều lớp thì mạng càng "sâu", mỗi lớp các nút nơ-ron và các kết nối giữa chúng.

Trọng số kết nối: Mỗi một cặp kết nối, giữa chúng sẽ có một trọng số tương ứng để biểu thị được mức độ quan trọng của trọng số đó với mạng. Trọng số đó càng cao thì sự ảnh hưởng của trọng số đó càng lớn.

Chức năng kích hoạt: Mỗi một nơ-ron sẽ có một chức năng kích hoạt với mục đích để chuẩn hóa dữ liệu đầu ra của nơ-ron. Chức năng kích hoạt giúp xác định sự ảnh hưởng của thông tin và truyền thông tin trong mạng.

Dữ liệu đầu vào và đầu ra: Dữ liệu sẽ được đưa vào trong mạng thần kinh, đi qua tất cả các lớp để trích xuất đầy đủ các đặc trưng của dữ liệu. Các đặc trưng của dữ liệu sẽ được tổng hợp lại thành kết quả và đưa ra tại đầu ra. Tùy vào mục đích của mạng, đầu ra có thể là kết quả phân lớp hoặc là kết quả dự đoán.

Quá trình đào tạo: Bằng các thuật toán như lan truyền ngược, mô hình được điều chỉnh bộ trọng số để tối ưu hóa được hiệu năng của mô hình.

Ưu điểm:

Khả năng học tự động: Deep Learning có khả năng tự học và trích xuất các đặc trưng phức tạp từ dữ liệu, giúp giảm thiểu sự phụ thuộc vào việc thiết kế đặc trưng thủ công.

Hiệu Suất Cao Trong Nhiều Lĩnh Vực: Deep Learning đã đạt được kết quả ấn tượng trong nhiều lĩnh vực, bao gồm nhận diện hình ảnh và giọng nói, xử lý ngôn ngữ tự nhiên, và thị giác máy tính.

Khả năng thích ứng cao: Deep Learning có thể học hỏi và thích nghi với dữ liệu mới, giúp cải thiện hiệu suất theo thời gian.

Xử lý đa dạng dữ liệu: Deep Learning có thể xử lý hiệu quả các dữ liệu có cấu trúc (như văn bản bảng tính) và phi cấu trúc (như hình ảnh, âm thanh).

Đa Dạng và Linh Hoạt: Deep Learning có thể được áp dụng trong nhiều lĩnh vực khác nhau, từ y tế đến tài chính, và có thể mô hình hóa nhiều loại dữ liệu khác nhau.

Nhược điểm:

Yêu cầu dữ liệu lớn: Để đạt được hiệu suất tốt, Deep Learning thường yêu cầu một lượng lớn dữ liệu huấn luyện. Việc thu thập và chuẩn bị dữ liệu có thể tốn kém và mất nhiều thời gian.

Tính toán phức tạp: Việc huấn luyện các mô hình Deep Learning có thể đòi hỏi nguồn lực tính toán mạnh mẽ, bao gồm cả GPU hoặc các hệ thống tính toán song song.

Yêu cầu phần cứng mạnh mẽ: Việc huấn luyện mô hình Deep Learning thường yêu cầu sử dụng phần cứng mạnh mẽ, đặc biệt là trong trường hợp của các mạng neural sâu và lớn.

Nguy cơ Overfitting: Trong một số trường hợp, các mô hình Deep Learning có thể dễ bị quá mức hóa mô hình (overfitting), đặc biệt là khi dữ liệu đào tạo không đủ.

Deep Learning là một công nghệ đột phá với sức mạnh to lớn có thể thay đổi thế giới theo nhiều cách tích cực. Việc tiếp tục nghiên cứu và phát triển Deep Learning một cách có trách nhiệm sẽ giúp chúng ta khai thác tối đa tiềm năng của nó, mang đến lợi ích cho toàn nhân loại.

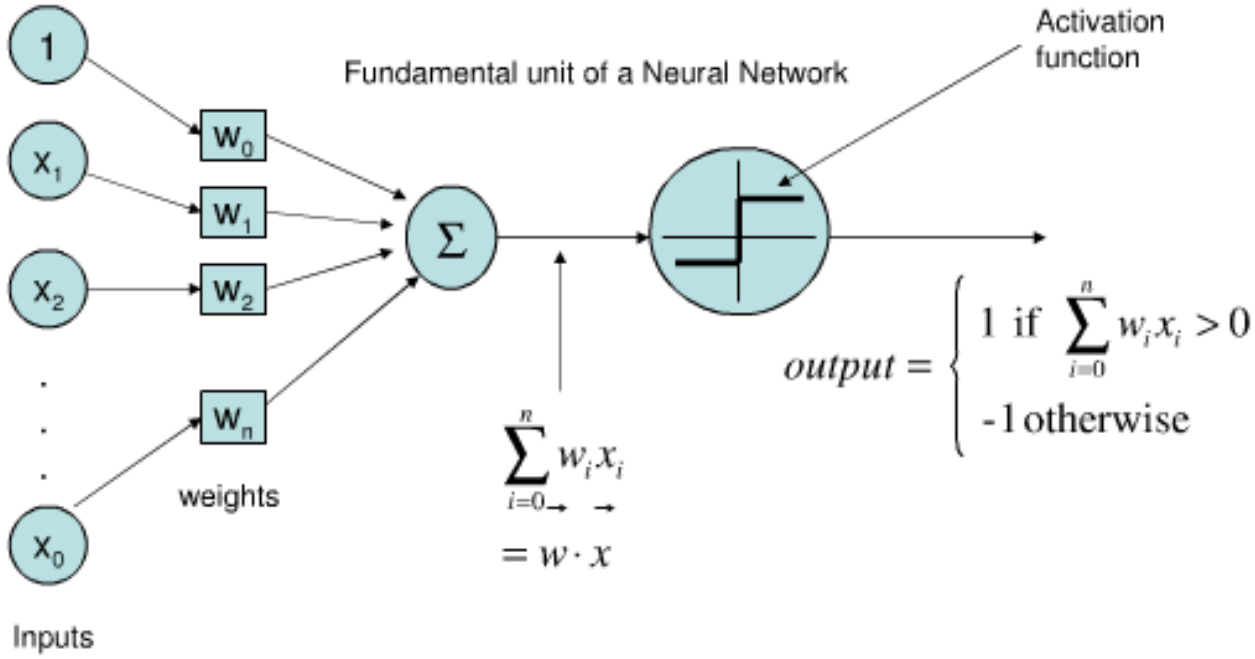
2.2. Neural network

Neural Network là một phương pháp trong trí tuệ nhân tạo được lấy cảm hứng từ cấu trúc và chức năng bộ não của con người. Nó bao gồm các đơn vị tính toán được gọi là nơ-ron, liên kết với nhau bằng các đường dẫn truyền tín hiệu. Mỗi nơ-ron nhận tín hiệu đầu vào từ các nơ-ron khác, xử lý tín hiệu đó thông qua một hàm kích hoạt, và sau đó truyền tín hiệu đầu ra cho các nơ-ron khác. Neural Network có khả năng tự học từ dữ liệu, tự điều chỉnh và cải thiện hiệu suất theo thời gian.

2.2.1. Perceptrons

Perceptron là thuật toán Binary Classification, là nền tảng quan trọng để phát triển Neural Network. Nó được phát hiện bởi Frank Rosenblatt vào những năm 1950 – 1960.

Mỗi một hình tròn trong một lớp (Layer) được gọi là nút (node), đơn vị (unit) hoặc là perceptron.



Hình 2: Cấu tạo của một Perceptron

Trong mỗi một perceptron gồm có:

- Đầu vào (input): Dữ liệu sẽ được đưa vào mô hình.
- Trọng số (Weight): Mỗi đầu vào sẽ được kết nối với một trọng số. Trọng số xác định mức độ ảnh hưởng của mỗi giá trị đầu đầu vào lên kết quả đầu ra của từng node.

- Độ lệch (Bias): Là một hằng số cố định, cho phép dịch chuyển hàm kích hoạt một khoảng cách nhất định.
- Tổng trọng số (Weighted Sum): được biểu diễn như sau:

$$Y=\sum(weight*input) + bias$$

- Hàm kích hoạt (Activation Function): quyết định xem một neural có nên hoạt động hay không và hoạt động với mục đích gì.
- Đầu ra (Output): là kết quả của một node.

Weight và Bias là hai phần quan trọng nhất trong Perceptron. Trong quá trình huấn luyện mô hình, cả hai tham số này để thay đổi với mục đích tìm ra bộ tham số tốt nhất cho mô hình.

Weight (trọng số) được dùng để xác định sự ảnh hưởng của từng đặc trưng trong việc dự đoán giá trị đầu ra. Các trọng số có giá trị gần bằng 0 được coi là có ít trọng số hoặc mức độ liên quan hơn. Những giá trị này ít quan trọng hơn trong quá trình dự đoán so với các đặc trưng có giá trị lớn hơn 0, được gọi là trọng số có giá trị cao hơn. Nếu trọng số dương thì đặc trưng đó có liên quan trực tiếp với giá trị mục tiêu, nếu nó âm, đặc trưng đó có mối quan hệ nghịch đảo với giá trị đích.

Bias (Độ lệch) trong học máy được sử dụng với mục đích là tạo ra sự linh hoạt và điều chỉnh cho mô hình. Trong trường hợp mô hình không thể hiểu được độ phức tạp của mẫu, bias cho phép mô hình dịch chuyển kết quả của phép tính tổng hợp tuyến tính để tối ưu hóa. Cùng với các trọng số của mô hình, chúng giúp cho mô hình xử lý được các dữ liệu phức tạp và tối ưu hóa khả năng học hỏi của mô hình. Vì vậy mà bias có vai trò quan trọng trong việc nâng cao khả năng linh hoạt, thích ứng với dữ liệu và học được các mối quan hệ phức tạp trong quá trình huấn luyện.

Hàm kích hoạt (activation function) là một thành phần quan trọng trong mạng neural, đặc biệt là trong các lớp ẩn của mô hình. Hàm kích hoạt đóng vai trò thiết yếu trong việc nâng cao khả năng học tập và hiệu quả của mạng nơ-ron. Việc hiểu rõ chức năng và cách lựa chọn hàm kích hoạt phù hợp sẽ giúp chúng ta xây dựng mô hình mạng nơ-ron hiệu quả hơn cho bài toán của mình. Dù mạng nơ-ron được mô phỏng dựa trên bộ não của con người,

các nơ-ron kết nối với nhau thông qua các khớp nối, nhưng nếu không có hàm kích hoạt, kết quả đầu ra của mạng nơ-ron sẽ chỉ là phép toán tuyến tính đơn giản giữa đầu vào và trọng số, dẫn đến khả năng học tập bị giới hạn. Bằng cách thêm tính không tuyến tính vào đầu ra của các lớp ẩn, mô hình trở nên phi tuyến tính hơn, hay còn gọi là khả năng phi tuyến tính hóa. Nhờ vậy, mạng nơ-ron có thể học được các mối quan hệ phức tạp giữa các đặc trưng đầu vào. Hàm kích hoạt cung cấp khả năng học các biểu diễn đồng thời của đầu vào. Điều này cho phép mạng neural giải quyết các vấn đề phức tạp và khám phá không gian biểu diễn rộng lớn hơn. Điều này có nghĩa là cho phép mô hình xử lý và học các tương tác phức tạp giữa các đặc trưng đầu vào.

Cách hoạt động của một perceptron:

$$\text{Output} = f\left(\sum_{i=1}^{n} w_i * x_i + b\right)$$

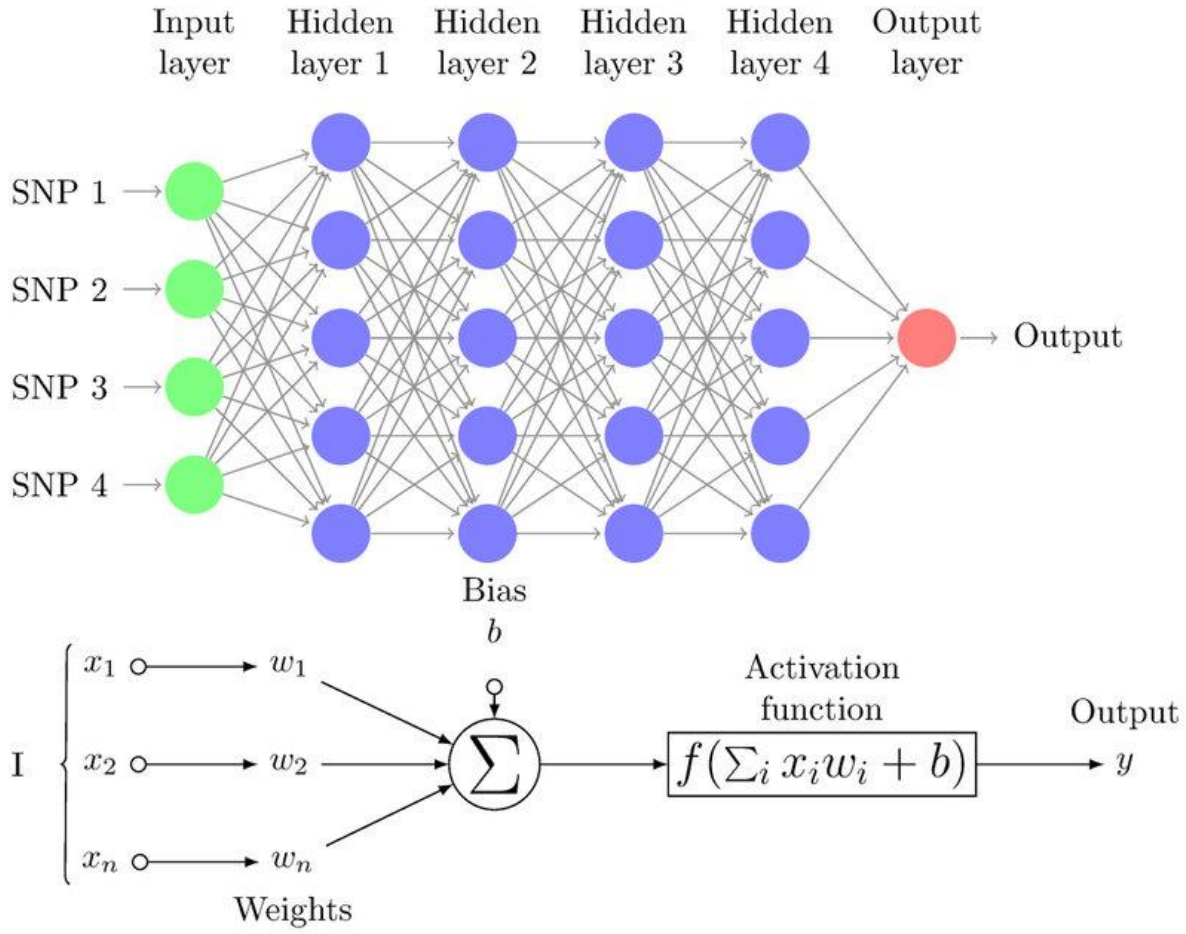
Trong đó:

- $f()$: là hàm kích hoạt
- w_i: là trọng số tương ứng với mỗi đầu vào
- x_i : là các giá trị đầu vào
- b : là bias

Mô hình perceptron bắt đầu bằng phép nhân tất cả các giá trị đầu vào và trọng số của chúng, cộng các giá trị này lại với nhau để tạo ra tổng có trọng số.

Sau đó, tổng có trọng số này được áp dụng cho hàm kích hoạt 'f' để thu được đầu ra mong muốn. Hàm kích hoạt này còn được gọi là hàm bước và được biểu thị bằng 'f'.

2.2.2. Multi-Layer Perceptron



Hình 3: Kiến trúc mạng Neural Network

Neural network (mạng nơ ron) là sự kết hợp của những tầng perceptron hay còn gọi là perceptron đa tầng:

- Tầng input layer (tầng vào): Tầng này nằm bên trái cùng của mạng, thể hiện cho các đầu vào của mạng.
- Tầng output layer (tầng ra): Là tầng bên phải cùng và nó thể hiện cho những đầu ra của mạng.

- Tầng hidden layer (tầng ẩn): Tầng này nằm giữa tầng vào và tầng ra, nó thể hiện cho quá trình suy luận logic của mạng.

Thuật toán Backpropagation là một công cụ mạnh mẽ trong học sâu, cho phép huấn luyện các mạng nơ ron hiệu quả để giải quyết nhiều loại bài toán. Thuật toán Backpropagation sử dụng phương pháp gradient descent (hạ dốc) để cập nhật các tham số. Gradient descent là một thuật toán tối ưu hóa lặp đi lặp lại, di chuyển các tham số theo hướng giảm thiểu hàm mục tiêu (lỗi).

Thuật toán Backpropagation, thực hiện theo các bước như sau:

- Khởi tạo trọng số: Trước khi huấn luyện, các trọng số của mạng neural được khởi tạo ngẫu nhiên với các giá trị nhỏ.

- Feedforward (lan truyền tiến): Với mỗi mẫu dữ liệu huấn luyện, đầu vào được đưa vào mạng neural, các chức năng kích hoạt sẽ tính toán đầu ra của mỗi lớp bắt đầu từ lớp đầu vào và kết thúc ở lớp đầu ra.

- Tính toán độ lệch: So sánh đầu ra được dự đoán với giá trị thực tế để tính toán độ lệch của mỗi perceptron.

- Backpropagation (lan truyền ngược): Tính gradient của độ lệch đối với trọng số cuối cùng của mạng bằng cách sử dụng đạo hàm của hàm mất mát theo đầu ra của mạng và đầu ra thực tế. Sau đó lan truyền ngược từ lớp cuối cùng đến lớp đầu tiên, tính toán gradient của độ lệch đối với từng trọng số ở mỗi lớp, sử dụng quy tắc chuỗi đạo hàm (chain rule).

- Cập nhật trọng số: Sử dụng đạo hàm của hàm mất mát so với các tham số để điều chỉnh trọng số và bias của các kết nối trong mạng theo hướng làm giảm độ lệch

- Lặp lại quá trình: Quá trình feedforward, tính toán độ lỗi, backpropagation và cập nhật trọng số được lặp lại cho tất cả các mẫu dữ liệu huấn luyện. Quá trình này tiếp tục cho đến khi một số tiêu chí dừng được đạt được, như số lần lặp đã được thiết lập trước, hoặc độ lỗi đạt đến một ngưỡng mong muốn.

- Kiểm tra và đánh giá: Sử dụng dữ liệu kiểm tra để đánh giá hiệu suất của mô hình sau khi huấn luyện. Đánh giá các độ lệch và chỉ số hiệu suất bằng các thước đo như độ chính xác (accuracy), Precision hoặc Recall để đánh giá khả năng tổng quát hóa của mô hình.

Do đó, mô hình perceptron nhiều lớp được coi là nhiều mạng nơ ron nhân tạo có nhiều lớp khác nhau trong đó hàm kích hoạt không duy trì tuyến tính, tương tự như mô hình perceptron một lớp. Thay vì tuyến tính, hàm kích hoạt có thể được thực thi dưới dạng sigmoid, TanH, ReLU, v.v. để triển khai.

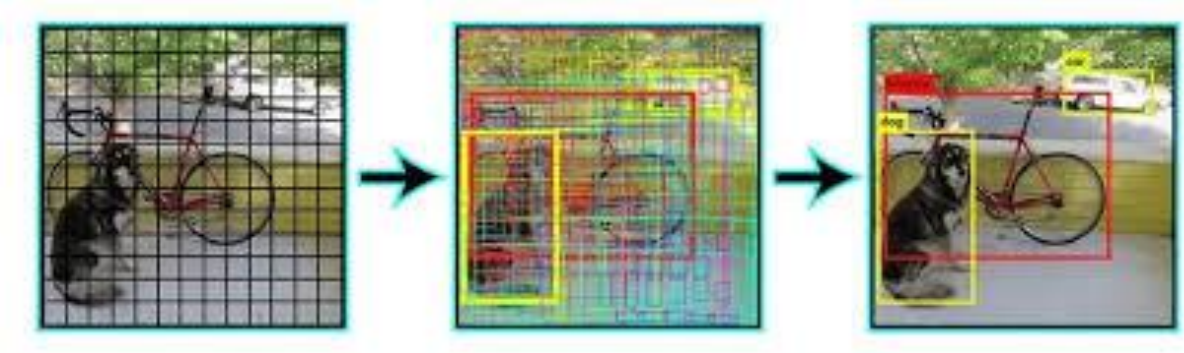
2.3. Object Detection – Phát hiện đối tượng

Phát hiện đối tượng là một lĩnh vực quan trọng trong thị giác máy tính và xử lý hình ảnh, liên quan đến việc xác định vị trí và phân loại các đối tượng trong hình ảnh hoặc video. Nói một cách đơn giản hơn, Object Detect là vẽ một khung hình chữ nhật xung quanh từng đối tượng quan tâm trong ảnh và gán cho chúng một nhãn.

Hình 4: Thuật toán Phát hiện đối tượng

Các thuật toán phát hiện đối tượng có thể được chia thành hai loại chính: Phát hiện vật thể truyền thống (bbox-level localization) và Phân đoạn vật thể (pixel-level hoặc mask-level localization)

2.3.1. Phát hiện vật thể truyền thống (bbox-level localization)



Hình 5: Phát hiện vật thể truyền thống

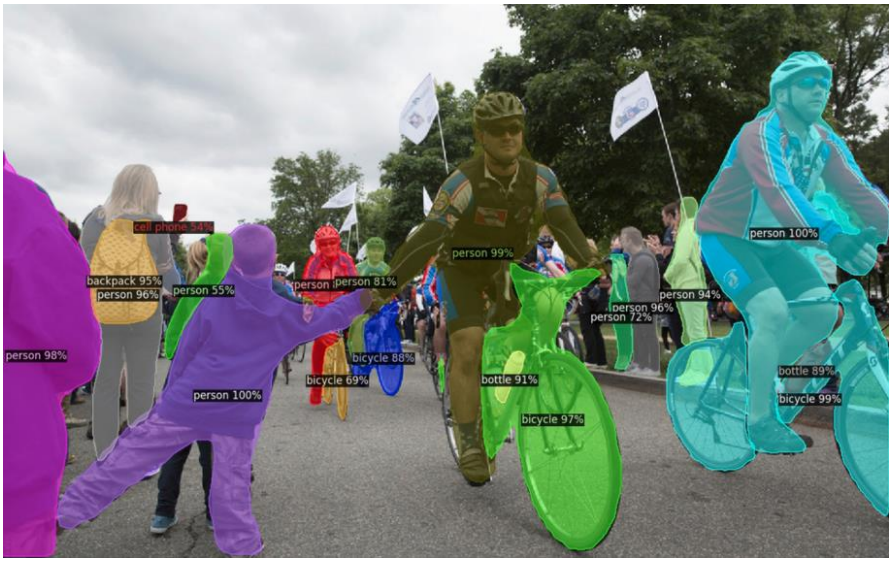
Phát hiện vật thể truyền thống (bbox-level localization) là nhiệm vụ xác định vị trí và kích thước của các vật thể trong hình ảnh hoặc video. Toàn bộ ảnh được xử lý trong một lần, giúp tiết kiệm tài nguyên tính toán. Tuy nhiên, độ chính xác của phương pháp này thường thấp hơn các phương pháp khác và ít hiệu quả hơn trong việc phát hiện các đối tượng nhỏ.

Các thuật toán dạng này phù hợp cho các ứng dụng thời gian thực có tài nguyên hạn chế.

Các phương pháp truyền thống thường sử dụng các thuật toán như:

- Selective Search: Phân chia hình ảnh thành các vùng đề xuất và sau đó áp dụng thuật toán phân loại để xác định xem mỗi vùng có chứa vật thể hay không.
- Sliding Window: Di chuyển một cửa sổ có kích thước cố định qua hình ảnh và áp dụng thuật toán phân loại cho từng cửa sổ.

2.3.2. Phân đoạn vật thể (pixel-level hoặc mask-level localization)



Hình 6: Phân đoạn vật thể

Phân đoạn vật thể (pixel-level hoặc mask-level localization) là nhiệm vụ xác định ranh giới của các vật thể trong hình ảnh hoặc video. Lần quét đầu tiên dùng để tạo ra một tập các vùng đề xuất - những vị trí tiềm năng chứa đối tượng. Lần quét thứ hai dùng để tinh chỉnh các vùng đề xuất này và đưa ra dự đoán cuối cùng. Cách tiếp cận này cho độ chính xác cao hơn phát hiện đối tượng một lần quét nhưng cũng tốn nhiều tính toán hơn.

Các phương pháp truyền thống thường sử dụng các thuật toán như:

- Level Sets: Sử dụng một hàm để biểu diễn ranh giới của vật thể và sau đó tối ưu hóa hàm này để tìm ranh giới chính xác nhất.
- Graph Cuts: Xác định một đồ thị với các nút biểu diễn các pixel trong hình ảnh và các cạnh biểu diễn mối quan hệ giữa các pixel. Sau đó, áp dụng thuật toán cắt đồ thị để tìm ranh giới của vật thể.

2.3.3. So sánh giữa hai thuật toán trên:

Phát hiện vật thể truyền thống (bbox-level localization):

- Mục tiêu: Xác định vị trí và kích thước của các vật thể trong hình ảnh hoặc video.
- Đầu ra: Hộp bao (bounding box) bao quanh vật thể.
- Độ chính xác: Thấp hơn. Do chỉ xác định vị trí và kích thước chung của vật thể, có thể xảy ra sai sót khi vật thể bị che khuất hoặc có hình dạng phức tạp.
- Tốc độ: Nhanh hơn. Do chỉ cần xử lý một số vùng đề xuất thay vì toàn bộ hình ảnh.
- Khả năng khái quát hóa: Tốt hơn. Do có thể áp dụng cho nhiều loại vật thể khác nhau với ít dữ liệu huấn luyện hơn.

Phân đoạn vật thể (pixel-level hoặc mask-level localization):

- Mục tiêu: Xác định ranh giới của các vật thể trong hình ảnh hoặc video.
- Đầu ra: Mặt nạ (mask) cho biết pixel nào thuộc về vật thể và pixel nào không.
- Độ chính xác: Cao hơn. Do xác định ranh giới chính xác của vật thể, ít bị ảnh hưởng bởi các yếu tố như che khuất hay hình dạng phức tạp.
- Tốc độ: Chậm hơn. Do cần xử lý toàn bộ hình ảnh để xác định ranh giới của vật thể.
- Khả năng khái quát hóa: Tệ hơn. Do cần nhiều dữ liệu huấn luyện hơn để học cách phân biệt ranh giới của các vật thể khác nhau trong các điều kiện khác nhau.

2.3.4. Mô hình điển hình:

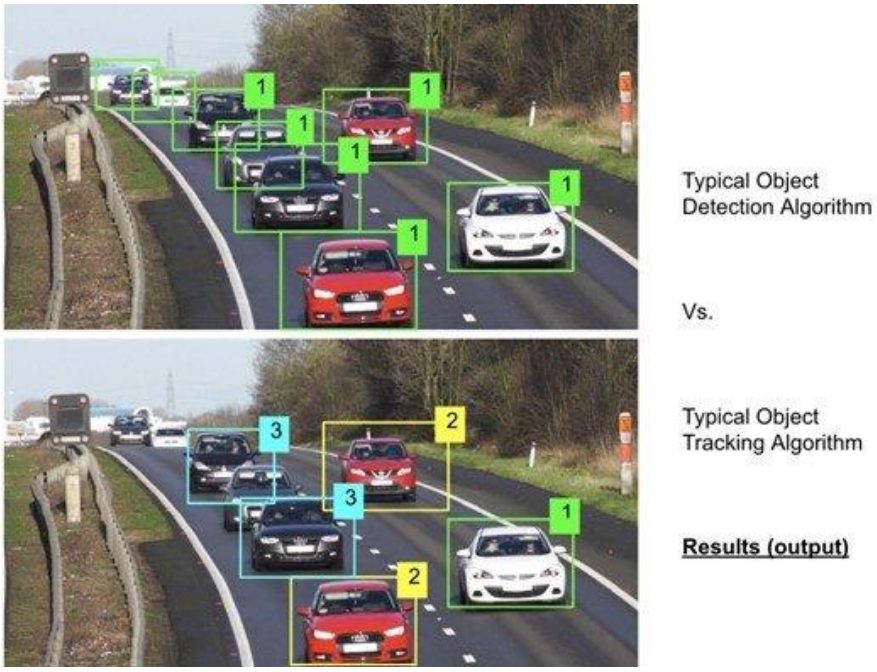
YOLO (You Only Look Once): Một mô hình hiệu quả sử dụng mạng nơ-ron tích chập để phát hiện đối tượng trong thời gian thực.

Mask R-CNN (Region-based Convolutional Neural Network for Object Detection and Segmentation): Một mô hình học sâu tiên tiến có thể phát hiện và phân đoạn các đối tượng trong hình ảnh.

FCNs (Fully Convolutional Networks): Sử dụng mạng nơ-ron tích chập để dự đoán nhãn cho mỗi pixel trong hình ảnh, cho biết pixel đó có thuộc về vật thể hay không.

2.4. Object Tracking

Theo dõi đối tượng là một lĩnh vực trong thị giác máy tính (Computer Vision) có khả năng xác định các đối tượng trong video và diễn giải chúng dưới dạng tập hợp quỹ đạo. Object Tracking sẽ phát hiện đối tượng trong từng khung hình, sau đó liên kết chúng theo thời gian thực.



Hình 7: Object Detection vs Object Tracking

2.4.1. Các giai đoạn của Object Tracking:

- Giai đoạn #1: Chỉ định hoặc Phát hiện: Các mục tiêu quan tâm được ghi nhận và đánh dấu trong giai đoạn chỉ định. Thuật toán phân tích các khung đầu vào để xác định các đối tượng thuộc lớp đích. Hộp giới hạn được sử dụng để thực hiện phát hiện như một phần của thuật toán.

- Giai đoạn #2: Chuyển động: Các thuật toán trích xuất tính năng phân tích các phát hiện để trích xuất các tính năng xuất hiện và tương tác. Trong hầu hết các trường hợp, bộ dự đoán chuyển động được sử dụng để dự đoán các vị trí tiếp theo của từng mục tiêu được theo dõi.

- Giai đoạn #3: Nhớ lại : Dự đoán tính năng được sử dụng để tính điểm tương tự giữa các câu đối phát hiện. Những điểm số đó sau đó được sử dụng để liên kết các phát hiện thuộc về cùng một mục tiêu. ID được gán cho các phát hiện tương tự và các ID khác nhau được áp dụng cho các phát hiện không thuộc cặp.

Các kỹ thuật liên kết các phát hiện theo thời gian thực: Giúp xác định các khung hình tiếp theo để tạo ra quỹ đạo di chuyển của các đối tượng trong khung hình.

- Luồng Hungary (Hungarian Algorithm): Sử dụng thuật toán tối ưu hóa để gán các phát hiện trong khung hình hiện tại với các đối tượng được theo dõi trong khung hình trước.

- Bộ lọc Kalman: Sử dụng mô hình dự đoán để ước tính vị trí của các đối tượng trong khung hình tiếp theo và gán các phát hiện với các đối tượng được dự đoán.

- Mạng nơ-ron: Sử dụng mạng nơ-ron để học cách liên kết các phát hiện theo thời gian.

- Kỹ thuật lai (Hybrid Techniques): Kết hợp các kỹ thuật khác nhau để tận dụng ưu điểm của từng phương pháp.

2.4.2. DeepSORT (Deep Object Tracking and Re-identification)

DeepSORT (Deep Object Tracking and Re-identification): là một thuật toán theo dõi đối tượng theo thời gian thực được phát triển bởi nhóm nghiên cứu của Abe Ross tại Đại học Stanford.

Nguyên lý hoạt động của DeepSORT:

- Phát hiện đối tượng: Sử dụng các mô hình phát hiện đối tượng (Object Detection) để xác định vị trí và gán ID cho từng vật thế.

- Trích xuất đặc trưng: Sử dụng mạng nơ-ron tích chập để trích xuất các đặc trưng biểu diễn cho mỗi đối tượng được phát hiện.

- Liên kết các phát hiện: Sử dụng thuật toán liên kết các phát hiện theo thời thực để liên kết các đặc trưng được phát hiện trong khung hình hiện tại với các đối tượng được theo dõi trong khung hình trước dựa trên khoảng cách (Mahalanobis) giữa các đặc trưng.

- Cập nhật trạng thái: Sử dụng bộ lọc Kalman để ước tính vị trí, vận tốc và trạng thái khác của các đối tượng theo thời gian thực

- Xử lý các trường hợp che khuất: Sử dụng kỹ thuật Re-identification để xác định lại các đối tượng bị che khuất hoặc xuất hiện/biến mất đột ngột dựa trên sự tương đồng giữa các đặc trưng.

Khoảng cách Mahalanobis: là một thước đo thống kê được sử dụng để đo lường khoảng cách giữa một điểm và một đám mây điểm trong không gian đa chiều. Nó được định nghĩa dựa trên phép biến đổi Mahalanobis, giúp chuẩn hóa dữ liệu và làm cho các chiều dữ liệu có tầm quan trọng như nhau.

Khoảng cách Mahalanobis của vec tơ $x = (x_1, x_2, x_3, \dots, x_n)^T$ so với một nhóm có trung bình là $\mu = (\mu_1, \mu_2, \mu_3, \dots, \mu_n)^T$ và ma trận hiệp phương sai

Công thức: $D(x, \mu) = \sqrt{(x - \mu)' \Sigma^{-1} (x - \mu)}$:

- x: Điểm cần đo khoảng cách.

- μ: Trung tâm của đám mây điểm.

- Σ: Ma trận hiệp phương sai của đám mây điểm.

- Σ^{-1}: Ma trận nghịch đảo của ma trận hiệp phương sai.

Phép biến đổi Mahalanobis: Phép biến đổi này giúp chuẩn hóa dữ liệu bằng cách chia mỗi chiều dữ liệu cho độ lệch chuẩn của nó. Nhờ vậy, các chiều dữ liệu có tầm quan trọng như nhau trong việc tính toán khoảng cách.

Ma trận hiệp phương sai: Ma trận hiệp phương sai thể hiện sự tương quan giữa các chiều dữ liệu. Nó cho biết mức độ biến thiên của mỗi chiều dữ liệu và mức độ liên quan giữa các chiều dữ liệu.

Ma trận nghịch đảo: Ma trận nghịch đảo của ma trận hiệp phương sai được sử dụng để điều chỉnh ảnh hưởng của sự tương quan giữa các chiều dữ liệu.

Ưu điểm:

- Khả năng xử lý dữ liệu đa chiều: Khoảng cách Mahalanobis có thể được sử dụng để đo lường khoảng cách trong không gian đa chiều, nơi có nhiều hơn hai chiều dữ liệu.

- Chống nhiễu: Khoảng cách Mahalanobis ít bị ảnh hưởng bởi nhiễu dữ liệu so với các thước đo khoảng cách khác như khoảng cách Euclidean.

- Khả năng điều chỉnh: Ma trận hiệp phương sai có thể được điều chỉnh để phù hợp với các đặc điểm cụ thể của dữ liệu, giúp cải thiện độ chính xác của phép đo.

Nhược điểm:

- Yêu cầu dữ liệu: Khoảng cách Mahalanobis cần ma trận hiệp phương sai của dữ liệu, có thể khó tính toán cho các tập dữ liệu lớn hoặc có nhiều chiều dữ liệu.

- Giải thích: Việc giải thích kết quả của khoảng cách Mahalanobis có thể khó khăn hơn so với các thước đo khoảng cách khác.

Tuy DeepSORT là thuật toán phức tạp hơn so với mạng nơ-ron và bộ lọc Kalman, nhưng DeepSORT vẫn có tốc độ xử lý nhanh và độ chính xác cao trong việc theo dõi các đối tượng trong thời gian thực, ngay cả khi đối tượng đó bị che khuất. DeepSORT còn cho thấy

khả năng thích ứng cao với môi trường không có được điều kiện thuận lợi, như ánh sáng yếu.

2.5. Pre-trained model

Pre-trained models là một loại mô hình máy học đã trải qua quá trình huấn luyện trước đó trên một lượng lớn dữ liệu. Trong ngữ cảnh của Object Detection, dữ liệu thường là các bộ dữ liệu chứa hình ảnh với nhãn định danh các đối tượng trong ảnh. Các mô hình này thường được huấn luyện trên các tài nguyên tính toán mạnh mẽ và được cung cấp sẵn cho cộng đồng để sử dụng.

Một số mô hình trong lớn trong Computer Vision như Yolo, GAN, ResNET đã đạt được những thành tựu ấn tượng trong các nhiệm vụ liên quan đến hình ảnh và video, như phân loại hình ảnh, tạo hình ảnh và video, ...

Việc sử dụng các mô hình đã được huấn luyện trước:

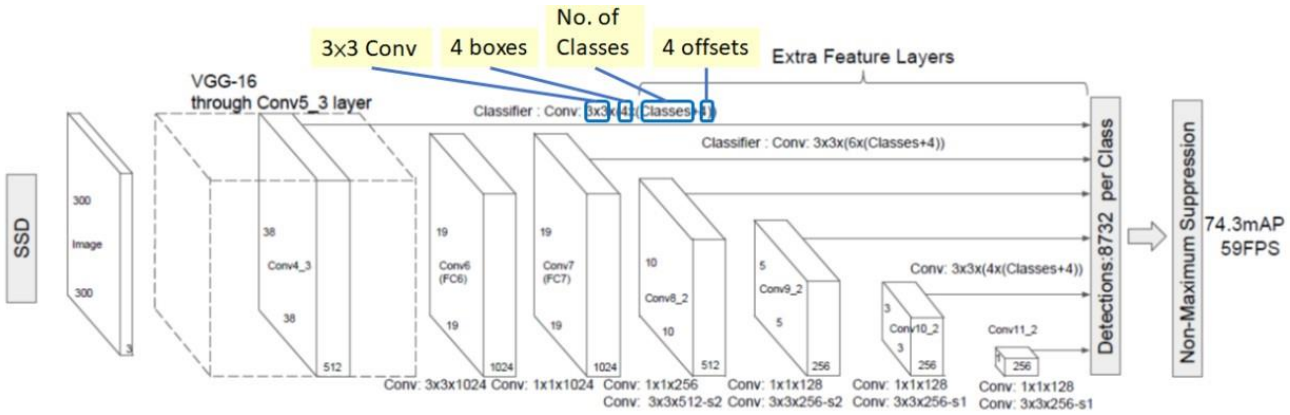
- Giảm thời gian đào tạo: tận dụng trọng số và tính năng được đào tạo trước từ một tập dữ liệu lớn giúp chúng ta đào tạo mô hình nhanh hơn nhiều so với bắt đầu từ đầu. Điều này giúp tiết kiệm tài nguyên tính toán và thời gian.

- Mô hình được đào tạo trước đã học được các đặc trưng của hình ảnh ở mức thấp và trung bình. Điều này cung cấp một nền tảng vững chắc cho mô hình tùy chỉnh của bạn để xây dựng, thường dẫn đến hiệu suất tốt hơn.

- Yêu cầu dữ liệu giảm: Việc đào tạo mô hình thường đòi hỏi một lượng lớn dữ liệu được dán nhãn. Mô hình được đào tạo trước giúp giải quyết thách thức này bằng cách cho phép chúng ta đào tạo với các tập dữ liệu nhỏ hơn cụ thể cho nhiệm vụ của mình. Điều này đặc biệt có lợi khi việc thu thập một lượng lớn dữ liệu được dán nhãn là tốn kém hoặc tốn thời gian.

2.5.1. SSD (Single Shot MultiBox Detector):

Mô hình SSD (Single Shot MultiBox Detector) là một kiến trúc mạng nơ-ron được giới thiệu lần đầu tiên trong bài báo "SSD: Single-Shot MultiBox Detection" của Wei Liu et al. vào năm 2016. Điểm đặc biệt của SSD so với các mô hình phát hiện đối tượng khác như R-CNN, Faster R-CNN là nó chỉ sử dụng một mạng nơ-ron duy nhất để thực hiện cả hai nhiệm vụ trích xuất đặc trưng và phát hiện đối tượng. Nhờ vậy, SSD có tốc độ xử lý nhanh hơn đáng kể so với các mô hình khác.



Hình 8: Kiến trúc của mạng SSD

Kiến trúc của SSD bao gồm hai thành phần chính:

- Mạng cơ sở (Base Network): Sử dụng một mạng nơ-ron được đào tạo trước như VGG16, ResNet để trích xuất đặc trưng ảnh.

- Các lớp neuron phụ trợ (Auxiliary Layers): Được thêm vào các tầng khác nhau của mạng cơ sở để dự đoán vị trí và lớp của các đối tượng trong ảnh.

Ưu điểm:

- Tốc độ nhanh: SSD có thể xử lý ảnh với tốc độ hàng chục khung hình mỗi giây (FPS), phù hợp cho các ứng dụng thời gian thực.

- Độ chính xác cao: SSD đạt được độ chính xác cao trong các bài toán phát hiện đối tượng tiêu chuẩn.

- Dễ triển khai: SSD có thể được triển khai dễ dàng trên các nền tảng phần cứng khác nhau.

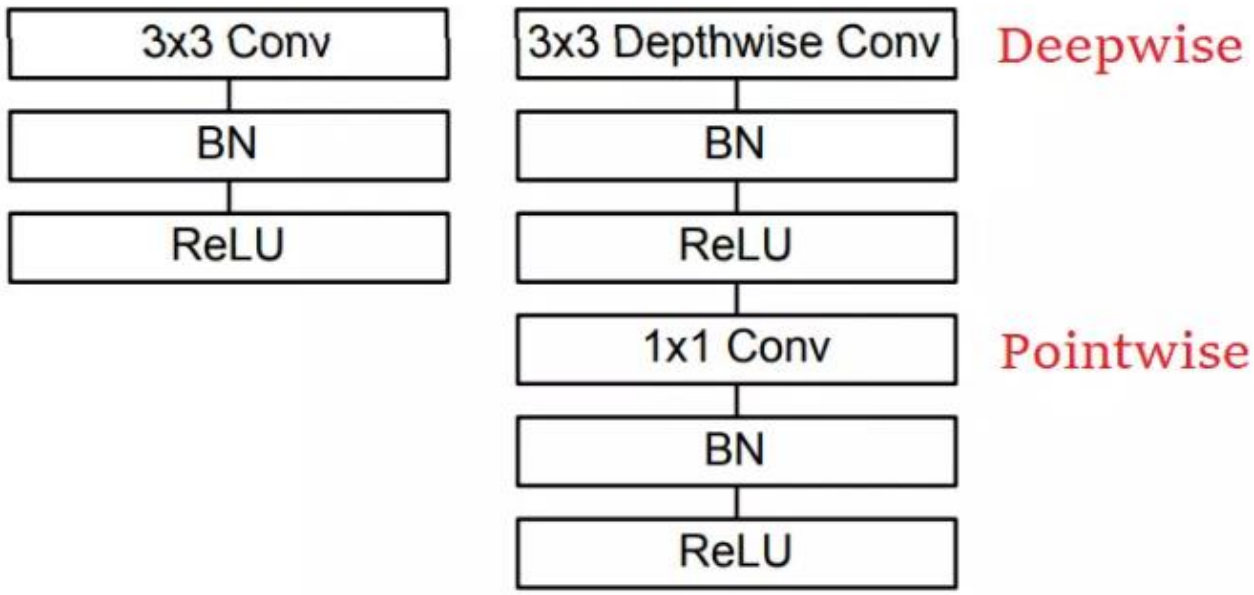
Nhược điểm:

- Độ nhạy cảm với kích thước đối tượng: SSD có thể gặp khó khăn trong việc phát hiện các đối tượng có kích thước nhỏ.

- Độ nhạy cảm với sự thay đổi hình dạng đối tượng: SSD có thể gặp khó khăn trong việc phát hiện các đối tượng có hình dạng khác biệt so với dữ liệu huấn luyện.

2.5.2. Mô hình Mobile NET:

Mô hình Mobile NET là kiến trúc CNN nhẹ được thiết kế để chạy hiệu quả trên các thiết bị di động, các thiết bị yếu cần giảm thiểu đáng kể số phép tính cần thiết. Kiến trúc của mô hình MobileNET được thiết kế dành riêng cho các thiết bị di động có nguồn lực tính toán hạn chế. Nó được ra mắt lần đầu tiên vào năm 2017 bởi nhóm nghiên cứu Google AI và nhanh chóng trở thành một trong những mô hình phân loại ảnh phổ biến nhất cho các ứng dụng di động.



Hình 9: Trái: Standard Convolutional layer với batchnorm và hàm kích hoạt ReLU; Phải: Depthwise Separable convolutions với Deepwise và Pointwise layers cùng với batchnorm và hàm kích hoạt ReLU

Depthwise Separable Convolutions chia CNN cơ bản ra làm hai phần:

- Deepwise Convolution (Tích chập theo chiều sâu): thực hiện phép tích chập trên từng kênh (channel) riêng biệt để trích xuất được thông tin không gian từ mỗi kênh đầu vào. Để có thể thực hiện được việc này, nó sử dụng nhiều bộ lọc (một bộ lọc cho mỗi kênh) nhưng mỗi bộ lọc chỉ có một kênh. Điều này giúp giảm đáng kể số lượng phép tính so với phép toán tích chập thông thường với nhiều kênh trong bộ lọc.

- Pointwise Convolution (Tích chập theo chiều điểm): thực hiện phép tích chập giữa các kênh nhằm kết hợp các đặc trưng được trích xuất và giảm số lượng kênh nếu cần. Nó sử

dụng phép toán tích chập 1x1 với một số lượng kênh nhất định tùy thuộc vào thiết kế của mạng.

Depthwise Separable Convolutions làm giảm đáng kể số lượng trọng số cần học mà vẫn giữ được khả năng trích xuất đặc trưng không gian và làm tăng hiệu quả về tính toán.

Ưu điểm:

- Hiệu quả tính toán: MobileNet có thể đạt được độ chính xác tương đương với các mô hình CNN phức tạp hơn nhưng với lượng tính toán thấp hơn nhiều, making it ideal for mobile devices.

- Kích thước nhỏ gọn: MobileNet có kích thước nhỏ gọn, giúp giảm thời gian tải xuống và lưu trữ trên thiết bị di động.

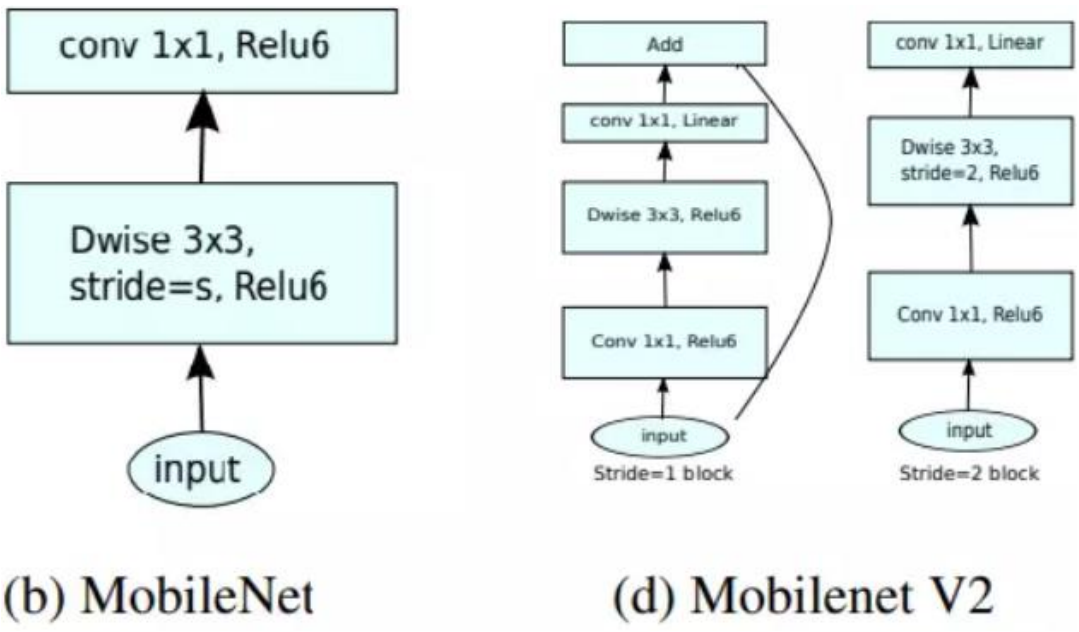
- Độ chính xác cao: Mặc dù nhẹ, MobileNet vẫn đạt được độ chính xác cao trong các tác vụ nhận dạng và phân loại ảnh.

2.5.3. Mô hình MobileNETv2:

Trong mô hình MobileNETv2 còn đề xuất thêm hai kỹ thuật khác gồm:

- Linear bottlenecks: sử dụng các lớp convolutions 1x1 để thu hẹp kích thước kênh của dữ liệu đầu vào, sau đó mở rộng kích thước kênh của các lớp convolutions 3x3 để thực hiện các phép tính toán chính trong mạng nơ-ron và sử dụng đó giúp giảm số lượng phép tính cần thiết cho các phép toán convolutions.

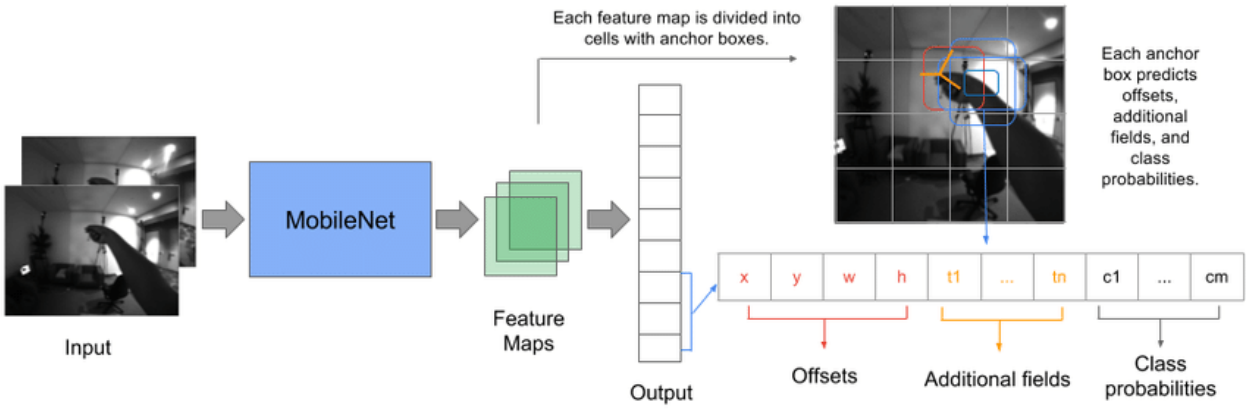
- Inverted Residual Block: sử dụng một lớp pointwise convolution với số lượng kênh lớn hơn để mở rộng số lượng kênh đầu vào. Sau đó sử dụng kĩ thuật Depthwise Separable Convolutions để tính toán và sử dụng một lớp point convolution khác để thu hẹp số lượng kênh xuống kính thước mong muốn. Cuối cùng là lấy kết quả đầu ra đó cộng với đầu vào để có được kết quả cuối cùng.



Hình 10: So sánh giữa MobileNET và MobileNET V2

2.5.4. Mô hình SSD MobileNET:

SSD (Single Shot MultiBox Detector) và MobileNet V2 là hai kiến trúc nổi bật trong lĩnh vực thị giác máy tính (computer vision), đặc biệt là trong bài toán phát hiện đối tượng (object detection) và phân loại ảnh (image classification). Việc kết hợp hai mô hình này tạo ra SSD MobileNet V2, một mô hình mạnh mẽ, hiệu quả và nhẹ, phù hợp với các thiết bị có tài nguyên hạn chế như thiết bị di động và nhúng.



Hình 11: Cấu trúc mạng SSD MobileNET V2

SSD MobileNet V2 là sự kết hợp giữa SSD và MobileNet V2, thừa hưởng tất cả các ưu điểm của cả hai mô hình:

Hiệu suất cao: SSD MobileNet V2 có khả năng phát hiện đối tượng với độ chính xác cao nhờ việc kết hợp các feature maps của SSD và các kỹ thuật tối ưu của MobileNet V2.

Tốc độ nhanh: MobileNet V2 với các depthwise separable convolutions và inverted residuals giúp giảm thiểu số lượng phép tính, tăng tốc độ xử lý.

Nhẹ và tiết kiệm tài nguyên: Thiết kế của MobileNet V2 làm cho SSD MobileNet V2 trở thành một mô hình nhẹ, phù hợp với các thiết bị di động và nhúng.

2.5.5. Feature Pyramid Network Lite - FPNLite:

Feature Pyramid Network - Mạng Cấu Trúc Kim Tự Tháp khai thác cấu trúc kim tự tháp đa tầng, phân cấp nhất của các mạng tích chập sâu để xây dựng các kim tự tháp đặc trưng với chi phí bổ sung không đáng kể. Một kiến trúc từ trên xuống với các kết nối bên được phát triển để xây dựng bản đồ đặc trưng ngữ nghĩa cấp cao ở mọi tỷ lệ. Kiến trúc này cho thấy sự cải thiện đáng kể như một trích xuất đặc trưng chung trong một số ứng dụng. Sử dụng FPN trong hệ thống Faster R-CNN cơ bản, đạt kết quả tiên tiến theo mô hình đơn trên bảng đánh giá phát hiện COCO, vượt trội hơn tất cả các bản ghi mô hình đơn tồn tại bao gồm cả những người chiến thắng thách thức COCO 2016.

FPN Lite là một phiên bản nhẹ hơn của kiến trúc FPN (Feature Pyramid Network) ban đầu, được thiết kế để sử dụng trên các thiết bị di động và các thiết bị nhúng có tài nguyên tính toán hạn chế.

Cấu trúc của FPN Lite bao gồm ba thành phần chính:

- Mạng cơ sở: Mạng cơ sở là một mạng nơ-ron được sử dụng để trích xuất các đặc điểm từ hình ảnh đầu vào. FPN Lite thường sử dụng MobileNet hoặc VGG16 làm mạng cơ sở.

- Bộ phát hiện: Bộ phát hiện là một mạng nơ-ron được sử dụng để phát hiện các đối tượng trong hình ảnh. FPN Lite thường sử dụng SSD (Single Shot MultiBox Detector) làm bộ phát hiện.

- Bộ trích xuất đặc trưng: Bộ trích xuất đặc trưng là một mô-đun kết hợp các đặc trưng từ các cấp độ khác nhau của mạng cơ sở để tạo ra một kim tự tháp đặc trưng đa cấp. Kim tự tháp đặc trưng này sau đó được sử dụng bởi bộ phát hiện để phát hiện các đối tượng ở các kích thước khác nhau.

FPN Lite có một số ưu điểm so với các kiến trúc phát hiện đối tượng khác, bao gồm:

- Hiệu suất: FPN Lite rất hiệu quả và có thể chạy trên các thiết bị di động và các thiết bị nhúng có tài nguyên tính toán hạn chế.

- Độ chính xác: FPN Lite có độ chính xác cao và có thể phát hiện các đối tượng với độ chính xác cao.

- Khả năng khái quát hóa: FPN Lite có khả năng khái quát hóa tốt và có thể được sử dụng để phát hiện các đối tượng trong nhiều miền khác nhau.

FPN Lite thường được kết hợp với các mạng khác để tăng cường khả năng xử lý các đặc trưng đa tỷ lệ một cách hiệu quả mà không đòi hỏi tài nguyên tính toán lớn.

2.6. Hàm kích hoạt ReLU (Rectified Linear Unit):

Hàm kích hoạt ReLU (Rectified Linear Unit) là một hàm phi tuyến phổ biến được sử dụng trong mạng nơ-ron nhân tạo, được giới thiệu lần đầu tiên bởi Vinod Nair và Geoffrey Hinton trong bài báo "Rectified Linear Units for Improved MNIST Classification" vào năm 2011.

Công thức: f(x) = max(0, x):

ReLU trả về giá trị đầu vào x nếu nó lớn hơn hoặc bằng 0, và trả về 0 nếu x nhỏ hơn 0.

Ưu điểm:

- Tính đơn giản: ReLU là một hàm toán học đơn giản và dễ tính toán, giúp tăng tốc độ đào tạo mạng nơ-ron.

- Tính phi tuyến: ReLU là một hàm phi tuyến, cho phép mạng nơ-ron học được các mối quan hệ phức tạp giữa dữ liệu đầu vào và đầu ra.

- Giảm thiểu vấn đề biến mất gradient: ReLU giúp giảm thiểu vấn đề biến mất gradient, một hiện tượng có thể khiến mạng nơ-ron khó học được các đặc trưng phức tạp.

Nhược điểm:

- Vấn đề "neuron chết": Nếu giá trị đầu vào x luôn nhỏ hơn 0, neuron sử dụng hàm ReLU sẽ luôn tạo ra giá trị 0, khiến nó "chết" và không đóng góp vào việc học tập của mạng nơ-ron.

- Độ nhạy cảm với giá trị khởi tạo: Hiệu suất của ReLU có thể phụ thuộc vào giá trị khởi tạo ban đầu của trọng số mạng nơ-ron.

2.7. Các phương pháp đánh giá:

2.7.1. Accuracy (Độ chính xác):

- Độ chính xác (Accuracy) là một thước đo phổ biến để đánh giá hiệu suất của mô hình học máy, đặc biệt là trong các bài toán phân loại.

- $Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$

- Nhược điểm của cách đánh giá này là chỉ cho ta biết được bao nhiêu phần trăm lượng dữ liệu được phân loại đúng mà không chỉ ra được cụ thể mỗi loại được phân loại như thế nào, lớp nào được phân loại đúng nhiều nhất hay dữ liệu của lớp nào thường bị phân loại nhầm nhất vào các lớp khác.

2.7.2. Precision:

- Precision là thước đo dùng để đánh giá chất lượng của mô hình.

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

- Precision cho biết khả năng của mô hình chỉ định các vùng đối tượng mà nó dự đoán là chứa đối tượng một cách chính xác.

2.7.3. Recall:

- Recall là một metric đánh giá khác để đo lường khả năng của mô hình trong việc bắt kịp tất cả các đối tượng thực tế.

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

- Recall cho biết mức độ mà mô hình có thể bắt kịp tất cả các đối tượng thực tế, tránh việc bỏ sót một đối tượng có thể gây ra hậu quả nghiêm trọng.

- Cả ba phương pháp này thường được dùng chung với nhau đem lại một cách nhìn tổng quan hơn về hiệu suất của mô hình.

2.8. Phương pháp lượng tử hoá mô hình (Model Quantization):

Phương pháp lượng tử hoá mô hình (Model Quantization) là một kỹ thuật làm giảm kích thước mô hình và cải thiện hiệu năng khi chạy trên các thiết bị có tài nguyên hạn chế, chẳng hạn như điện thoại di động, thiết bị nhúng, và các thiết bị IoT. Thay vì sử dụng giá

trị float 32 tiêu chuẩn, lượng tử hóa mô hình sẽ biểu diễn các giá trị này bằng định dạng số ít bit hơn, chẳng hạn như float 16 hoặc int8. Việc thay đổi định dạng này có thể làm ảnh hưởng để độ chính xác và mốt ố sai sót trong kết quả dự đoán nhưng thường không đáng kể và có thể được bù đắp bằng các kỹ thuật tối ưu hóa khác.

Lợi ích chính của việc lượng tử hóa mô hình bao gồm:

Giảm kích thước mô hình: Việc sử dụng định dạng số ít bit hơn giúp giảm đáng kể kích thước mô hình, cho phép lưu trữ và triển khai mô hình dễ dàng hơn trên thiết bị di động và nhúng.

Tăng tốc độ suy luận: Lượng tử hóa mô hình có thể giúp tăng tốc độ suy luận trên thiết bị di động và nhúng bằng cách giảm thiểu số lượng phép toán cần thiết để thực hiện các dự đoán.

Giảm mức tiêu thụ điện năng: Việc giảm thiểu số lượng phép toán và kích thước mô hình cũng có thể giúp giảm mức tiêu thụ điện năng khi triển khai mô hình trên thiết bị di động và nhúng.

Tensorflow Lite cung cấp hai phương pháp chính để lượng tử hóa mô hình:

Lượng tử hóa tĩnh (Static quantization): Phương pháp này thực hiện lượng tử hóa mô hình trong quá trình đào tạo, nghĩa là các giá trị trọng số và kích hoạt được lượng tử hóa trước khi triển khai mô hình.

Lượng tử hóa động (Dynamic quantization): Phương pháp này thực hiện lượng tử hóa mô hình trong quá trình suy luận, nghĩa là các giá trị trọng số và kích hoạt được lượng tử hóa dựa trên dữ liệu đầu vào cụ thể.

Chương 3: Ứng dụng cho nhiệm vụ cảnh báo các trường hợp nguy hiểm tại khúc cua.

3.1. Giới thiệu:

An toàn giao thông luôn là một trong những vấn đề được ưu tiên trong việc bảo vệ tính mạng và tài sản của con người. Đặc biệt, những khúc cua luôn là địa điểm nóng tiềm ẩn những nguy hiểm khó lường. Để giải quyết vấn đề này, em xin đưa ra giải pháp ứng dụng học sâu vào "Hệ thống cảnh báo các trường hợp nguy hiểm tại khúc cua" sử dụng công nghệ Object Detection tiên tiến.

Hệ thống được phát triển dựa trên nền tảng trí tuệ nhân tạo và học sâu (Deep Learning), với công nghệ Object Detection làm cốt lõi. Nhờ đó, hệ thống có khả năng nhận diện và phân tích nhanh chóng, chính xác các tình huống nguy hiểm như xe đi ngược chiều, người đi bộ, hoặc các vật cản bất ngờ tại khúc cua. Khi phát hiện nguy hiểm, hệ thống lập tức gửi cảnh báo đến người điều khiển phương tiện, giúp họ phản ứng kịp thời và an toàn.

Những lợi ích mà các công nghệ Object Detection được triển khai trên các thiết bị di động, các thiết bị nhúng mang lại:

- Nhận diện nguy hiểm nhanh chóng và chính xác: Nhờ vào các mô hình Object Detection, điện thoại di động có thể nhận diện các tình huống nguy hiểm như xe cộ đi ngược chiều, người đi bộ, hoặc các vật cản bất ngờ tại khúc cua một cách nhanh chóng và chính xác. Điều này giúp người điều khiển phương tiện có thể phản ứng kịp thời, giảm thiểu nguy cơ tai nạn.

- Tính di động và khả năng triển khai rộng rãi: Việc tích hợp các mô hình Object Detection vào điện thoại di động giúp tăng tính di động và khả năng triển khai rộng rãi. Chúng ta có thể triển khai hàng loạt mà không cần đầu tư quá nhiều về phần cứng và thiết bị.

- Cập nhật và cải tiến liên tục: Các mô hình Object Detection trên điện thoại di động thường xuyên được cập nhật và cải tiến thông qua các phiên bản phần mềm mới. Điều này đảm bảo rằng hệ thống luôn được cải thiện về độ chính xác và hiệu suất, đồng thời có thể nhận diện các tình huống nguy hiểm mới phát sinh.

- Hỗ trợ giám sát và phân tích giao thông: Ngoài việc cảnh báo người điều khiển phương tiện, các mô hình này còn có thể thu thập dữ liệu về giao thông tại các khúc cua nguy hiểm, hỗ trợ cho công tác giám sát và phân tích giao thông. Dữ liệu này có thể được sử dụng để đưa ra các giải pháp cải thiện an toàn giao thông trong tương lai.

- Tích hợp với các hệ thống giao thông thông minh: Các mô hình Object Detection trên điện thoại di động có thể được tích hợp với các hệ thống giao thông thông minh (ITS) để tạo ra một mạng lưới cảnh báo và quản lý giao thông hiệu quả hơn. Điều này giúp tối ưu hóa luồng giao thông và giảm thiểu tai nạn tại các điểm nguy hiểm

Qua nhiều năm phát triển, các mô hình được huấn riêng dành cho các thiết bị di động, các thiết bị nhúng như Tiny Yolo, EfficientDet đã mang lại những đóng góp quan trọng vào việc nâng cao an toàn giao thông, đặc biệt tại những khúc cua nguy hiểm. Đồng thời, việc triển khai các mô hình Object Detection trên các thiết bị di động, các thiết bị nhúng mở ra nhiều triển vọng mới cho việc ứng dụng công nghệ tiên tiến trong cuộc sống hàng ngày.

3.2. Ứng dụng mô hình pre-trained model Object Detection:

Trong vấn đề an toàn giao thông, các mô hình Object Detection được sử dụng để nhận diện và theo dõi các phương tiện, người đi bộ, biển báo giao thông, và các vật cản khác trên đường. Các mô hình pre-trained trong Object Detection đã cho thấy rằng chúng có thể được fine-tured hiệu quả đối với bài toán này.

Dưới đây là giới thiệu về cách áp dụng pre-trained models cho vấn đề an toàn giao thông:

Bước 1: Thử nghiệm và lựa chọn một pre-trained model phù hợp. Với tiêu chí là mô hình cần đạt được độ chính xác cao, tốc độ xử lý nhanh và triển khai được trên các thiết bị di động, các thiết bị nhúng, các mô hình như Tiny Yolo, EfficientDet, SSD300, SSD512, v.v đã được pre-trained trên dữ liệu lớn và có khả năng phân tích các đặc trưng và trích xuất được vật thể trong ảnh. Các model đã được training và cung cấp trên hunggingface, tensorflow-hub.

Bước 2: Fine-tuning là quá trình điều chỉnh pre-trained model để nó có thể phù hợp với bài toán cụ thể. Chúng ta cần một bộ dữ liệu huấn luyện chứa các hình ảnh về các vật thể đặc trưng của khu vực, sau đó gán nhãn cho từng vật thể. Quá trình fine-tuning giúp mô hình phân tích được các đặc trưng của vật thể có trong ảnh mà chúng ta cung cấp.

Bước 3: Chuẩn bị Dữ liệu là bước quan trọng để fine-tuning. Dữ liệu huấn luyện cần được chia thành tập huấn luyện và tập kiểm tra. Mỗi hình ảnh có thể có một hoặc nhiều vật thể, và mỗi vật thể đều cần được gắn nhãn để biết đó là vật thể nào.

Bước 4: Huấn luyện và Đánh giá Thực hiện quá trình fine-tuning trên tập huấn luyện và sử dụng tập kiểm tra để đánh giá hiệu suất của mô hình. Thông thường, đánh giá đo lường độ chính xác (accuracy) hoặc các độ đo đánh giá khác tùy thuộc vào yêu cầu cụ thể của bài toán.

Bước 5: Sử dụng mô hình đã fine-tuned để phân tích và trích xuất vật thể trong các ảnh mới.

3.3. Các công cụ sử dụng:

3.3.1. Ngôn ngữ lập trình Python:

Python chơi một vai trò quan trọng trong lĩnh vực Trí tuệ nhân tạo (AI). Dưới đây là một số điểm mạnh và ứng dụng của Python trong lĩnh vực này:

Ngôn ngữ Lập trình Phổ biến: Python là ngôn ngữ lập trình phổ biến và được ưa chuộng trong cộng đồng AI. Điều này là do cú pháp đơn giản, dễ đọc, và tính linh hoạt của Python, giúp các nhà phát triển tập trung vào việc phát triển mô hình AI mà không mất nhiều thời gian với các chi tiết lập trình phức tạp.

Thư viện và Frameworks: Python có nhiều thư viện và frameworks mạnh mẽ hỗ trợ phát triển dự án AI. Các thư viện như NumPy và pandas hỗ trợ xử lý và phân tích dữ liệu, trong khi TensorFlow và PyTorch cung cấp các công cụ mạnh mẽ cho việc xây dựng và huấn luyện mô hình machine learning và deep learning.

Community Support: Cộng đồng Python rất đông đảo và tích cực, với nhiều nguồn tài nguyên và diễn đàn hỗ trợ. Điều này làm cho việc học và giải quyết vấn đề trong lĩnh vực trí tuệ nhân tạo trở nên thuận tiện.

Ứng dụng Rộng Rãi: Python được sử dụng trong nhiều ứng dụng của trí tuệ nhân tạo, bao gồm xử lý ngôn ngữ tự nhiên (NLP), thị giác máy tính, dự đoán và phân loại, và nhiều lĩnh vực khác.

Nhờ những ưu điểm này, Python đã trở thành ngôn ngữ lập trình hàng đầu cho nhiều dự án và nghiên cứu trong lĩnh vực trí tuệ nhân tạo.

3.3.2. Visual Studio Code:

Visual Studio Code (VS Code) là một trình biên tập mã nguồn mở và miễn phí, được phát triển bởi Microsoft. Đây là một trong những công cụ phổ biến nhất dành cho lập trình viên nhờ vào tính năng mạnh mẽ và khả năng tùy biến cao.

Các tính năng chính thường sử dụng:

- Giao Diện Người Dùng:

  Giao diện người dùng trực quan và dễ sử dụng.

  Có thể tùy chỉnh giao diện bằng các chủ đề (themes) và bố cục (layouts).

- Hỗ Trợ Ngôn Ngữ Đa Dạng:

  Hỗ trợ nhiều ngôn ngữ lập trình như JavaScript, Python, Java, C++, C#, PHP, Go, v.v.

  Cung cấp IntelliSense cho tự động hoàn thành mã và gợi ý.

- Tiện Ích Mở Rộng:

Có một kho tiện ích mở rộng phong phú, giúp mở rộng tính năng của VS Code.

Các tiện ích mở rộng phổ biến như Prettier (định dạng mã), ESLint (kiểm tra lỗi mã), Live Server (chạy máy chủ cục bộ), và GitLens (quản lý Git).

- Tích Hợp Git:

Tích hợp sẵn công cụ Git giúp lập trình viên quản lý mã nguồn một cách hiệu quả.

Có thể thực hiện các thao tác như commit, push, pull trực tiếp từ VS Code.

- Debugging:

Hỗ trợ debugging trực tiếp trong VS Code cho nhiều ngôn ngữ lập trình.

Có thể đặt breakpoint, theo dõi biến và call stack.

- Terminal Tích Hợp:

Terminal tích hợp sẵn, cho phép chạy các lệnh trực tiếp trong VS Code mà không cần chuyển đổi cửa sổ.

3.4. Các thư viện và framework sử dụng:

3.4.1. Numpy:

NumPy là một thư viện Python cung cấp hỗ trợ cho các mảng và ma trận nhiều chiều. NumPy không chỉ giúp thực hiện các phép toán toán học và thống kê một cách hiệu quả trên dữ liệu số, mà còn là cơ sở cho nhiều thư viện khác, đặc biệt là trong lĩnh vực machine learning và scientific computing.

3.4.2. Pandas:

Pandas là một thư viện Python cung cấp các cấu trúc dữ liệu và công cụ phân tích dữ liệu dễ sử dụng, đặc biệt là DataFrame. DataFrame là một bảng hai chiều với các hàng và cột có thể được gán nhãn, giúp quản lý và xử lý dữ liệu dạng bảng một cách thuận tiện.

3.4.3. Json:

Thư viện JSON (JavaScript Object Notation) là một module tích hợp sẵn trong Python giúp bạn làm việc với dữ liệu JSON một cách dễ dàng. JSON là định dạng dữ liệu phổ biến được sử dụng để trao đổi dữ liệu giữa các ứng dụng và nền tảng khác nhau. Dữ liệu JSON thường được sử dụng để trao đổi thông tin giữa máy chủ và ứng dụng web dưới

dạng văn bản dễ đọc. Thư viện này cho phép bạn chuyển đổi (serialize) các đối tượng Python thành chuỗi JSON và ngược lại, chuyển đổi (deserialize) chuỗi JSON thành các đối tượng Python.

3.4.4. OpenCV:

Thư viện OpenCV (Open Source Computer Vision Library) là một thư viện mã nguồn mở mạnh mẽ và phổ biến để xử lý ảnh và thị giác máy tính. OpenCV cung cấp các công cụ và thuật toán tiên tiến để thực hiện nhiều thao tác xử lý ảnh cơ bản và nâng cao, phát hiện và nhận dạng đối tượng, xử lý video, và nhiều ứng dụng khác trong thị giác máy tính. Thư viện này rất phổ biến và được sử dụng rộng rãi trong nhiều lĩnh vực nghiên cứu và ứng dụng thực tế.

3.4.5. Shapely:

Shapely là một thư viện mã nguồn mở trong Python dành cho thao tác và phân tích các đối tượng hình học hai chiều, được xây dựng dựa trên thư viện GEOS (Geometry Engine Open Source) mạnh mẽ. Thư viện này cung cấp các công cụ để tạo, thao tác, và phân tích các hình học như điểm, đường thẳng, đa giác, và các tập hợp phức tạp hơn của các hình học này. Shapele giúp dễ dàng thực hiện các phép toán hình học phức tạp và kiểm tra các quan hệ không gian giữa các đối tượng hình học.

3.4.6. Tensorflow Lite:

TensorFlow Lite là một tập hợp các công cụ giúp bạn chạy mô hình TensorFlow trên các thiết bị di động, nhúng và biên giới. Nó cho phép bạn chuyển đổi mô hình TensorFlow đã được đào tạo sang định dạng nhẹ và hiệu quả hơn, phù hợp để triển khai trên các thiết bị có tài nguyên hạn chế.

Lợi ích sử dụng TensorFlow Lite:

Giảm kích thước mô hình: TensorFlow Lite có thể nén mô hình TensorFlow xuống còn một phần nhỏ kích thước ban đầu, giúp tiết kiệm dung lượng lưu trữ và băng thông.

Tăng tốc độ: TensorFlow Lite được tối ưu hóa cho hiệu suất cao trên các thiết bị di động và nhúng, giúp mô hình chạy nhanh hơn.

Hỗ trợ đa nền tảng: TensorFlow Lite có thể được sử dụng trên nhiều nền tảng di động và thiết bị nhúng khác nhau như Android, iOS, Raspberry Pi, v.v.

Dễ sử dụng: TensorFlow Lite cung cấp API Python dễ sử dụng để chuyển đổi và chạy mô hình.

3.5. Bộ dữ liệu sử dụng

COCO Dataset:

COCO (Common Objects in Context) là một tập dữ liệu quy mô lớn được sử dụng rộng rãi cho các bài toán về nhận diện và phân tích hình ảnh. Nó bao gồm hơn 330.000 hình ảnh, trong đó hơn 200.000 hình ảnh được chú thích với thông tin về các đối tượng trong ảnh.

Bộ dữ liệu COCO 2017 được sử dụng phổ biến cho nhiều tác vụ liên quan đến Computer Vision:

- Phát hiện đối tượng: Xác định vị trí và lớp của các đối tượng trong ảnh.
- Phân đoạn đối tượng: Vẽ đường viền xung quanh mỗi đối tượng trong ảnh.
- Chú thích hình ảnh: Tạo mô tả bằng văn bản cho nội dung của ảnh.
- Phân tích điểm chính: Xác định vị trí của các điểm chính trên cơ thể người trong ảnh.
- Phân đoạn ngữ cảnh: Phân chia ảnh thành các vùng ngữ cảnh khác nhau, chẳng hạn như bầu trời, mặt đất, cây cối, v.v.

Bộ dữ liệu COCO 2017 được đánh giá cao vì sự phong phú và đa dạng của nó. Nó bao gồm nhiều loại đối tượng khác nhau trong nhiều bối cảnh khác nhau, giúp nó trở thành một nguồn tài nguyên quý giá để đào tạo và đánh giá các mô hình thị giác máy tính.

Dưới đây là một số thông tin chi tiết về bộ dữ liệu COCO 2017:

- Số lượng hình ảnh: 330.000
- Số lượng hình ảnh được chú thích: 200.000
- Số lượng lớp đối tượng: 80
- Số lượng chú thích: 1.5 triệu
- Định dạng dữ liệu: JSON, COCO API

3.6. Triển khai mô hình

3.6.1. Cấu hình:

Bước 1: Cắt bỏ các khu vực không cần thiết:

Để tránh việc hệ thống mất thêm chi phí tính toán cho các vật thể không xuất hiện dưới lòng đường và xuất hiện ở khu vực không cần thiết (ví dụ: vỉa hè, ...), tôi sẽ cắt bỏ các khu vực không cần thiết đó ra hỏi khung hình của video hoặc của camera thu thập được.

Dưới đây là các bước thực hiện:

- Sử dụng thư viện shapely để vẽ được một hình polygon để bao chọn khu vực mà ta cần giữ lại.
- Khởi tạo các đỉnh của hình polygon
- Chuyền các đỉnh đó vào trong class Polygon của thư viện Shapely để tiến hình vẽ hình.
- Sử dụng hàm mask = np.zeros_like(frame): Sử dụng hàm np.zeros_like() của thư viện Numpy và chuyền khung hình (frame) vào trong hàm đó để có thể tạo ra một mặt nạ đen với kích thước bằng với kích thước của khung hình.
- cv2.fillPoly(mask, [self.points['area']],(255, 255, 255)):
    - cv2.fillPoly() được sử dụng để vẽ một đa giác lên mask. Đa giác này được xác định bởi các điểm trong self.points['area'].
    - self.points['area'] là một danh sách các điểm (tọa độ x, y) tạo thành đa giác.
    - (255, 255, 255) là màu trắng trong không gian màu RGB. Khi đa giác được vẽ lên mask, các vùng bên trong đa giác sẽ có giá trị màu trắng.
- result = cv2.bitwise_and(frame, mask):
    - cv2.bitwise_and(frame, mask) áp dụng phép toán bitwise AND giữa frame và mask.

Bước 2: Phân vùng khu vực dành cho hai làn đường
Để xác định vị của vật thể trên khung hình, tôi phần vùng khu vực mà tôi chọn thành hai phần khu vực tương ứng với hai làn đường trái và phải:

- points = self.points: là một từ điển chứa hai danh sách điểm, một cho phía 'left' (trái) và một cho phía 'right' (phải). Mỗi điểm được biểu diễn dưới dạng một tuple (x, y).
- for point in points['left']: frame = cv2.circle(frame, (point[0], point[1]), 3, (255, 0, 0), -1):
    - cv2.circle() được sử dụng để vẽ một vòng tròn tại tọa độ (x, y) chỉ định.
    - Bán kính của vòng tròn là 3 pixel.
    - Màu của vòng tròn là màu xanh dương (255, 0, 0) theo định dạng BGR.
    - -1 chỉ định rằng vòng tròn được tô đầy.
- for point in points['right']: frame = cv2.circle(frame, (point[0], point[1]), 3, (255, 0, 0), -1):
    - Tương tự như trên, nhưng màu của vòng tròn là màu xanh lá cây (0, 255, 0) theo định dạng BGR.

- frame = cv2.polylines(frame, [np.int32(points['left'])], False, (255, 0, 0), thickness=2); frame = cv2.polylines(frame, [np.int32(points['right'])], False, (0, 255, 0), thickness=2):
    - cv2.polylines() được sử dụng để vẽ các đường polyline nối các điểm trong mỗi danh sách.
    - [np.int32(points['left'])] và [np.int32(points['right'])] chuyển đổi các điểm thành mảng NumPy kiểu số nguyên, cần thiết cho OpenCV.
    - False chỉ định rằng đường polyline không được khép kín.
    - Màu của đường polyline cho 'left' là màu xanh dương (255, 0, 0) và cho 'right' là màu xanh lá cây (0, 255, 0).
    - Độ dày của đường polyline là 2 pixel.
- Phương thức trả về khung hình đã được chỉnh sửa với các điểm và các đường polyline đã được vẽ lên.

Bước 3: Vẽ hai điểm để có thể kiểm tra được hướng đi của vật thể và khu vực xuất hiện của vật thể

Vẽ hai điểm trên khung hình:

- frame = cv2.circle(frame, (point[0], point[1]), 5, (65,33,1), -1): hàm này sẽ vẽ lên trên khung hình một điểm có tọa độ là (point[0], point[1]), bán kính là 5 pixel, giá trị màu là (65,33,1) theo hệ màu RGB và có độ dày là -1 pixel.

Vẽ khung chứa vật thể và tâm của vật thể cho trình theo dõi cập nhật khung vật thể và mô hình trích xuất khung vật thể.

- Các tham số đầu vào:
    - self: Tham chiếu đến đối tượng của lớp mà phương thức này thuộc về.
    - frame: Là khung hình (ảnh) mà các hình chữ nhật sẽ được vẽ lên.
    - boxes: Một danh sách các hộp bao quanh (bounding boxes), mỗi hộp là một tuple (x, y, width, height) đại diện cho tọa độ và kích thước của hộp.
    - colors: Một danh sách các màu, mỗi màu là một tuple (B, G, R) đại diện cho màu sắc của mỗi hộp bao quanh.
- for i, newbox in enumerate(boxes):
  p1 = (int(newbox[0]), int(newbox[1]))
  p2 = (int(newbox[0] + newbox[2]), int(newbox[1] + newbox[3]))
  cv2.rectangle(frame, p1, p2, colors[i], 2, 1)
    - enumerate(boxes): Duyệt qua từng hộp bao quanh trong danh sách boxes cùng với chỉ số của nó.
    - newbox: Mỗi phần tử trong boxes, là một tuple (x, y, width, height).
    - p1: Tọa độ của góc trên bên trái của hình chữ nhật. Được xác định bằng (x, y) và chuyển đổi sang kiểu số nguyên.
    - p2: Tọa độ của góc dưới bên phải của hình chữ nhật. Được xác định bằng (x + width, y + height) và chuyển đổi sang kiểu số nguyên.

    - cv2.rectangle(frame, p1, p2, colors[i], 2, 1): Vẽ một hình chữ nhật lên frame từ p1 đến p2 với màu sắc colors[i], độ dày đường viền là 2 pixel, và kiểu đường viền là 1 (kiểu đường viền mặc định của OpenCV).
- Phương thức trả về khung hình đã được chỉnh sửa với các hình chữ nhật đã được vẽ lên.

Kiểm tra xem vật thể có trong khung hình không:

- Sử dụng hàm polygon.contains(centroid) thuộc lớp Polygon trong thư viện Shapely:

Kiểm tra hướng di chuyển của vật thể:

- Chuyển đổi điểm thành đối tượng trong lớp Point của thư viện Shapely:
    - Point(point_new): Chuyển đổi point_new thành một đối tượng Point. Có thể giả định Point là một lớp hoặc hàm từ một thư viện hình học (geometry) nào đó, chẳng hạn như shapely.geometry.Point.
    - Point(point_old): Chuyển đổi point_old thành một đối tượng Point.
    - Point(point_check): Chuyển đổi point_check thành một đối tượng Point.
- Tính toán khoảng cách và trả về kết quả:
    - point_check.distance(point_old): Tính toán khoảng cách từ point_check đến point_old bằng phương thức distance.
    - point_check.distance(point_new): Tính toán khoảng cách từ point_check đến point_new bằng phương thức distance.
    - Kết quả trả về là hiệu số giữa khoảng cách từ point_check đến point_old và khoảng cách từ point_check đến point_new.
- Mục đích của phương thức: Phương thức này xác định sự thay đổi khoảng cách của point_check từ point_old đến point_new. Từ đó có thể xác định được hướng di chuyển của vật thể:
    - Vật thể sẽ di chuyển đúng chiều nếu point_check gần point_new hơn point_old
    - Vật thể sẽ di chuyển ngược chiềum nếu point_check xa point_new hơn point_old.
    - Vật thể sẽ đứng yên nếu point_check cách đều cả hai điểm.

3.6.2. Chạy chương trình:

- Khởi tạo trình theo dõi vật thể - Object Tracking:
    - Sử dụng hàm cv2.legacy.MultiTracker_create() để tạo trình theo dõi nhiều vật thể.
    - Sau đó dùng hàm add() để thêm kiểu tracker vào trong khung hình và khung vật thể.
    - Mục đích là để theo dõi từng vật thể qua từng khung hình.
- Lấy thông tin về các điểm và hình đa giác trong file json
    - Các toạ độ điểm và toạ độ các đỉnh của đa giác sẽ được đọc và lưu vào đối tượng polygon_cal lớp polygon_calculate.

- Lấy các giá trị trọng số của mô hình và các nhãn vật thể:
    - Các giá trị trong số của mô hình sẽ được đọc từ tệp detect.tflite lưu đối tượng interpreter trong lớp Interpreter
    - "input_details" và "output_details" là các thông tin cần thiết về kích thước và định dạng của dữ liệu đầu vào và đầu ra của mô hình.
    - Các nhãn của vật thế sẽ được đọc từ tệp labelmap.txt và lưu vào mảng label
- Xử lý khung hình:
    - Đầu tiên, khung hình được chuyển từ màu BGR sang RGB và được resize về kích thước phù hợp với mô hình (width x height).
    - Hai thông số chiều rộng - width và chiều cao – height được lấy từ "input_details".
    - Dữ liệu đầu vào được chuẩn hóa nếu mô hình yêu cầu (biến floating_model).
- Đưa dữ liệu đầu vào vào trong mô hình:
    - Mô hình được gọi thông qua interpreter.invoke() để nhận dạng đối tượng và kết quả nhận được từ các tensor đầu ra (boxes, classes, scores).
    - Các hộp giới hạn (boxes_new), lớp (classes_new), và điểm tự tin (scores_new) của các đối tượng được lọc dựa trên ngưỡng tin cậy (min_conf_threshold) và được trả về.
- Theo dõi và xử lý kết quả:
    - Trong vòng lặp chính của chương trình, mỗi khung hình được đọc từ videostream.
    - Khung hình được xử lý để cắt bớt theo vùng đa giác được xác định bởi polygon_cal.cut_frame_polygon(frame).
    - Một MultiTracker được sử dụng để theo dõi các đối tượng trên các khung hình liên tiếp.
    - Mỗi num_frame_to_detect khung hình, thông tin chi tiết về các đối tượng được xử lý và lưu lại vào result_queue_cam.
    - Mỗi count == 0, hàm detect_ssd(frame) được gọi để nhận diện các đối tượng mới, và sau đó các hộp giới hạn và nhãn của các đối tượng được vẽ lên khung hình.
    - Các hộp giới hạn được truyền vào MultiTracker để tiếp tục theo dõi trên các khung hình tiếp theo.
- Lưu trữ và giải phóng tài nguyên:
    - Mở tệp JSON (RESULT_JSON_PATH) để ghi, sau đó lưu dữ liệu từ result_queue_cam vào tệp dưới định dạng JSON với indentation để dễ đọc.
    - Video kết quả sẽ được lưu vào tập output.mp4.
    - Giải phóng tài nguyên bằng cách:
        - videostream.release(): Giải phóng luồng video đang sử dụng.
        - out.release(): Giải phóng đối tượng VideoWriter, đối tượng này được sử dụng để ghi video.

- cv2.destroyAllWindows(): Đóng tất cả các cửa sổ hiển thị OpenCV đang mở.

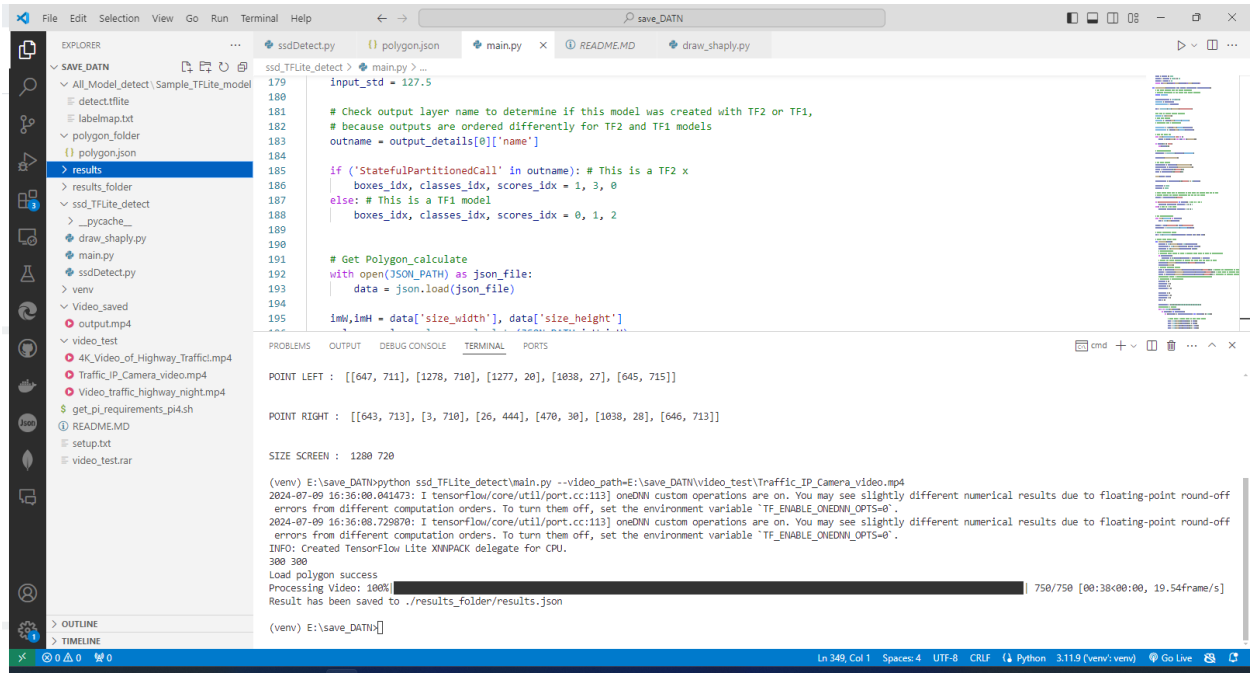
Chương 4: Kết quả và hướng phát triển

4.1. Cách thực hiện:

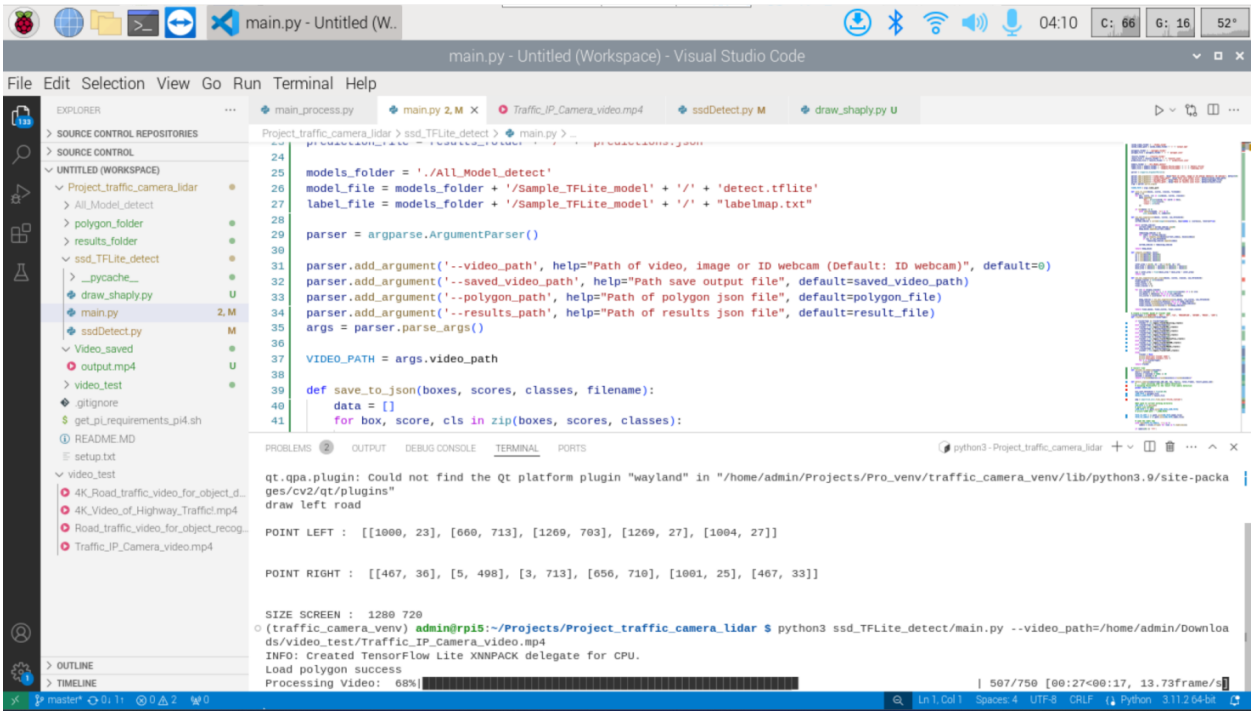
Những video xe cộ trên đường được em thu thập từ các nguồn trên mạng và tự quay. Sau đó các video trên sẽ được đưa vào chương trình xử lý để nhận diện và đưa ra kết quả. Chương trình được viết bằng ngôn ngữ Python trên IDE Visual Studio Code, sử dụng các thư viện như, OpenCV, Numpy, Shapely, JSON và framework Tenserflow.

Trước khi chạy đưa video vào trong chương trình để chạy, chúng ta cần cấu hình cho video, vẽ và đánh dấu các điểm và các hình polygon lên trên video và lưu toạ đồ vào file polygon.json.

Sau khi có được file polygon.json của video được cấu hình, chúng ta có thể đưa video vào trong chương trình.



Hình 12: Quá trình hoạt động trên hệ điều hành Windows 11 – 64 bits



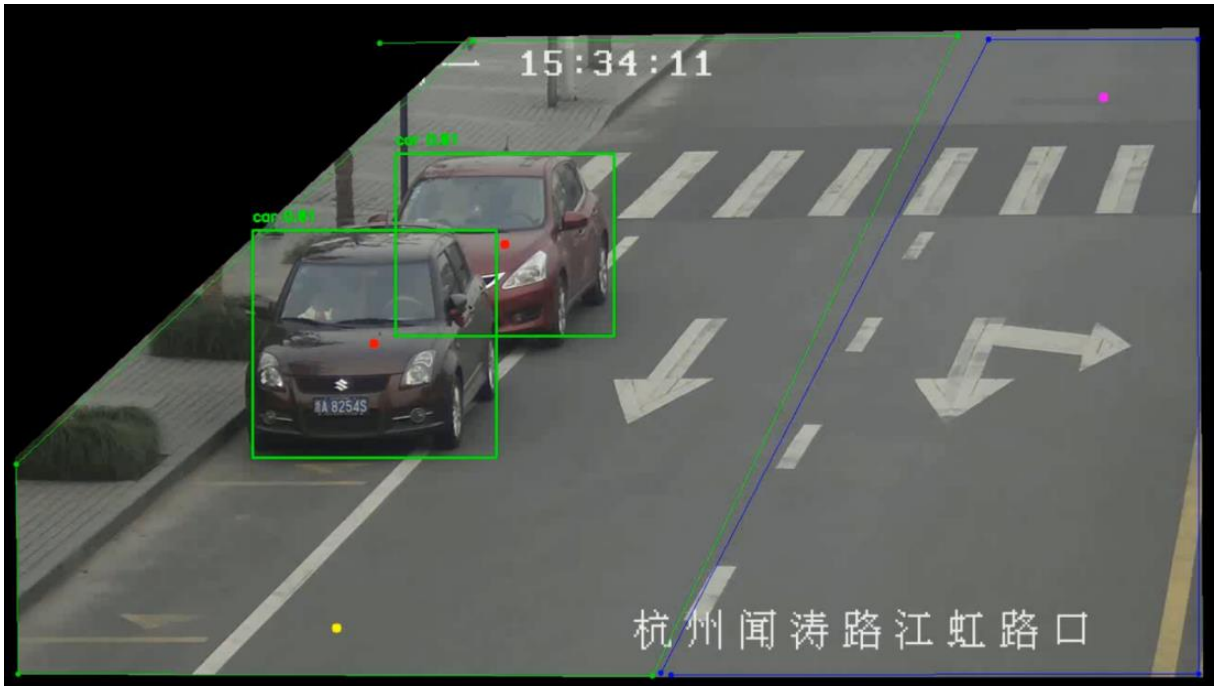
Hình 13: Quá trình hoạt động trên hệ điều hành Raspberry PI OS - 64 bit – Raspberry PI 5

4.2. Kết quả:

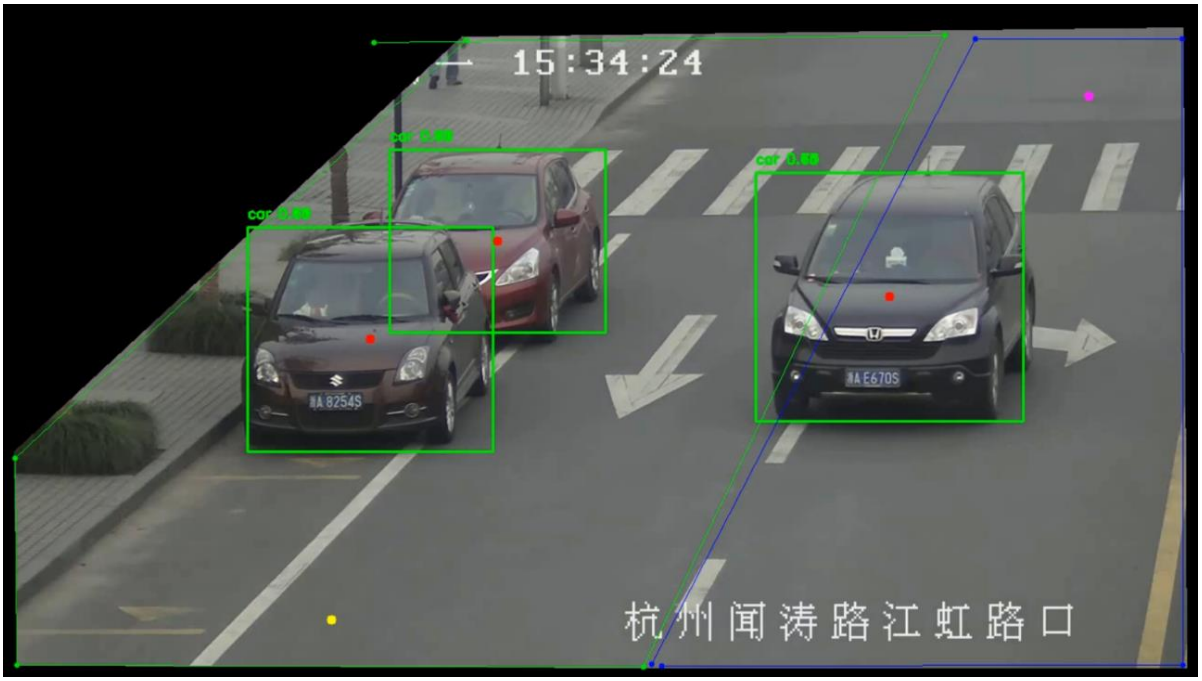
Hệ thống có thể nhận diện và theo dõi được các vật thể như ô tô,… với độ chính xác trong khoảng từ 50% đến 70% dựa trên điều kiện thời tiết và góc đặt camera.

Dưới đây là các kết quả đạt được:

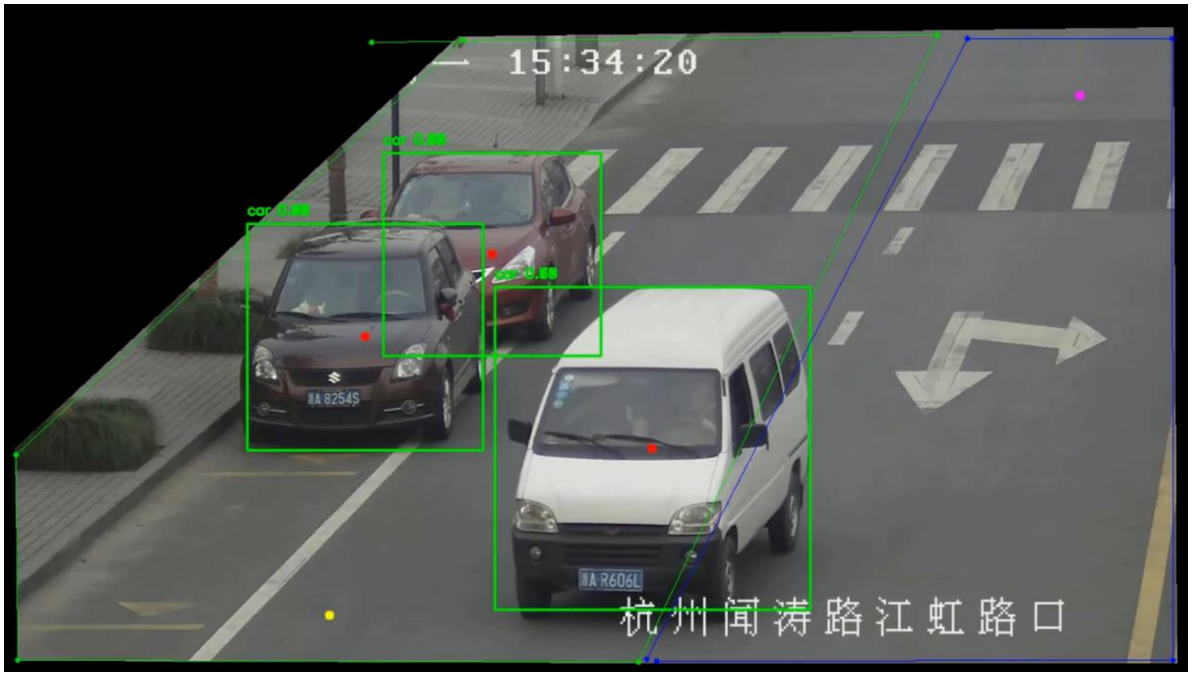
Trong điều kiện ánh sáng ban này và góc quay phù hợp, độ chính xác trong khoảng 68% đến 72%:



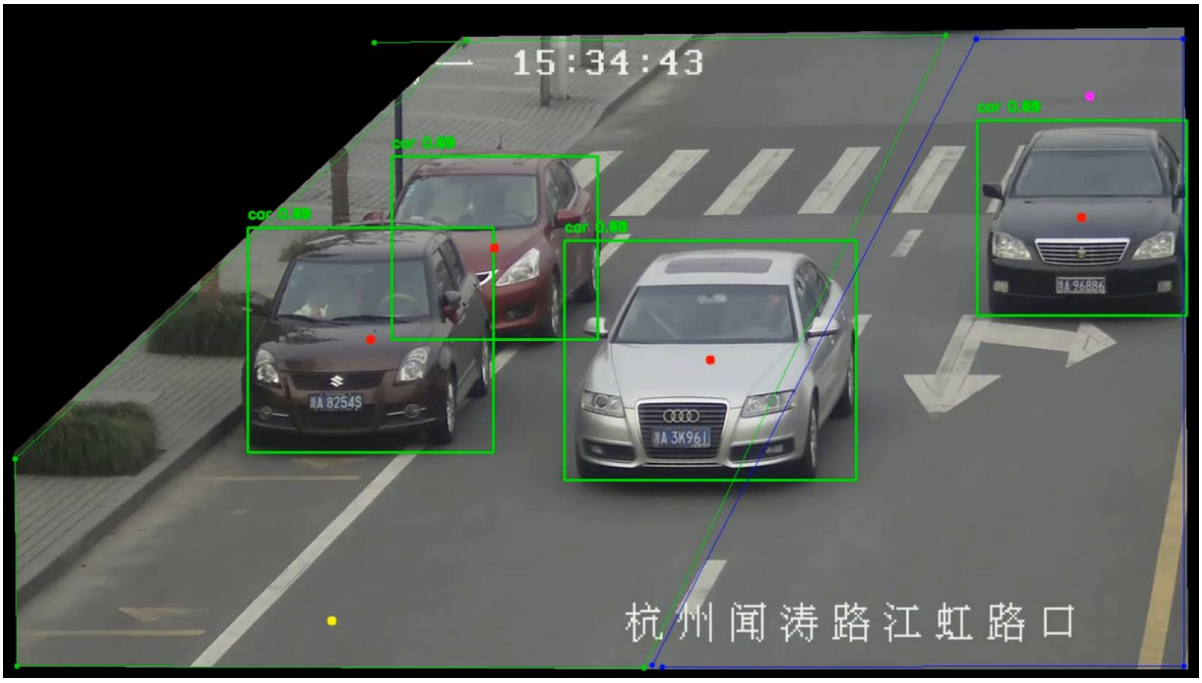
Hình 14: Hai xe đứng yên trên đường



Hình 15: Một xe đi ngược chiều

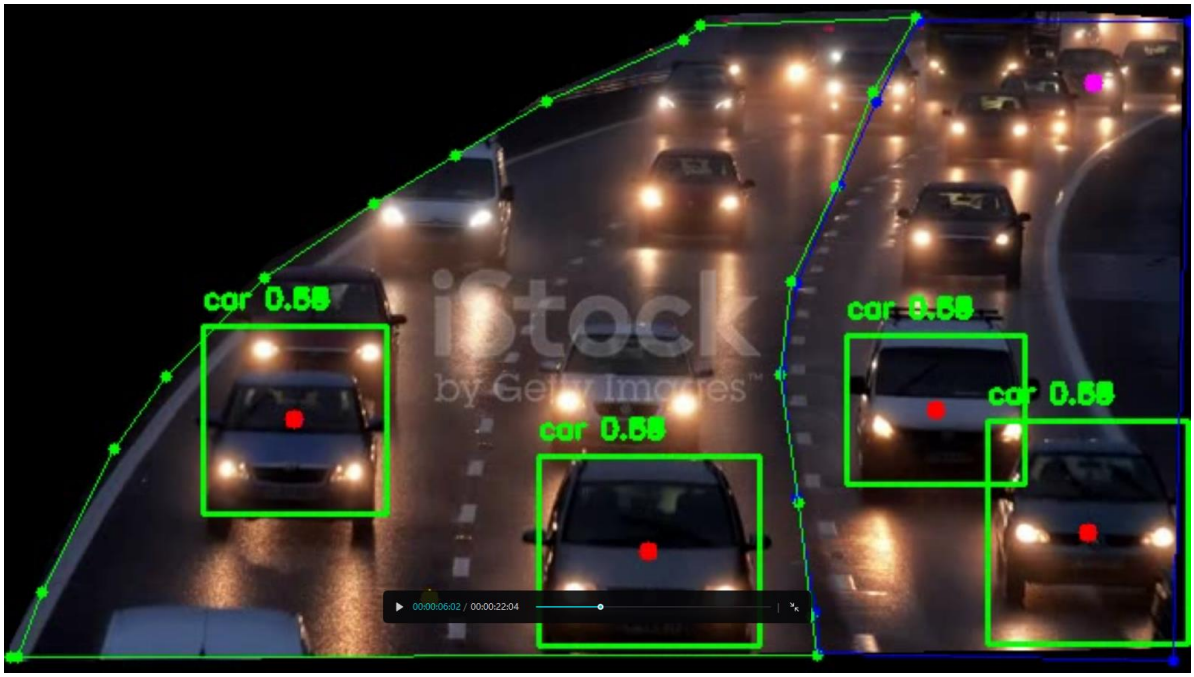


Hình 16: Một xe đi đúng chiều

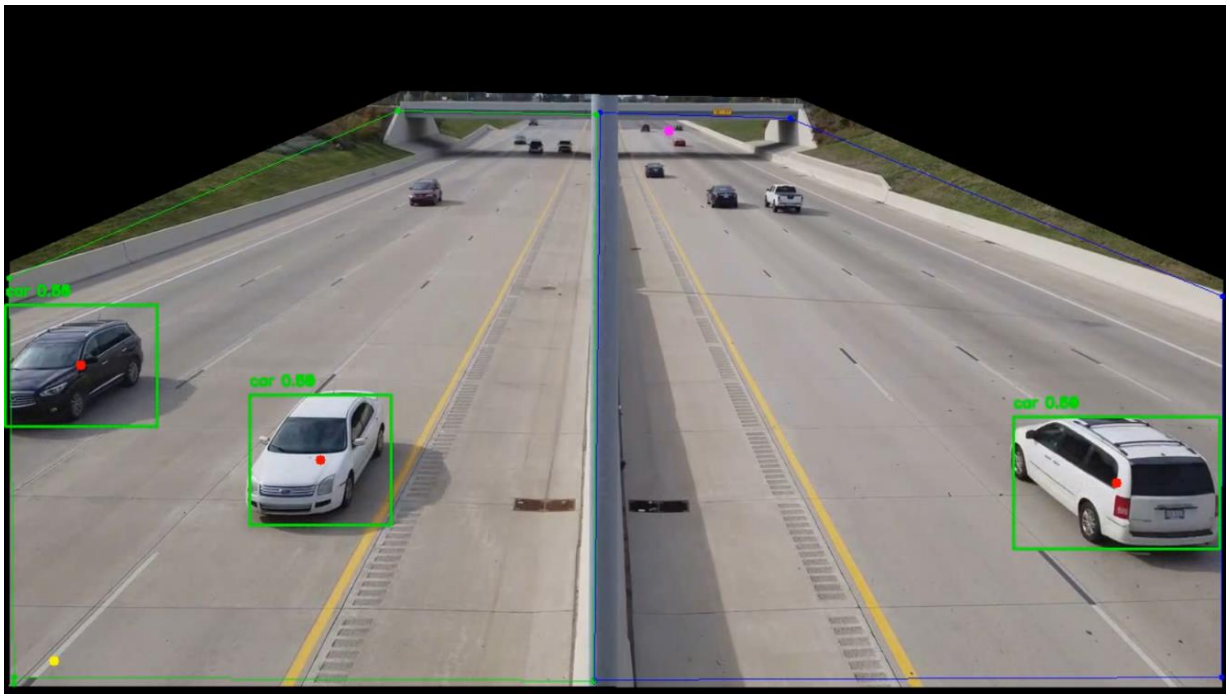


Hình 17: Một xe đi đúng, một xe đi ngược chiều và hai xe đứng yên

Trong điều kiện môi trường thiếu điều kiện, góc quay chưa được phù hợp, hệ thống chỉ nhận được những vật thể ở gần, độ chính xác bị giảm xuống giao động trong khoảng 50% đến 65%:



Hình 18: Trong điều kiện môi trường tối nhưng nguồn sáng mạnh



HÌnh 19: Góc quay chưa phù hợp, bị rung lắc

Hình 20: Góc quay quá xa và thiếu ánh sáng

Kết quả của việc nhận diện sai của hệ thống chủ yếu do điều kiện môi trường và góc đặt camera. Hệ thống cũng có thể bị ảnh hưởng từ các tác nhân bên ngoài như bị rung lắc, ảnh sáng phản chiếu dẫn đến khả năng phát sinh lỗi trong quá trình xử lý hình ảnh.

Hệ thống sử dụng mô hình SSD Mobile NET đã đặt được những kết quả khả quan, tuy nhiên vẫn tồn tại một số thách thức và hạn chế cần được xem xét. Một trong những vấn đề đó là độ nhiễu và độ mờ của hình ảnh. Những vấn đề đó đến từ việc camera và hệ thống cần phải được bảo trì và vệ sinh. Ngoài ra vẫn có nguy cơ nhầm lẫn giữa các vật thể, khoảng trống giữa hai vật thể,…

4.3. Định hướng phát triển:

Từ những kết quả thu được, tôi nhận thấy việc hệ thống cảnh báo các trường hợp nguy hiểm tại các khúc cua sử dụng mô hình SSD Mobinet V2 cho tốc độ xử lý thông tin và độ chính xác chưa có hiệu quả tốt. Trong tương lai, tôi có thể chuyển sang những mô hình khác ví dụ như YOLO-Tiny để có được sự hiệu quả tốt hơn.

Ngoài ra hệ thống có thể được bổ sung các chức năng nhỏ như nhận tín hiệu và gửi dữ liệu về cho máy chủ, gửi thông tin cảnh báo tới các tài xế thông qua sóng 5G, …

Kết Luận

Việc xây dựng hệ thống cảnh báo các trường hợp nguy hiểm tại khúc cua sử dụng mô hình học sâu trong thị giác máy tính là một bước tiến quan trọng trong việc đảm bảo an toàn giao thông. Ứng dụng công nghệ học sây vào an toàn giao thông không chỉ giúp nhận diện và cảnh báo các tình huống nguy hiểm mà còn tạo điều kiện cho việc quản lý giao thông trở nên hiệu quả hơn. Ở Việt Nam, nơi mật độ xe cộ đông đúc và các tình huống giao thông phức tạp, hệ thống cảnh báo này có thể đóng vai trò quan trọng trong việc giảm thiểu tai nạn và tăng cường an toàn cho người tham gia giao thông.

Sự kết hợp giữa công nghệ nhận diện hình ảnh và các mô hình học sâu sẽ mang lại những giải pháp tiên tiến, giúp nhận diện nhanh chóng các tình huống nguy hiểm tại khúc cua và cảnh báo kịp thời cho người lái xe. Việc này không chỉ tạo ra một môi trường giao thông an toàn hơn mà còn đóng góp đáng kể vào việc xây dựng hệ thống giao thông thông minh và tiện lợi. Qua đó, hệ thống này sẽ hỗ trợ hiệu quả cho lực lượng quản lý giao thông trong việc giám sát và điều phối các tình huống giao thông, góp phần quan trọng vào việc nâng cao chất lượng cuộc sống và đảm bảo an toàn cho cộng đồng.

Qua đề tài này em xin chân thành cảm ơn thầy ***ThS. Trương Xuân Nam*** đã dành thời gian và công sức hướng dẫn, hỗ trợ em trong suốt quá trình thực hiện đồ án tốt nghiệp. Trong suốt thời gian nghiên cứu, em đã cố gắng hết mình để tìm hiểu và triển khai nội dung đề tài. Tuy nhiên, đề tài này cũng khá phức tạp và thời gian nghiên cứu còn hạn chế, có thể sẽ còn thiếu sót và sai lệch.

Em rất mong được sự đóng góp ý kiến của các thầy cô giáo và các bạn để đồ án của em được hoàn thiện hơn.

Tài liệu tham khảo

[1] Các thuật toán trong Object Detection: https://phamdinhkhanh.github.io/2019/09/29/OverviewObjectDetection.html

[2] Computer Vision: https://en.wikipedia.org/wiki/Computer_vision#History

[3] Đánh giá các mô hình học máy: https://viblo.asia/p/danh-gia-cac-mo-hinh-hoc-may-RnB5pp4D5PG

[4] Hankel determinants of harmonic numbers and related topics: https://arxiv.org/pdf/1703.00763

[5] Luận văn Thạc sĩ Hệ thống thông tin: Tìm kiếm hình ảnh bằng phương pháp học sâu: https://tailieu.vn/doc/luan-van-thac-si-he-thong-thong-tin-tim-kiem-hinh-anh-bang-phuong-phap-hoc-sau-2576713.html

[6] Lượng tử hoá sau đào tạo: https://www.tensorflow.org/lite/performance/post_training_quantization?hl=vi

[7] Machine Learning Cơ Bản: https://machinelearningcoban.com/

[8] Mahalanobis distance: https://en.wikipedia.org/wiki/Mahalanobis_distance

[9] Mask R-CNN (Mask Region-based Convolutional Neural Network): https://github.com/topics/mask-rcnn

[10] MobileNET model: https://phamdinhkhanh.github.io/2020/09/19/MobileNet.html

[11] MobileNET V2 SSD FPN: https://edge-impulse.gitbook.io/docs/edge-impulse-studio/learning-blocks/object-detection/mobilenetv2-ssd-fpn

[12] Model SSD trong Object Detection: https://phamdinhkhanh.github.io/2019/10/05/SSDModelObjectDetection.html

[13] MobileNet: Convolutional Neural Networks for Mobile Image Recognition: https://arxiv.org/pdf/1704.04731

[14] Models Recomments: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/tf2_detection_zoo.md

[15] Multiple Object Tracking in Recent Times: A Literature Review: https://arxiv.org/pdf/2209.04796

[16] Object Detection with Deep Learning: https://arxiv.org/pdf/1807.05511

[17] Object Detection With Single Shot MultiBox Detector: https://github.com/enginBozkurt/Object_Detection_With_SSD

[18] Object Detection: https://medium.com/visionwizard/object-detection-4bf3edadf07f

[19] Object Tracking: https://viso.ai/deep-learning/object-tracking/

[20] Recent Advances in Deep Learning for Object Detection: https://arxiv.org/pdf/1908.03673
[21] Rectified Linear Units for Improved MNIST Classification: https://arxiv.org/pdf/2211.03232
[22] SSD (Single Shot MultiBox Detector): https://www.goozit.com/tutorial/ObjectDetection
[23] SSD: Single Shot MultiBox Detector: https://arxiv.org/abs/1512.02325
[24] Tensorflow API docx: https://www.tensorflow.org/api_docs/python/tf
[25] Tensorflow Lite Object Detection on Android and Raspbery PI: https://github.com/EdjeElectronics/TensorFlow-Lite-Object-Detection-on-Android-and-Raspberry-Pi
[26] Tensorflow Official build: https://github.com/tensorflow/tensorflow/tree/v2.11.0
[27] Train model SSD MobileNET với tensorlow 2 trên colab Mì AI: https://www.miai.vn/2022/01/03/train-model-ssd-mobilenet-voi-tensorflow-2-tren-colab-mi-ai/
[28] Ưu và nhược điểm của Deep Learning: https://www.vngcloud.vn/vi/blog/advantages-and-disadvantages-of-deep-learning
[29] Vehicle Detection and Tracking: https://github.com/kcg2015/Vehicle-Detection-and-Tracking
[30] What is Computer Vision?: https://medium.com/@ambika199820/what-is-computer-vision-history-applications-challenges-13f5759b48a5