

ĐẠI HỌC BÁCH KHOA HÀ NỘI

LUẬN VĂN THẠC SĨ

Ứng dụng giám sát thông minh trên thiết bị biên sử
dụng học sâu

LÊ ĐỨC LỘC

loc.ld211033M@sis.hust.edu.vn

Ngành: Công nghệ thông tin

Giảng viên hướng dẫn: TS. Đặng Tuấn Linh

Chữ ký GVHD

Khoa: Kỹ thuật máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 10/2023

LỜI CẢM ƠN

Tôi muốn gửi lời cảm ơn tới TS.Đặng Tuấn Linh với tất cả sự chân thành nhất vì những sự giúp đỡ, hướng dẫn mà thầy dành cho tôi trong suốt quá trình tham gia nghiên cứu. Từ những buổi họp nhóm cho đến những lần làm việc chung với thầy, tôi không chỉ học được những kiến thức về trí tuệ nhân tạo, thiết bị biến mà còn có cái nhìn rõ ràng hơn về định hướng nghiên cứu và học tập sau này. Ngoài ra, tôi cũng xin gửi lời cảm ơn đến bạn Hiếu, Tùng và Hưng. Không có các bạn thì tôi cũng khó có thể hoàn thành được luận án này. Được làm việc cùng với những bạn sinh viên đầy tài năng như các bạn là một may mắn đối với bản thân tôi. Và cuối cùng, tôi xin chân thành cảm ơn gia đình đã luôn ủng hộ tôi vô điều kiện trong suốt quá trình học cao học của mình.

TÓM TẮT NỘI DUNG LUẬN VĂN

Các phương pháp xử lý mới dựa trên trí tuệ nhân tạo và học sâu đang ngày càng trở nên phổ biến và dần thay thế các thuật toán thị giác máy tính truyền thống. Cùng với đó là sự phát triển của công nghệ về truyền dữ liệu tốc độ cao, ổn định, khiến cho nhiều nhà sản xuất và cung cấp dịch vụ lựa chọn hình thức triển khai thiết bị trên môi trường đám mây, đặc biệt là trong các ứng dụng liên quan đến theo dõi, xử lý hình ảnh. Dòng dữ liệu video từ hàng ngàn chiếc camera sẽ được gửi về các trung tâm dữ liệu của nhà sản xuất, tại đây, các máy tính được trang bị năng lực xử lý mạnh mẽ sẽ tiến hành phân tích, dự đoán để phục vụ cho người dùng. Tuy nhiên, phương pháp này gặp phải những thách thức liên quan đến độ trễ, khả năng mở rộng, và bảo mật. Ngược lại, trong môi trường xử lý tại biên, việc phân tích dữ liệu, đưa ra kết quả sẽ được thực hiện ngay tại nguồn dữ liệu mà không phải gửi về máy chủ qua mạng Internet. Lợi ích của việc xử lý tại biên đó là giảm độ trễ, giảm áp lực băng thông dữ liệu, và chia sẻ tác vụ tính toán với hệ thống máy chủ.

Hạn chế của mô hình này chính là khả năng thực hiện các phép tính toán bị giới hạn bởi tài nguyên của thiết bị, đặc biệt là khi áp dụng các thuật toán học sâu trong lĩnh vực thị giác máy tính. Với những thiết bị có năng lực xử lý mạnh mẽ thì đi kèm với đó là giá thành cao, không phù hợp để đầu tư triển khai thực tế. Ngược lại, những thiết bị có giá thành thấp với tài nguyên tính toán hạn chế lại không đảm bảo về tốc độ xử lý và độ chính xác của đầu ra. Điều này dẫn đến các mô hình tính toán tại biên vẫn chưa được triển khai rộng rãi trên thực tế mà mới chỉ dừng lại ở mức độ nghiên cứu.

Với mong muốn có thể đưa mô hình xử lý trên thiết bị biên có thể áp dụng vào thực tế một cách rộng rãi, nghiên cứu này đã đề xuất cách thức cài đặt và triển khai các mô hình học sâu chạy trên thiết bị biên có cấu hình phần cứng hạn chế, giá thành thấp mà vẫn đảm bảo yêu cầu về độ trễ, độ chính xác của đầu ra. Đề tài này sử dụng máy tính nhúng Jetson Nano để triển khai hai ứng dụng: phát hiện biển số xe và tái định danh người. Trong bài toán phát hiện biển số phương tiện, các mô hình học sâu tiên tiến đã được thử nghiệm, đánh giá trên Jetson Nano, và cho kết quả tương đương với chạy trên nền tảng đám mây Google Colab. Cụ thể, các mô hình phát hiện phương tiện được thử nghiệm cho độ chính xác lên đến 70% trong phép đo AP@0.5 cùng với tốc độ xử lý tối thiểu 25 FPS. Mô hình phát hiện biển số xe cho kết quả ấn tượng với tốc độ xử lý trên 200 FPS khi chạy trên thiết bị biên.

Trong bài toán tái định danh, đề tài đề xuất mô hình kết hợp giữa các thiết bị biên và máy chủ để xác định cùng một người trên nhiều camera. Kết quả cho thấy mô hình có thể đáp ứng yêu cầu thời gian thực với tốc độ 29 FPS. Ngoài ra, thuật toán phát hiện người trên thiết bị biên đạt kết quả 0,96 mAP.

Trong quá trình thực hiện luận văn, các kết quả của bài toán phát hiện biển số đã được trình bày tại một hội nghị quốc tế, kết quả của bài toán tái định danh đang được gửi phản biện tại một tạp chí quốc tế uy tín của ngành.

Học viên thực hiện

(Ký và ghi rõ họ tên)

ABSTRACT

New processing methods based on artificial intelligence and deep learning are becoming increasingly popular, gradually replacing traditional computer vision algorithms. Alongside this trend, the progress of high-speed, reliable data transmission technologies has prompted numerous manufacturers and service providers to embrace cloud computing environments for deploying applications, particularly those associated with monitoring and image processing. Video data streams from thousands of cameras are sent to the manufacturer's data centers, where powerful computers equipped with advanced processing capabilities will perform analysis and predictions. Despite this, there are still critical problems, including issues with scalability, security, and latency. In contrast, within edge computing environments, data analysis and result generation are carried out directly at the data source, eliminating the necessity of transmitting data over the Internet to servers. The benefits of edge processing are reduced latency, reduced data bandwidth pressure, and shared computational tasks with the server system.

The limitation of this model lies in its computing capability, which is constrained by the device's resources, especially when applying deep learning algorithms in computer vision. Devices with powerful processing capabilities come at a high cost, making them unsuitable for practical deployment. On the other hand, low-cost devices are restricted by CPU, GPU, RAM resources, leading to inadequate processing speed and output accuracy. As a result, edge computing models have mostly been used in research and have not yet been widely deployed in practice.

With the desire to bring edge device processing models into practical, widespread use, this research project proposes a method to set up and deploy deep learning models on a low-value edge device while still meeting requirements for latency and output accuracy. In this project, the Jetson Nano embedded computer is used to deploy two applications: license plate detection and person re-identification. For the license plate detection application, advanced deep learning models were tested and evaluated on Jetson Nano, yielding results equivalent to running on a cloud platform Google Colab. In details, the vehicle detection models were tested and achieved an accuracy of up to 70% in the AP@0.5 measurement along with a minimum processing speed of 25 FPS. Whereas, The license plate detection model gave impressive results with processing speeds of over 200 FPS when running on edge devices. In the re-identification application, the project utilized a combined

deployment approach between edge devices and server to identify the same person across multiple cameras. The results demonstrated that the model could meet real-time requirements with a frame rate of up to 29 FPS. The person detection algorithm on the edge device achieved a result of 0.96 mAP.

During the course of my thesis, the outcomes of the license plate detection application have been presented at an international conference, while the results of the re-identification application are currently undergoing review stage at a reputable international journal in the industrial field.

MỤC LỤC

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....	1
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi luận văn	3
1.3 Định hướng giải pháp.....	3
1.4 Bố cục luận văn thạc sĩ.....	4
CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT	6
2.1 Phát hiện đối tượng.....	6
2.1.1 Phương pháp truyền thống	6
2.1.2 Phương pháp sử dụng kỹ thuật học sâu.....	7
2.1.3 Mạng YOLO	10
2.2 Mạng WPOD	17
2.3 Theo vết đối tượng.....	19
2.4 Thiết bị AI tại biên	26
2.5 Thư viện hỗ trợ.....	28
2.5.1 TensorRT	28
2.5.2 Darknet	30
2.5.3 PyTorch.....	30
2.5.4 OpenCV	31
2.5.5 ZMQ.....	32
CHƯƠNG 3. BÀI TOÁN PHÁT HIỆN BIỂN SỐ XE TRÊN THIẾT BỊ BIÊN.....	34
3.1 Các nghiên cứu liên quan	34
3.2 Mô hình đề xuất	35
3.2.1 Tổng quan giải pháp	35

3.2.2 Mô-đun phát hiện phương tiện tham gia giao thông	37
3.2.3 Mô-đun phát hiện biển số xe	40
3.3 Thực nghiệm và kết quả	41
3.3.1 Thiết lập môi trường thực nghiệm.....	41
3.3.2 Bộ dữ liệu	42
3.3.3 Kết quả thực nghiệm	44
3.4 Kết luận	47
CHƯƠNG 4. BÀI TOÁN TÁI ĐỊNH DANH NGƯỜI	49
4.1 Nghiên cứu liên quan	49
4.2 Mô hình đề xuất	50
4.2.1 Tổng quan mô hình.....	50
4.2.2 Mô-đun phát hiện người.....	51
4.2.3 Mô-đun theo vết người	54
4.3 Thực nghiệm và kết quả	57
4.3.1 Thiết lập môi trường thực nghiệm.....	57
4.3.2 Bộ dữ liệu	58
4.3.3 Kết quả thực nghiệm	61
4.4 Kết luận	65
CHƯƠNG 5. KẾT LUẬN	66
5.1 Kết luận	66
5.1.1 Kết quả	66
5.1.2 Hạn chế.....	66
5.2 Hướng phát triển trong tương lai	67
TÀI LIỆU THAM KHẢO.....	74

DANH MỤC HÌNH VẼ

Hình 1.1	Thống kê sự quan tâm về lĩnh vực điện toán tại biên từ năm 2013 đến 2020 của Google Trend [1]	2
Hình 2.1	Lưu đồ thực hiện phát hiện đối tượng theo phương pháp truyền thống	7
Hình 2.2	Lưu đồ thực hiện phát hiện đối tượng áp dụng kỹ thuật học sâu	8
Hình 2.3	Mô hình kiến trúc mạng nơ-ron nhận dạng hai giai đoạn và một giai đoạn [21]	9
Hình 2.4	Mô hình kiến trúc mạng R-CNN [19]	9
Hình 2.5	Mô hình kiến trúc mạng Faster R-CNN [19]	10
Hình 2.6	Mô hình kiến trúc mạng YOLO [19]	10
Hình 2.7	Nguyên lý của mạng YOLO. Nó chia ảnh thành một lưới $S \times S$ và cho mỗi ô lưới, dự đoán B khung giới hạn, độ tin cậy cho những và C xác suất lớp cho mỗi ô [25]	11
Hình 2.8	Mô tả công thức tính IOU [30]	12
Hình 2.9	Kiến trúc mạng YOLOv4 [37]	14
Hình 2.10	Kiến trúc mạng YOLOv4-tiny [38]	15
Hình 2.11	Kiến trúc mạng YOLOv5 [39]	16
Hình 2.12	Kiến trúc mạng YOLOv7 [40]	17
Hình 2.13	Kết quả đánh giá và so sánh YOLOv7 [35]	17
Hình 2.14	Luồng xử lý bài toán nhận diện biển số xe	18
Hình 2.15	Vùng phát hiện biển số có nhiều chi tiết thừa do ảnh chụp không chính diện	18
Hình 2.16	Vùng biển số được phát hiện bởi mạng WPOD	19
Hình 2.17	Nguyên lý hoạt động mạng WPOD [42]	19
Hình 2.18	Kiến trúc mạng WPOD [42]	20
Hình 2.19	Luồng xử lý của thuật toán SORT	21
Hình 2.20	Máy tính nhúng Jetson Nano của NVIDIA	28
Hình 2.21	Năm loại tối ưu trên TensorRT [55]	29
Hình 2.22	Luồng chuyển đổi mô hình từ TensorFlow sang TensorRT [55]	30
Hình 3.1	Mô hình tổng quan của giải pháp phát hiện biển số xe	35
Hình 3.2	Mô-đun phát hiện phương tiện tham gia giao thông	36
Hình 3.3	Mô-đun phát hiện biển số xe	37

Hình 3.4	Luồng biến đổi từ YOLOv4-tiny sang TensorRT	38
Hình 3.5	lớp Conv2d được chuyển đổi thành một engine TensorRT, trong khi log_sigmoid sẽ sử dụng TorchScript JIT để thực thi [64] . .	39
Hình 3.6	Thực thi các toán tử PyTorch và TensorRT [64]	40
Hình 3.7	Kiến trúc của khối phát hiện trong mạng WPOD trước và sau khi được sửa đổi	41
Hình 3.8	Hình ảnh từ tập dữ liệu VVD	42
Hình 3.9	Hình ảnh từ tập dữ liệu VOID	43
Hình 3.10	Hình ảnh từ tập dữ liệu biển số xe	44
Hình 3.11	Kết quả phát hiện phương tiện trên tập OVD với các điều kiện ánh sáng khác nhau	45
Hình 3.12	Kết quả phát hiện phương tiện trên tập VVD với các địa điểm và thời gian khác nhau	46
Hình 3.13	Kết quả phát hiện biển số bằng WPOD	48
Hình 4.1	Mô hình đề xuất	50
Hình 4.2	Mô-đun phát hiện người trên thiết bị biên	51
Hình 4.3	Mô-đun truy vết người trên server	51
Hình 4.4	Quá trình phát hiện người	52
Hình 4.5	Quá trình loại bỏ bounding box bằng NMS	54
Hình 4.6	Tổng quan thuật toán truy vết	55
Hình 4.7	Thuật toán truy vết SORT	56
Hình 4.8	Vị trí lắp các thiết bị biên	58
Hình 4.9	Vị trí thực nghiệm số 1	58
Hình 4.10	Vị trí thực nghiệm số 2	59
Hình 4.11	Ứng dụng giám sát camera trên máy chủ	59
Hình 4.12	Một số hình ảnh từ bộ dữ liệu Penn-Fudan	60
Hình 4.13	Một số hình ảnh từ bộ dữ liệu CUHK03	61
Hình 4.14	Hình ảnh từ bộ dữ liệu BKREID	62
Hình 4.15	Kết quả thực nghiệm với những môi trường khác nhau	63
Hình 4.16	Kết quả thực nghiệm với các số lượng người khác nhau trong khung hình	63
Hình 4.17	Mô hình thực nghiệm của hệ thống tái định danh người	64
Hình 4.18	Kết quả thực nghiệm phát hiện và tái định danh người	64
Hình 4.19	Một số kết quả thực nghiệm phát hiện và tái định danh người .	64

DANH MỤC BẢNG BIỂU

Bảng 2.1 So sánh cấu hình phần cứng các máy tính nhúng thuộc dòng Jetson của NVIDIA [51]–[54]	27
Bảng 3.1 Bộ dữ liệu phương tiện giao thông	43
Bảng 3.2 Bộ dữ liệu biển số xe	44
Bảng 3.3 Kết quả của mô hình phát hiện phương tiện trên nền tảng Colab và Jetson Nano	46
Bảng 3.4 Kết quả của mô hình phát hiện biển số	46
Bảng 3.5 Kết quả phân loại biển số một dòng và hai dòng	47
Bảng 4.1 Kết quả của mô-đun phát hiện người trước và sau khi biến đổi sang TensorRT	62

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

Thuật ngữ	Ý nghĩa
AI	Trí tuệ nhân tạo
ALPR	Hệ thống nhận diện biển số xe (Automated License Plate Recognition)
AP@0.5	Average Precision với ngưỡng IOU 0.5
CNN	Mạng nơ-ron tích chập
CPU	Bộ xử lý trung tâm
CUDA	Kiến trúc thiết bị tính toán hợp nhất(phát triển bởi NVIDIA)
FPS	Số khung hình trên một giây (Frames per second)
GPU	Bộ xử lý hình ảnh
ID	Định danh
IoT	Internet vạn vật
IOU	Intersection over Union
ITS	Hệ thống giao thông thông minh
mAP	Mean Average Precision
mIoU	Mean Intersection over Union
NMS	Non-maximum Supresion
ONNX	Open Neural Network Exchange
RAM	Bộ nhớ truy cập ngẫu nhiên
Re-ID	Tái định danh
RNN	Mạng nơ-ron hồi quy
ROI	Region of Interest
RPN	Region Proposal Network
SDK	Công cụ phát triển phần mềm
SORT	Simple Online and Realtime tracking
SSD	Single Shot Detection
WPOD	Wraped Planer Object Detection Network
YOLO	You Only Look Once

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

Chương 1 của luận văn trình bày hiện trạng triển khai các ứng dụng trí tuệ nhân tạo theo hình thức điện toán đám mây và sự cần thiết của hai bài toán nhận diện biển số, tái định danh người. Từ đó, luận văn đưa ra đề xuất ứng dụng thiết bị biến để giải quyết hai bài toán phát hiện biển số và tái định danh. Cùng với đó, các định hướng giải pháp sẽ được đề xuất để có thể hài hòa giữa độ chính xác của mô hình và tốc độ xử lý trên thiết bị.

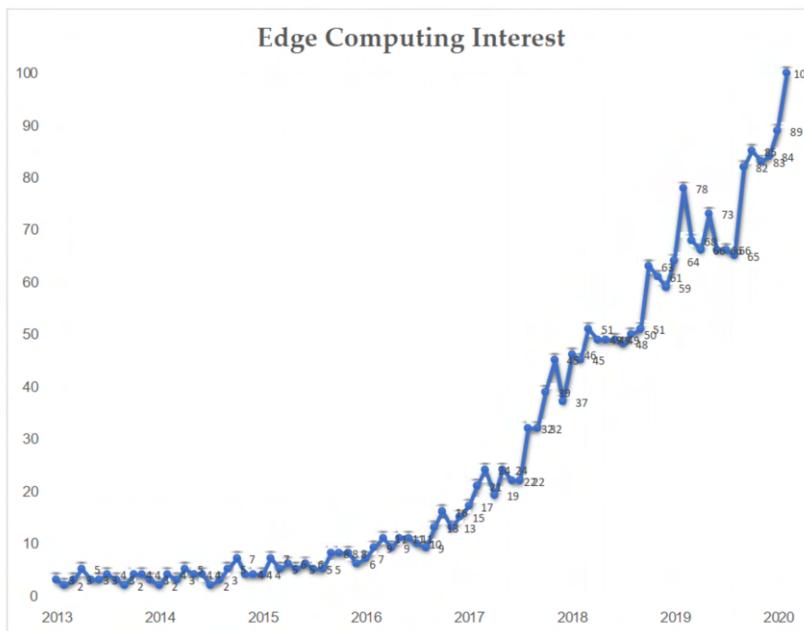
1.1 Đặt vấn đề

Những năm gần đây chứng kiến bước tiến mạnh mẽ của các công nghệ trí tuệ nhân tạo và IoT [1]. Trong đó, học sâu, một phương pháp xây dựng các thuật toán dựa trên mô phỏng hoạt động của các nơ-ron thần kinh trong não của con người, đang là lĩnh vực nhận được nhiều sự quan tâm trong trí tuệ nhân tạo. Phương pháp học sâu đã đạt kết quả vượt trội so với các phương pháp học máy truyền thống trong nhiều ứng dụng khác nhau, từ thị giác máy tính, xử lý ngôn ngữ tự nhiên cho đến phân tích dữ liệu lớn [2]. Từ đó đến nay, các nhà nghiên cứu đã tập trung xây dựng, cải tiến, và đạt được nhiều bước tiến rõ rệt với các mô hình học sâu khác nhau để phục vụ các bài toán về xử lý hình ảnh. Có thể kể đến như các ứng dụng về phát hiện, phân loại, và phân tách vật thể [3].

Tuy nhiên, sử dụng các mô hình học sâu cũng đi kèm các yêu cầu về khả năng tính toán và tài nguyên bộ nhớ của thiết bị phần cứng chạy các thuật toán này. Để có thể huấn luyện một mô hình học sâu phù hợp cần phải thực hiện rất nhiều phép tính toán, do phải tính toán và hiệu chỉnh hàng triệu tham số. Ngay cả sau khi đã hoàn thành việc rèn luyện mô hình, quá trình tính toán trong lúc suy luận thường phải đổi mới với việc tính hàng triệu phép toán với đầu vào là các dữ liệu đa chiều [3].

Để đáp ứng yêu cầu về tính toán của các mô hình học sâu, một xu thế chung đang được sử dụng rộng rãi hiện nay đó là áp dụng công nghệ điện toán đám mây. Tuy nhiên, khi mở rộng hệ thống với nhiều thiết bị thì hình thức triển khai mô hình học sâu trên đám mây gặp hạn chế về độ trễ xử lý và tiêu tốn nhiều tài nguyên băng thông Internet. Một hình thức triển khai khác đang thu hút sự chú ý của giới nghiên cứu trong thời gian gần đây là điện toán đám mây [1]. Trong đó, các thuật toán xử lý dữ liệu được thực hiện trên các thiết bị biên và sau đó gửi sẽ gửi về máy chủ tập trung những thông tin quan trọng, cần thiết nhất cho ứng dụng. Việc đưa thành phần xử lý dữ liệu về ngay tại biên sẽ giúp giảm độ trễ trong truyền tải dữ liệu, đồng thời giảm

tải áp lực về băng thông truyền dữ liệu và tính toán phức tạp trên máy chủ [3], [4]. Từ những ưu điểm trên, điện toán biên đem lại tiềm năng lớn trong các ứng dụng mà dữ liệu đầu vào có kích thước lớn, liên tục như ứng dụng giám sát qua video, hình ảnh. Đồ thị trong Hình 1.1 mô tả thống kê của Google Trend về sự quan tâm đến lĩnh vực điện toán biên [1].



Hình 1.1: Thống kê sự quan tâm về lĩnh vực điện toán tại biên từ năm 2013 đến 2020 của Google Trend [1]

Tại Việt Nam, mặc dù đã được trang bị hệ thống camera giao thông tại các tuyến đường, việc giám sát, xử lý vi phạm đang chủ yếu dựa vào lực lượng cảnh sát trực tại các chốt. Hình thức này không đảm bảo được việc xử lý liên tục 24/24 và bao quát được tất cả các tuyến đường cùng một lúc, khi có xảy ra sự cố hay tai nạn thì phải trích xuất lại dữ liệu camera để tìm phương tiện gây tai nạn. Điều này đặt ra nhu cầu cần có giải pháp phát hiện biển số tự động từ chính các camera trên đường mỗi khi có phương tiện vi phạm hoặc tai nạn [5]. Với số lượng thiết bị camera nhiều như hiện tại, hình thức triển khai điện toán tại biên sẽ là giải pháp phù hợp cho ứng dụng này.

Một bài toán khác cũng được áp dụng nhiều trong các ứng dụng quản lý, giám sát người tại các địa điểm công cộng, tòa nhà, siêu thị đó là tái định danh người [6] [7]. Mục đích chính của tái định danh người chính là xác định chính xác một người từ hình ảnh của nhiều camera, tại các thời điểm khác nhau, áp dụng ngay cả khi chưa được huấn luyện về hình ảnh của người đó. Trong lĩnh vực an ninh, hệ thống này sẽ giúp truy vết đường đi của những đối tượng khả nghi trong tòa nhà hoặc trên các tuyến đường khác nhau. Trong lĩnh vực bán lẻ, tái định danh sẽ giúp người bán

xác định được người mua hàng thường xuyên, hoặc các gian hàng mà người mua thường ghé qua trong siêu thị. Hiện tại, công việc này đang được thực hiện bởi người nhân viên giám sát, gây ra sự nhảm chán cho người lao động và tăng chi phí vận hành [8]. Vì vậy, cần có một hệ thống tự động dựa trên hình ảnh từ các máy ghi hình, trong đó, mỗi thiết bị biên này sẽ có khả năng xử lý hình ảnh tự động.

Từ những thực tế nêu trên, đề tài “Ứng dụng giám sát thông minh trên thiết bị biên sử dụng học sâu” được lựa chọn nghiên cứu với mong muốn đưa ra phương pháp triển khai các ứng dụng nhận diện biển số xe và tái định danh người dựa trên thiết bị biên có tài nguyên thấp. Qua đó, nghiên cứu này giúp giảm thiểu chi phí đầu tư khi triển khai với số lượng lớn.

1.2 Mục tiêu và phạm vi luận văn

Mục 1.1 đã đưa ra tiềm năng của thiết bị biên, cũng như hiện trạng và sự cần thiết của bài toán tự động nhận diện biển số và tái định danh người. Từ những ý đó, trong mục này sẽ trình bày những mục tiêu cần đạt được của luận văn.

Mục tiêu đầu tiên của luận văn đó là tạo ra bộ dữ liệu huấn luyện và triển khai thành công bài toán nhận diện biển số xe trên thiết bị biên với tốc độ xử lý ở mức thời gian thực, tương đương trên 24 FPS. Độ chính xác khi thực hiện dự đoán của mô hình cũng là một yếu tố để đánh giá việc áp dụng có thành công hay không, vì vậy, đề tài đặt ra mục tiêu độ chính xác trên 0,700 mIoU. Luận văn tập trung vào việc xử lý các mô hình học sâu trên thiết bị phần cứng, do đó, bài toán sẽ được tiến hành thực nghiệm trên thiết bị biên có cấu hình thấp, với dữ liệu đầu vào là ảnh được thu thập từ các bộ dữ liệu đề xuất.

Mục tiêu thứ hai của luận văn đó là giải quyết bài toán tái định danh dựa trên việc kết hợp giữa thiết bị biên và máy chủ tập trung, cùng với đó là xây dựng bộ dữ liệu cho ứng dụng tái định danh. Độ chính xác trên thiết bị biên cần đạt ít nhất 0,900 AP@0,5 và tốc độ xử lý tối thiểu 24 FPS. Để triển khai trên thực tế cần có phải đáp ứng yêu cầu về vị trí lắp đặt và đường truyền tải dữ liệu giữa các thiết bị biên, đường truyền Internet. Do đó, trong phạm vi của luận văn sẽ thực hiện xây dựng các mô hình nguyên mẫu và tiến hành thử nghiệm tại phòng nghiên cứu, trên thiết bị biên cấu hình thấp, dữ liệu đầu vào là hình ảnh thu từ webcam kết nối trực tiếp với phần cứng.

1.3 Định hướng giải pháp

Từ những mục tiêu được nêu trong mục 1.2, nội dung giải pháp thực hiện của luận văn bao gồm hai phần chính.

Trong phần đầu tiên, dựa trên các nghiên cứu về các bài toán nhận diện biển số xe cũng như năng lực xử lý của thiết bị biên, luận văn sử dụng các đề xuất mô hình xử lý bao gồm hai mô-đun là nhận diện phương tiện và mô-đun nhận diện biển số từ phương tiện. Trong đó, mô-đun nhận diện phương tiện sẽ phát hiện và phân loại các phương tiện thường xuyên xuất hiện trên đường bao gồm xe ô tô, xe buýt, xe tải, và xe gắn máy. Từ hình ảnh của phương tiện, mô-đun nhận diện biển số sẽ trích xuất hình ảnh biển số xe và hiệu chỉnh lại để góc nhìn biển số sao cho thẳng vuông góc với hướng nhìn từ mắt. Sau đó sử dụng phương pháp biến đổi mô hình để tăng tốc độ xử lý mạng học sâu phù hợp với thiết bị phần cứng. Để huấn luyện cho hai mô-đun, hai bộ dữ liệu liệu phương tiện giao thông và biển số xe được tạo ra bằng cách kết hợp nhiều bộ dữ liệu lại với nhau nhằm tạo ra sự đa dạng, cũng như gần với thực tế tại điều kiện giao thông của Việt Nam.

Trong phần thứ hai, mô hình triển khai theo hướng lai giữa thiết bị biên và máy chủ tập trung được đề xuất cho bài toán tái định danh người. Mô hình nhận dạng người sẽ được thực hiện trên thiết bị biên sau đó gửi thông tin lên máy chủ xử lý phần trích xuất đặc trưng và truy vết. Các thiết bị sẽ gửi các khung hình chứa người và khung bao vật thể theo giao thức tốn ít dung lượng đường truyền. Mô-đun nhận diện người được chuyển đổi sang dạng tối ưu để tăng tốc độ xử lý trên thiết bị biên. Mô hình nguyên mẫu được triển khai tại phòng thí nghiệm cho mục đích đánh giá. Bộ dữ liệu đề xuất với các đoạn video tự quay được đưa vào để tiến hành huấn luyện và đánh giá mô hình .

1.4 Bố cục luận văn thạc sĩ

Bố cục của luận văn sẽ bao gồm năm chương, với các thành phần được tổ chức như sau: Chương 1 trình bày thực trạng triển khai các ứng dụng trên thiết bị biên, lí do lựa chọn đề tài, các mục tiêu và giải pháp đề xuất của nghiên cứu.

Chương 2 sẽ trình bày cơ sở lý thuyết và khảo sát các nghiên cứu về ứng dụng được thực hiện trong luận văn như bài toán phát hiện đối tượng, bài toán theo vết đối tượng, các nền tảng thiết bị biên.

Chương 3 của luận văn đi vào chi tiết nội dung cách thức thiết kế, triển khai mô hình phân loại phương tiện, và phát hiện biển số xe trên thiết bị biên. Trong chương này, nghiên cứu sẽ đưa ra mô hình đề xuất, phương pháp và kết quả thực nghiệm trên phần cứng.

Chương 4 sẽ trình bày mô hình đề xuất cho ứng dụng tái định danh người trên thiết bị biên, các mô hình học sâu được thiết kế và áp dụng. Cũng trong chương

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

này, luận văn sẽ tổng hợp quá trình tiến hành thực nghiệm và đưa ra kết quả.

Chương 5 là kết luận, tóm tắt kết quả luận văn, những đóng góp, và hạn chế còn tồn đọng, hướng nghiên cứu để phát triển đề tài.

Như vậy, Chương 1 đã làm rõ lý do chọn đề tài, các mục tiêu và phạm vi luận văn mong muốn, định hướng sơ bộ về giải pháp xử lý, cũng như bối cảnh của quyển báo cáo. Từ đó, Chương 2 sẽ trình bày cơ sở lý thuyết để tiến hành triển khai xây dựng giải pháp cho bài toán phát hiện biến số và tái định danh trên thiết bị biên, đảm bảo mục tiêu đề ra của luận văn.

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

Chương 1 đã trình bày hiện trạng và động lực nghiên cứu ứng dụng bài toán tự động phát hiện biển số xe, tái định danh người. Cùng với đó, mục tiêu và định hướng giải pháp cho từng bài toán đã được giới thiệu. Trong phần này, báo cáo sẽ đề cập đến cơ sở lý thuyết của bài toán với các nội dung chính như sau: mục 2.1 và 2.2 trình bày các lý thuyết nền tảng và nghiên cứu mới nhất trong bài toán nhận dạng đối tượng và nhận diện biển số xe. Mục 2.3 trình bày về thuật toán theo vết đối tượng, tiếp theo là giới thiệu về nền tảng phần cứng AI tại biên của NVIDIA và cuối cùng là các thư viện hỗ trợ lập trình.

2.1 Phát hiện đối tượng

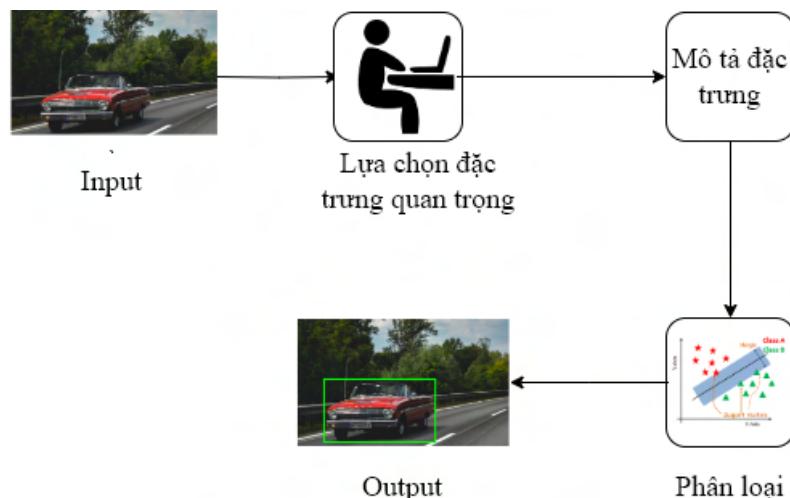
Phát hiện đối tượng là khả năng nhận biết, xác định vị trí và phân loại vật thể tồn tại trong một khung hình hoặc đoạn video. Đây là bài toán nền tảng cho các ứng dụng trong lĩnh vực thị giác máy tính, nó được sử dụng rộng rãi trong các ứng dụng phát hiện người, phát hiện khuôn mặt, phát hiện xe và các hệ thống đếm tự động [9]. Có rất nhiều các phương pháp khác nhau được nghiên cứu và áp dụng để phát hiện vật thể một cách chính xác với tốc độ ngày càng tiệm cận với khả năng của con người. Trong số đó, có thể chia ra làm hai trường phái chính đó là các phương pháp truyền thống và các phương pháp hiện đại ứng dụng kỹ thuật học sâu [10].

2.1.1 Phương pháp truyền thống

Trong phương pháp cổ điển, các đặc trưng quan trọng của đối tượng được thiết kế và trích xuất thủ công từ dữ liệu hình ảnh. Đặc trưng này được tạo ra dựa trên kiến thức và sự hiểu biết của con người về cấu trúc và đặc điểm của đối tượng cần phát hiện. Để biểu diễn vật thể, một vec-tơ đặc trưng được xây dựng thông qua thuật toán xử lý ảnh như HOG [11], SIFT [12], và SURF [13]. Bản chất của phương pháp HOG là sử dụng thông tin về sự phân bố của các cường độ gradient hoặc của hướng biên để mô tả các đối tượng cục bộ trong ảnh. Các toán tử HOG được cài đặt bằng cách chia nhỏ một bức ảnh thành các vùng con, được gọi là cell và với mỗi cell, ta sẽ tính toán một biểu đồ tần suất về các hướng của gradients cho các điểm nằm trong cell. Ghép các biểu đồ tần suất lại với nhau ta sẽ có một biểu diễn cho bức ảnh ban đầu. Để tăng cường hiệu năng nhận dạng, các biểu đồ tần suất cục bộ có thể được chuẩn hóa về độ tương phản bằng cách tính một ngưỡng cường độ trong một vùng lớn hơn cell, gọi là các khối và sử dụng giá trị ngưỡng đó để chuẩn hóa tất cả các cell trong khối. Kết quả sau bước chuẩn hóa sẽ là một

vec-tơ đặc trưng có tính bất biến đối với các thay đổi về điều kiện ánh sáng. Trong khi đó, thuật toán SIFT và SURF tập trung tạo ra các đặc trưng bất biến về tỷ lệ và độ xoay. Các vec-tơ đặc trưng của vật thể được trích xuất từ ảnh sẽ được đưa vào các bộ phân loại sử dụng thuật toán học máy như SVM [14], [15], AdaBoost [16].

Nhược điểm của phương pháp này là độ chính xác thấp khi hình ảnh bị ảnh hưởng bởi độ sáng, góc xoay, hướng nhìn, và không áp dụng được với sự đa dạng của các đối tượng trong ảnh [9], [17]. Để tăng độ chính xác cần phải chọn ra những đặc trưng thực sự quan trọng trong mỗi hình ảnh cụ thể, do đó, khi số lượng lớp cần phân loại tăng lên, việc trích xuất đặc trưng trở nên ngày càng phức tạp hơn. Việc quyết định những đặc trưng nào mô tả tốt nhất cho các lớp đối tượng khác nhau là dựa vào sự đánh giá của kỹ sư thị giác máy tính thông qua một quá trình thử nghiệm. Hơn nữa, mỗi định nghĩa đặc trưng yêu cầu xử lý một loạt các tham số, tất cả đều phải được điều chỉnh bởi kỹ sư thị giác máy tính [10] có nhiều kinh nghiệm như trong hình 2.1.

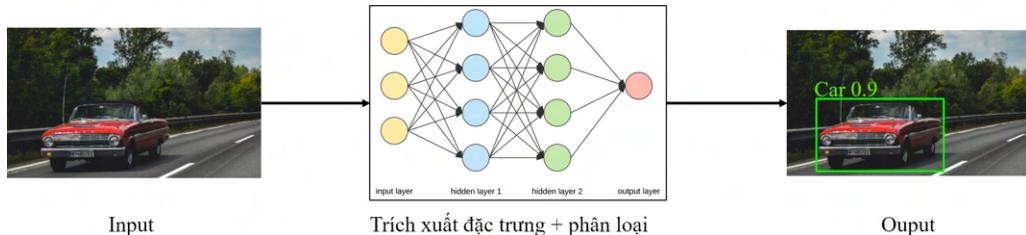


Hình 2.1: Lưu đồ thực hiện phát hiện đối tượng theo phương pháp truyền thống

2.1.2 Phương pháp sử dụng kỹ thuật học sâu

Sự phát triển của các kỹ thuật tính toán và tốc độ xử lý của máy tính đã dẫn đến sự ra đời của các kỹ thuật phát hiện vật thể dựa trên mạng nơ-ron nhân tạo vào những năm 2012 [18]. Với khả năng tự học những đặc trưng khái quát của từng loại vật thể, các mạng nơ-ron, cụ thể là kỹ thuật học sâu, đã mang lại sự cải thiện đáng kể về cả độ chính xác lẫn tốc độ xử lý [19]. Do đó, kỹ thuật này hiện đang là hướng được các nhà nghiên cứu và phát triển ứng dụng sử dụng chính trong các bài toán thị giác máy tính. Kỹ thuật học sâu mang đến khái niệm học toàn trình, trong đó máy tính chỉ cần một tập dữ liệu hình ảnh đã được phân loại sẵn các đối tượng hiện diện trong mỗi ảnh như Hình 2.2. Mô hình học sâu sau khi được huấn luyện

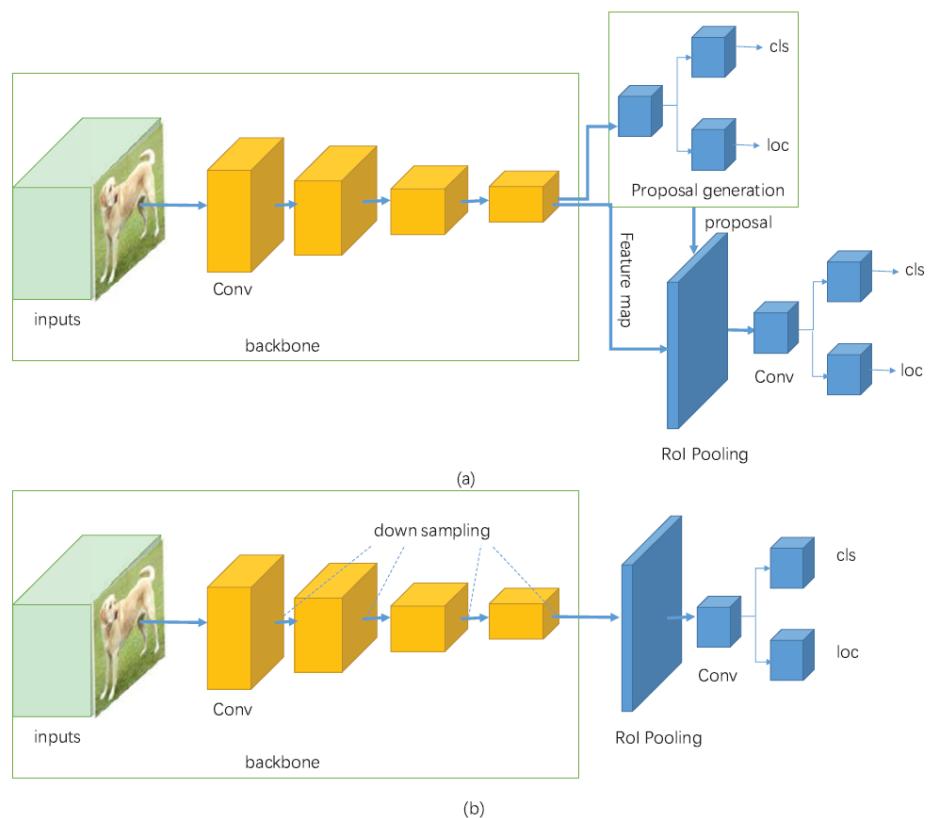
có khả năng tự động khám phá các đặc trưng nổi bật nhất của mỗi lớp đối tượng cụ thể. Các bộ dữ liệu trong học sâu phải đủ lớn và đa dạng để mô hình sau khi huấn luyện có khả năng khái quát hóa, cho phép nhận diện vật thể ở nhiều bối cảnh khác nhau [10].



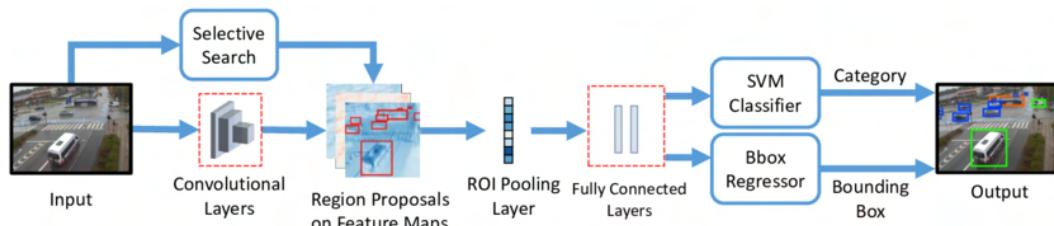
Hình 2.2: Lưu đồ thực hiện phát hiện đối tượng áp dụng kỹ thuật học sâu

Các mô hình học sâu liên quan đến phát hiện vật có thể chia ra làm hai dạng là bộ nhận dạng một giai đoạn và bộ nhận dạng hai giai đoạn [19] như trong Hình 2.3 . Các mô hình nhận dạng hai giai đoạn là các mô hình học sâu đầu tiên được phát triển cho bài toán phát hiện với họ các mạng nơ-ron tích chập theo vùng R-CNN [20]. Trong các mô hình R-CNN thường trích xuất các đặc trưng cần thiết nhất của đối tượng bằng cách sử dụng tìm kiếm chọn lọc. Quá trình lựa chọn các đặc trưng quan trọng nhất có thể được tính toán với sự trợ giúp của thuật toán tìm kiếm chọn lọc để tìm ra khoảng 2000 vùng có khả năng chứa đối tượng. Sau đó, các vùng này được chuẩn hóa về một kích thước cố định và đưa vào một mô hình mạng nơ-ron phân loại đã được huấn luyện để tiến hành xác định offset và gán nhãn đối tượng như trong Hình 2.4. Tuy nhiên, việc phải thực hiện tuần tự hai pha là lựa chọn vùng đề xuất rồi mới đến phân loại, gán nhãn khiến cho tốc độ thực thi của các mô hình này rất chậm [19].

Một số cải tiến đã được đề ra để giải quyết vấn đề tốc độ, trong đó nổi bật nhất là mô hình Faster R-CNN [22]. Với Faster-RCNN, thay vì việc sử dụng tìm kiếm chọn lọc, mô hình được thiết kế thêm một mạng con gọi là RPN để trích rút các vùng có khả năng chứa đối tượng của ảnh. Mô hình Faster R-CNN là một trong những phiên bản tốt nhất trong Fast R-CNN [23] và cải thiện tốc độ hoạt động rất nhiều so với các phiên bản tiền nhiệm. Trong khi mô hình R-CNN và Fast R-CNN sử dụng thuật toán tìm kiếm chọn lọc để tính toán các đề xuất vùng, thì phương pháp Faster R-CNN sẽ thay thế phương pháp hiện có này bằng một mạng RPN để xuất các vùng đặc trưng. Mạng RPN giúp giảm thời gian tính toán để trích chọn đặc trưng. Mạng này bao gồm các lớp tích hợp mà từ đó có thể thu được các đặc trưng cần thiết thông qua từng lớp tích chập liên tiếp nhau. Để đưa ra các vùng đặc trưng, RPN sử dụng các hộp neo (anchor box) với các tỉ lệ, kích thước và độ lớn

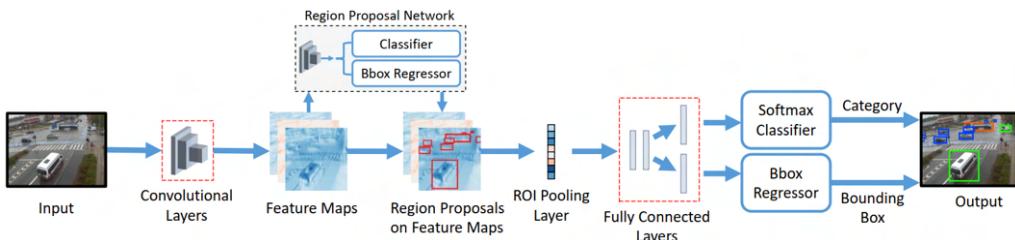


Hình 2.3: Mô hình kiến trúc mạng nơ-ron nhận dạng hai giai đoạn và một giai đoạn [21]



khác nhau. Đối với mỗi anchor box tại RPN, một bộ phân loại nhị phân được sử dụng để phân loại vùng trích chọn đó có khả năng chứa đối tượng hay không, và dự đoán ra các hộp giới hạn (bounding box) tương ứng. Sau đó, các vùng trích chọn sẽ được đưa qua một bộ lọc được cài đặt thuật toán NMS để loại bỏ các bounding box dư thừa. Đầu ra của NMS được cho qua một lớp gọi là ROI Pooling để cố định kích thước đầu ra của các vùng đặc trưng đã trích chọn như Hình 2.5.

Đối với bộ phát hiện một giai đoạn, sau khi nhận được ảnh đầu vào, đặc trưng điển hình của ảnh sẽ được trích xuất qua lớp tích chập, đồng thời tiến hành xác định vị trí và lớp đối tượng thuộc về. Do không có bước lựa chọn vùng có khả năng chứa đối tượng, kỹ thuật này cho tốc độ xử lý nhanh hơn so với các phương pháp hai giai đoạn mà vẫn đảm bảo độ chính xác cao, do vậy, đây đang là hướng đi chính cho các bài toán liên quan đến phát hiện, nhận dạng. Các đại diện của phương pháp này



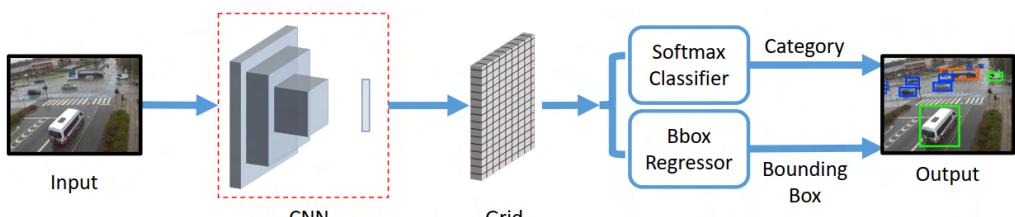
Hình 2.5: Mô hình kiến trúc mạng Faster R-CNN [19]

là SSD [24], YOLO [25], RetinaNet [26], RefineDet [27]. Trong số đó, mô hình YOLO đang được áp dụng trong nhiều ứng dụng thực tế và đã được phát triển liên tục qua nhiều phiên bản bởi mô hình này cho tốc độ xử lý thời gian thực và độ chính xác cao so với các bộ phát hiện một pha khác [28], [29].

Để thiết kế luồng xử lý cho các bài toán nhận diện biển số xe và tái định danh người, bước đầu tiên cần thực hiện đó là sử dụng các mô hình phát hiện đối tượng để tìm và phân loại phương tiện giao thông hay người ở trong khung hình. Dựa vào những ưu điểm và độ chính xác đã được nêu trên, đề tài sử dụng các bộ phát hiện đối tượng YOLO để trong mô hình xử lý của mình.

2.1.3 Mạng YOLO

YOLO là mô hình được nghiên cứu và sử dụng trong đề tài, vì vậy, mục này sẽ đề cập sâu hơn về kiến trúc và các đặc điểm của YOLO. So với các bộ phát hiện đối tượng khác, YOLO có ba ưu điểm chính đó (i) là tốc độ xử lý nhanh có thể đạt độ trễ nhỏ hơn 25 ms, ứng dụng được trong xử lý luồng video trực tiếp; (ii) khả năng trích xuất các thông tin của toàn bộ bức ảnh, do đó nó có thể hiểu được thông tin ngữ cảnh của các loại vật thể, giúp đưa dự đoán chính xác hơn; (iii) YOLO có khả năng học được các biểu diễn có tính tổng quát hóa của vật thể vì vậy có thể nhận diện vật thể ở nhiều góc chụp, độ sáng khác nhau [25]. Hình 2.6 thể hiện kiến trúc tổng quan của mạng YOLO.

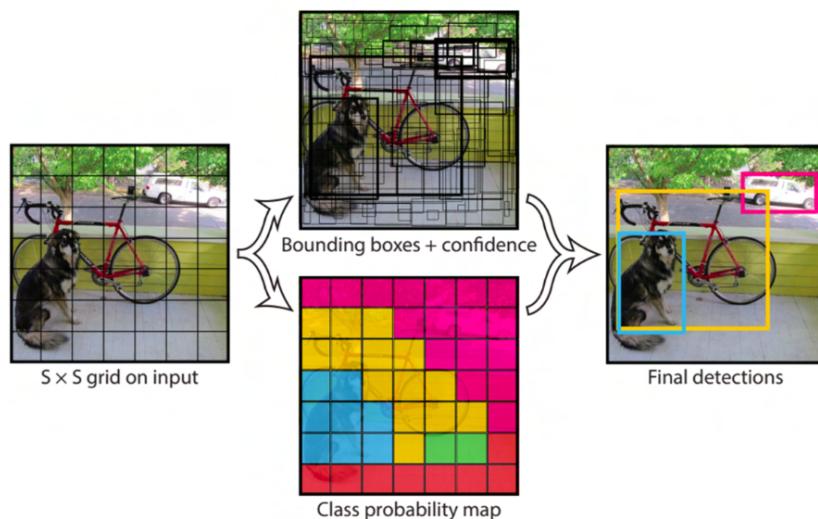


Hình 2.6: Mô hình kiến trúc mạng YOLO [19]

Mạng YOLO chia hình ảnh input ra thành một lưới kích thước $S \times S$ cell. Mỗi cell sau khi đi qua mạng nơ-ron sẽ cho ra kết quả dự đoán về khung giới hạn và xác

suất của vật thể có chứa trong ô đối với từng lớp. Do vậy, Với đầu vào là một ảnh, đầu ra mô hình sẽ là một ma trận ba chiều có kích thước $S \times S \times (5 \times B + C)$ với B và C lần lượt là số lượng ô giới hạn và số lớp mà mỗi ô lối cần dự đoán. Ví dụ, trong Hình 2.7, với hình ảnh được chia thành 7×7 ô lưới, mỗi ô cần dự đoán hai bounding box và xác suất với ba vật thể: con chó, ô tô, và xe đạp thì đầu ra sẽ là ma trận $7 \times 7 \times 13$. Trong đó,

- Bounding box là thể hiện tọa độ của vật thể trên ảnh và độ tin sự xuất hiện của vật thể trong khung, bao gồm năm thông số đó là $x, y, w, h, confidence$
- Lớp là tập đối tượng mà vật thể thuộc về. Mỗi cell sẽ dự đoán C giá trị chính là xác suất có điều kiện của các class trên một đối tượng $Pr(Class_i|Object)$. Ví dụ, cần phải phân loại vật thể thuộc một trong ba lớp thì đầu ra của cell sẽ là ba xác suất ứng với mỗi lớp.



Hình 2.7: Nguyên lý của mạng YOLO. Nó chia ảnh thành một lưới $S \times S$ và cho mỗi ô lưới, dự đoán B khung giới hạn, độ tin cậy cho những và C xác suất lớp cho mỗi ô [25]

Chỉ số tin cậy(Confidence) là một trong các tham số của bounding box phản ánh mức độ tin cậy của mô hình trong việc dự đoán ô đó chứa vật thể và độ chính xác của ô dự đoán là bao nhiêu. Chỉ số độ tin cậy được tính theo công thức sau:

$$Confidence = Pr(Object) \times IOU_{pred}^{truth} \quad (2.1)$$

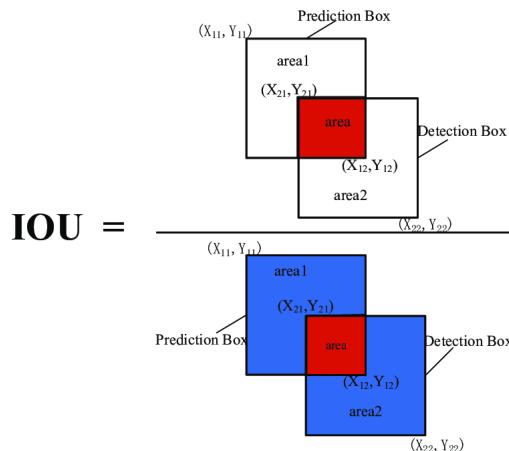
Trong đó,

- IOU_{pred}^{truth} là hàm đánh giá độ chính xác của object detector trên tập dữ liệu cụ

thể. IOU được tính bằng công thức sau:

$$IOU = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (2.2)$$

Với Area of Overlap là diện tích phần giao nhau giữa bounding box được dự báo với bounding box thực tế, còn Area of Union là diện tích phần hợp giữa bounding box được dự báo với bounding box thực tế. Những bounding box được gán nhãn bằng tay trong tập huấn luyện và tập kiểm tra. Nếu $IOU > 0.5$ thì prediction được đánh giá là tốt. Hình 2.8 mô tả công thức tính IOU .



Hình 2.8: Mô tả công thức tính IOU [30]

- $Pr(\text{Object})$ là xác suất ô đó có chứa vật thể. Nếu không tồn tại vật thể nào trong cell đó thì $Pr(\text{Object}) = 0 \implies \text{Confidence score} = 0$. Ngược lại, nếu cell đó chứa vật thể thì $Pr(\text{Object}) = 1 \implies \text{Confidence score} = IOU$

Để xác định được đối tượng trong cell thuộc về lớp nào cần phải biết thông tin của C xác suất ứng với mỗi lớp. Từ đó, tính được độ tự tin của dự đoán trong một ô theo công thức sau:

$$Pr(\text{Class}_i|\text{Object}) \times Pr(\text{Object}) \times IOU_{\text{pred}}^{\text{truth}} = Pr(\text{Class}_i) \times IOU_{\text{pred}}^{\text{truth}} \quad (2.3)$$

Kết quả cả công thức 2.3 cho biết xác suất của một Class_i xuất hiện trong ô và độ khớp của ô dự đoán với vật thể thực.

YOLOv1 ra đời đã khắc phục được nhược điểm về tốc độ của các mô hình trước đó, tuy nhiên vẫn còn các hạn chế sau:

- YOLOv1 đặt ra ràng buộc lớn về không gian lên việc dự đoán các bounding box: mỗi grid cell chỉ được dự đoán 2 boxes và một class duy nhất. Vì thế, ràng buộc này sẽ bộc lộ nhược điểm khi gặp grid cell có nhiều hơn một object, đặc

biệt là các object nhỏ chứa trọn trong một grid cell. YOLO sẽ gặp khó khăn trong việc dự đoán các object nhỏ và xuất hiện gần nhau hoặc theo nhóm.

- Vì YOLOv1 học cách dự đoán bounding boxes từ dữ liệu nên nó sẽ gặp khó khăn trong việc khái quát hóa đến các vật thể có aspect ratio mới hoặc bất thường.
- Regression Loss của YOLOv1 chưa tốt, dẫn đến việc đưa ra Bounding Box không tốt.

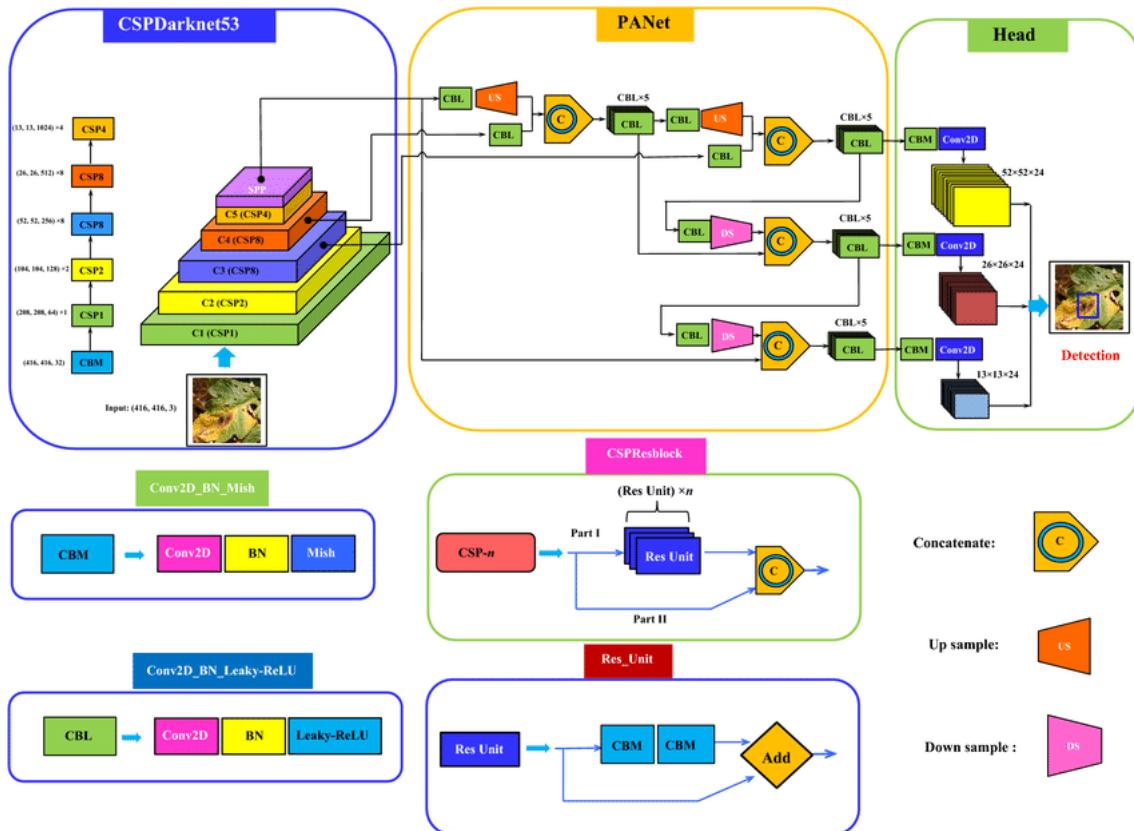
Để khắc phục các hạn chế trên đồng thời tăng độ chính xác và tốc độ xử lý, nhiều phiên bản khác nhau của YOLO đã được nghiên cứu và đề xuất bởi nhiều nhà khoa học với các phiên bản YOLOv2 - YOLOv8[31]–[36], có nhiều cải tiến về mặt kiến trúc mạng nơ-ron, các kỹ thuật xử lý toán học.

YOLOv4 [32] là phiên bản đầu tiên không được phát triển bởi Joseph Redmon - tác giả của mô hình YOLO, thay vào đó, nó được phát triển bởi các tác giả khác. Phiên bản này được bổ sung thêm kỹ thuật Bag of Freebies và Bag of Specials cộng với các điều chỉnh trong kiến trúc mạng để tăng tốc quá trình huấn luyện và cải thiện độ chính xác của mô hình. Đây cũng là phiên bản cuối cùng được thiết kế và huấn luyện trên Darknet framework. Một số cải tiến quan trọng về kiến trúc trên YOLOv4 như sau:

- Backbone: Thay thế khối Residual thông thường ở YOLOv3 bằng khối CSP.
- Neck: Sử dụng hai kiến trúc SPP và PAN để phát hiện vật thể ở những tỷ lệ khác nhau. Trong đó, PAN là một sự cải tiến của FPN trong YOLOv3, với việc tạo ra thêm một đường để đưa đặc trưng từ các lớp nông và lớp sâu ngoài một đường khác của FPN. Còn SPP chính là đưa bản đồ đặc trưng vào các lớp pooling với các kernel khác nhau, sau đó thực hiện ghép lại nhằm tăng cường trường tiếp nhận của mô hình.

Trong quá trình huấn luyện, YOLOv4 cũng áp dụng một số thay đổi nhằm cải thiện hiệu năng mô hình như sau:

- Tăng cường dữ liệu: Áp dụng Mosaic và MixUp.
- Hàm mất mát: Cũng gồm ba thành phần như YOLOV3. Tuy nhiên ở thành phần sai số định vị đối tượng, sai số bình phương được thay thế thành CIoU. Thành phần confidence loss cũng có thay đổi, khi YOLOV4 chọn nhiều anchor box trên mỗi ô làm positive box, cụ thể IoU với ground truth chỉ cần lớn hơn một ngưỡng là sẽ được xem là positive, dưới ngưỡng là negative. Cách làm



Hình 2.9: Kiến trúc mạng YOLOv4 [37]

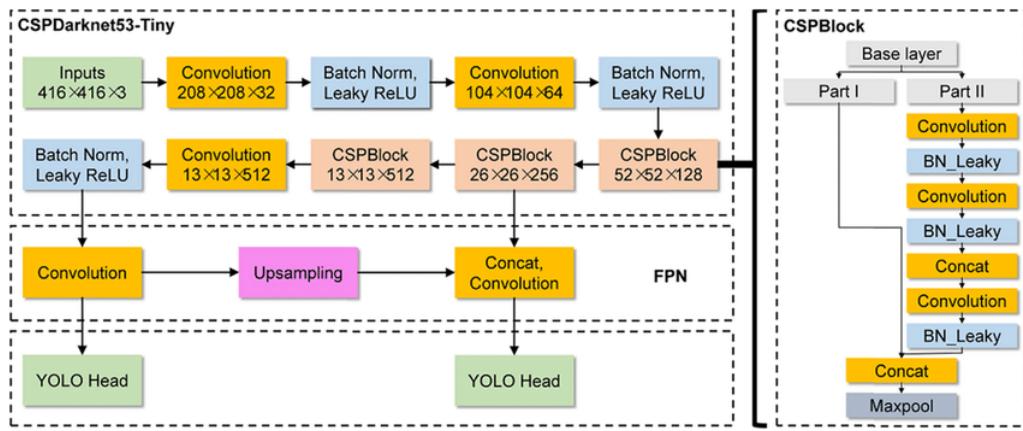
này nhằm hạn chế sự mất cân bằng giữa số lượng positive box và negative box xảy ra trong YOLOv3.

- Chính sửa công thức xác định tâm bounding box theo anchor box với mục đích giảm hiện tượng độ nhạy lưới:

$$b_x = \sigma(t_x) * 1.1 - 0.05 + c_x \quad (2.4)$$

$$b_y = \sigma(t_y) * 1.1 - 0.05 + c_y \quad (2.5)$$

Với những thay đổi trên, YOLOv4 đã đạt được sự cải tiến đáng kể về hiệu năng cũng như tốc độ so với các phiên bản tiền nhiệm. Trong bài toán nhận diện phương tiện, đề tài sử dụng mô hình phiên bản rút gọn của YOLOv4 đó là YOLOv4-tiny. Đây là một phiên bản thu gọn YOLOv4 với số lượng các lớp trong mạng học sâu được giảm đi đáng kể, từ 137 lớp tích chập được huấn luyện sẵn xuống còn 29 lớp. Do đó, YOLOv4-tiny cho tốc độ xử lý nhanh gấp tám lần so với bản đầy đủ trên tập dữ liệu kiểm tra COCO. Kiến trúc mạng YOLOv4-tiny được mô tả như trong Hình 2.10. Từ đó, nghiên cứu có thể đánh giá sự phù hợp của framework Darknet khi sử dụng để thiết kế cho các mạng cấu trúc nhẹ trên thiết bị biến.



Hình 2.10: Kiến trúc mạng YOLOv4-tiny [38]

Với YOLOv5, đây là mô hình đầu tiên được phát triển bởi một tổ chức, không phải một công bố khoa học. Phiên bản này cho kết quả chính xác vượt trội so với các phiên bản trước, cùng với kích thước mô hình nhỏ, tương thích với nhiều phần cứng hơn [33]. YOLOv5 kế thừa phần lớn những ý tưởng được đưa ra trong YOLOv4, cùng với đó áp dụng thêm một số thay đổi nhỏ để cải thiện hiệu năng mô hình.

- Neck: thay đổi khối SPP bằng SPP-Fast (SPPF). Theo đó, thay vì dùng ba MaxPooling song song với kích thước kernel 5, 9, 13 như SPP, khối SPPF sử dụng ba MaxPooling tuần tự nhưng có cùng kích thước kernel là 5. Về bản chất, hai mô-đun có cùng kích thước trường tiếp nhận, tuy nhiên SPPF có tốc độ xử lý cao hơn SPP do kích thước kernel bé hơn. Hình 2.11 mô tả kiến trúc mạng YOLOv5.
- Công thức xác định bouding box được cập nhật như sau:

$$b_x = 2 * \sigma(t_x) - 0.5 + c_x \quad (2.6)$$

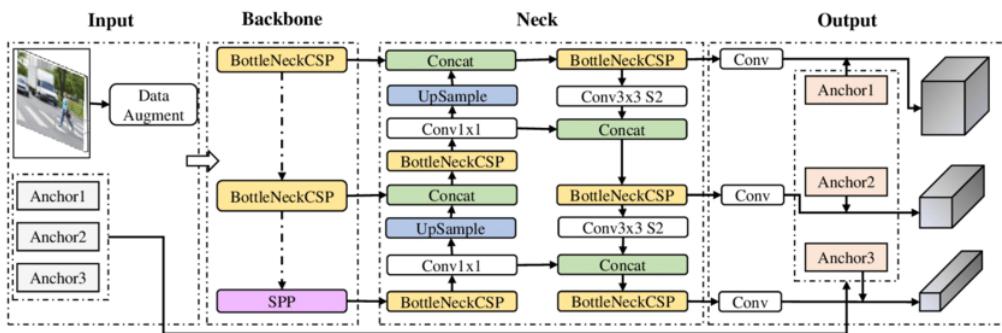
$$b_y = 2 * \sigma(t_y) - 0.5 + c_y \quad (2.7)$$

$$b_w = p_w * (2\sigma(t_w))^2 \quad (2.8)$$

$$b_h = p_h * (2\sigma(t_h))^2 \quad (2.9)$$

- Áp dụng công thức trung bình động theo hàm mũ (EMA) để cập nhật trọng số của mô hình trong quá trình huấn luyện.

Một phiên bản thu gọn của YOLOv5 là YOLOv5n, đây là phiên bản có số lượng tham số ít nhất trong tất cả các mạng YOLO tính đến thời điểm thực hiện thiết kế. Do đó, nó phù hợp với thiết bị biên và được lựa chọn để sử dụng để phát hiện người



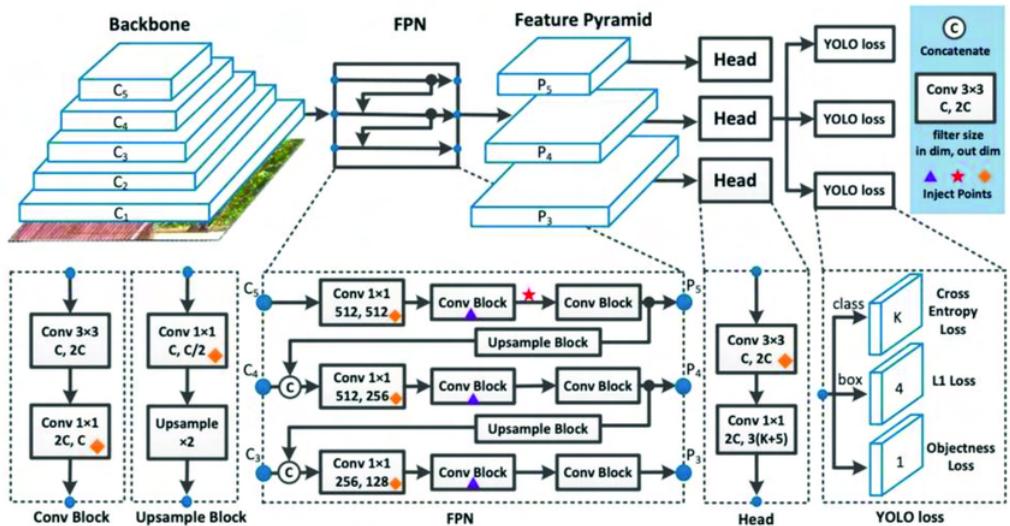
Hình 2.11: Kiến trúc mạng YOLOv5 [39]

trong bài toán tái định danh

YOLOv7 [35] là phiên bản mới nhất của YOLO tại thời điểm thực hiện đề tài, do đó nghiên cứu đã lựa chọn thực hiện thử nghiệm trên cả mạng YOLOv7 nhằm tìm ra phiên bản phù hợp nhất cho phần cứng đã chọn. YOLOv7 vượt qua mọi mô hình phát hiện đối tượng về phương diện tốc độ và độ chính xác. Nó cho tốc độ xử lý tối đa lên đến 5 ms và đạt độ chính xác cao nhất với 56,800% AP trong số toàn bộ các mô hình phát hiện đối tượng theo thời gian thực khi chạy thử trên GPU V100 (Hình 2.13) [35]. Hơn nữa, YOLOv7 được huấn luyện trên tập dữ liệu COCO từ đầu mà không sử dụng bất kỳ mô hình đã được huấn luyện sẵn nào. Kiến trúc mạng của YOLOv7 được mô tả như trong Hình 2.12. Một số các cải tiến trên mô hình YOLOv7 như sau:

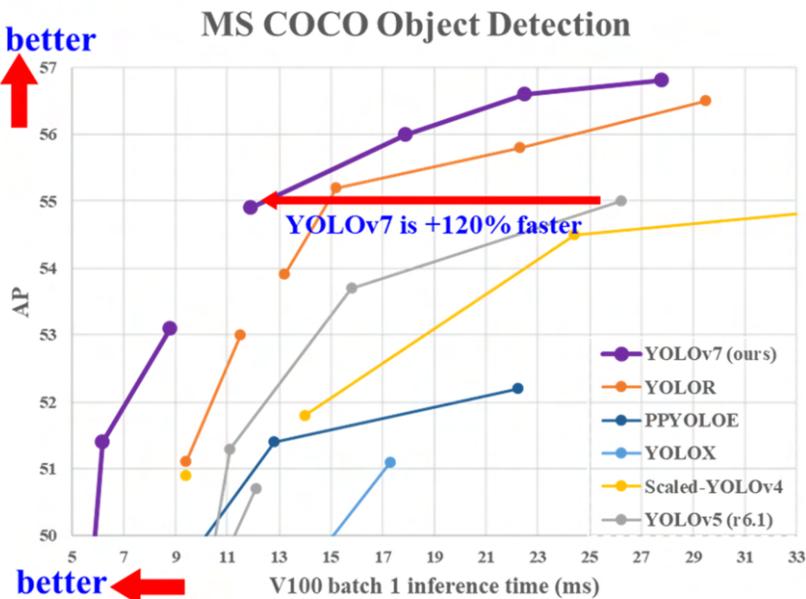
- Backbone: sử dụng Efficient Layer Aggregation Network (ELAN). Một khối ELAN gồm ba phần: Cross Stage Partial, khối Computation và phép PointWiseConv. Khối Computation chứa các lớp tích chập 3×3 được tính toán để sinh ra các đặc trưng mới. Cross Stage Partial là một lớp tích chập 1×1 được xử lý song song với khối Computation. Cuối cùng, các bản đồ đặc trưng được tổng hợp lại bằng toán tử concatenate và được đưa qua phép PointWiseConv. Các khối ELAN được kết nối với nhau thông qua các khối Transition. Mỗi lần đi qua khối Transition là một lần giảm kích cỡ của bản đồ đặc trưng đi hai lần.
- Neck: SPP được thay thế bằng SPPSPC, sử dụng phiên bản chỉnh sửa của PAN thay vì sử dụng PAN thông thường ở các phiên bản trước. Ngoài ra, YOLOv7 sử dụng RepConv 3×3 thay cho các lớp tích chập 3×3 để xử lý các bản đồ đặc trưng từ các tỷ lệ khác nhau, việc này cho tốc độ tương đương sử dụng tích chập 3×3 nhưng với độ chính xác cao hơn.
- YOLOv7 sử dụng chín anchor box, cho phép YOLO phát hiện phạm vi hình dạng và kích thước đối tượng rộng hơn so với các phiên bản trước, do đó giúp

giảm số lượng xác định sai.



Hình 2.12: Kiến trúc mạng YOLOv7 [40]

Để phát hiện phương tiện trên thiết bị biên, nghiên cứu lựa chọn mô hình YOLOv7-tiny, một phiên bản rút gọn của YOLOv7, để tiến hành thực nghiệm bởi những ưu điểm vượt trội của mạng như đã nêu trên.



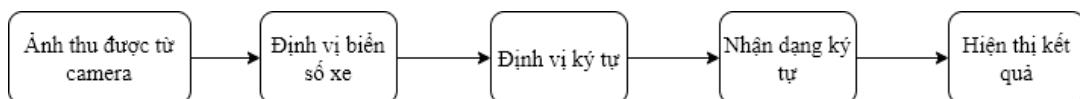
Hình 2.13: Kết quả đánh giá và so sánh YOLOv7 [35]

2.2 Mạng WPOD

Bài toán nhận diện biển số xe là một bài toán không còn mới, ứng dụng của nó đã được triển khai rộng rãi trong cuộc sống thường ngày như bãi đỗ xe thông minh, hệ thống thu phí không dừng tại các tuyến đường cao tốc. Hiện nay có rất nhiều các phương pháp khác nhau được đưa ra để giải quyết bài toán này, tuy nhiên, luồng xử

lý của các phương pháp đều có điểm chung như Hình 2.14 [41]. Do năng lực xử lý có hạn, thiết bị biên chỉ có thể thực hiện đến bước định vị vùng chứa biển số xe và gửi về máy chủ tập trung, còn các định vị và nhận diện ký tự sẽ được xử lý ở các máy chủ tùy theo từng ứng dụng cụ thể.

Nhiệm vụ của bài toán này là với một ảnh đầu vào cần trích xuất được toạ độ của vùng biển số xe trong ảnh và cắt ra được vùng biển số đó. Vì đây là một bài toán phát hiện vật thể, do vậy có nhiều phương pháp trong mục 2.1 đã được áp dụng. Vấn đề của các phương pháp này là khi biển số không còn được chụp thẳng chính diện với camera thì hình ảnh vùng không gian có chứa biển số xe được trả về sẽ bị méo, chứa nhiều chi tiết thừa trên thân xe (Hình 2.15). Do đó, gây khó khăn cho khâu phát hiện ký tự ở sau.



Hình 2.14: Luồng xử lý bài toán nhận diện biển số xe

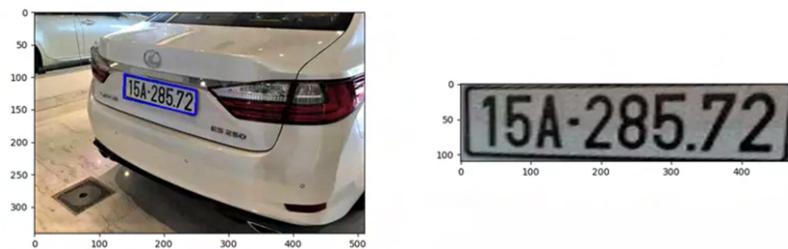


Hình 2.15: Vùng phát hiện biển số có nhiều chi tiết thừa do ảnh chụp không chính diện

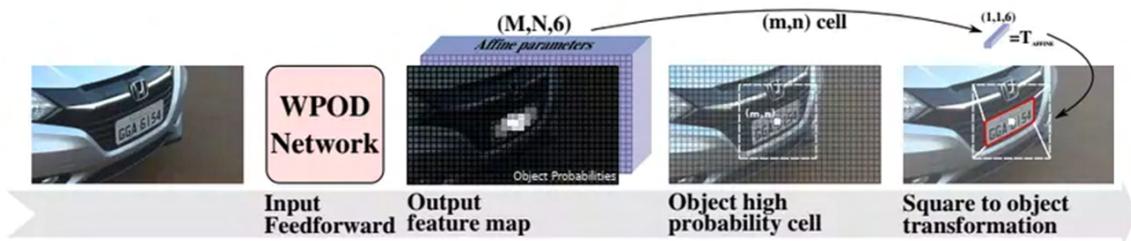
Mạng WPOD [42] được thiết kế để phục vụ cho bài toán phát hiện biển số xe. Trong khi các mạng YOLO và SSD chỉ trả về một hình chữ nhật bao quanh biển số xe mà không quan tâm đến không gian xung quanh biển số xe là như thế nào thì WPOD có thể trả về một vùng tứ giác bao quanh biển số xe và đưa biển số về hướng nhìn chính diện. Hình 2.16 biểu diễn kết quả đầu ra của mạng WPOD.

Hình 2.17 thể hiện cách thức hoạt động của mạng WPOD. Ảnh đầu vào thông qua quá trình lan truyền tiến qua mạng ta thu được bản đồ đặc trưng đầu ra gồm tám kẽm trong đó hai kẽm đầu tiên là xác suất có/không có biển số xe và sáu kẽm còn lại là những thông số để tính toán ma trận chuyển đổi góc nhìn của biển số xe.

Để trích xuất ra được vùng biển số xe thì nhóm tác giả đã xét một hình vuông với kích thước cố định (phần hình vuông viền trắng trên hình) xung quanh từng ô trên bản đồ đặc trưng đầu ra. Nếu xác suất có đối tượng của ô đó lớn hơn ngưỡng



Hình 2.16: Vùng biển số được phát hiện bởi mạng WPOD



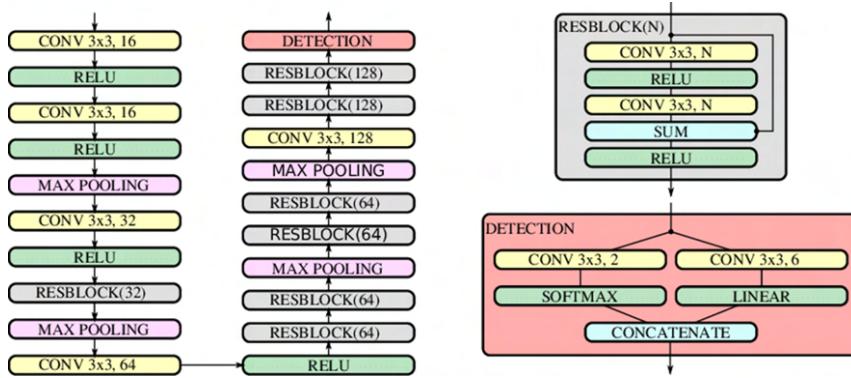
Hình 2.17: Nguyên lý hoạt động mạng WPOD [42]

quy định thì những giá trị của sáu kênh còn lại của ô đó sẽ được sử dụng để tính toán ma trận chuyển đổi từ vùng hình vuông về vùng biển số xe (vùng đa giác viền đỏ trên Hình 2.17). Và ta cũng sẽ sử dụng ma trận này để đưa biển số xe về hướng nhìn chính diện.

Kiến trúc của mạng bao gồm 21 lớp tích chập trong đó có 14 lớp là nằm trong các khối residual. Tất cả các lớp đều dùng bộ lọc với kích thước 3×3 và dùng hàm kích hoạt là ReLu ngoại trừ khối phát hiện đối tượng. Khối phát hiện là khối đáng chú ý nhất trong kiến trúc mạng này. Khối này có hai luồng riêng biệt, luồng thứ nhất dành cho việc tính toán xác suất có hoặc không có đối tượng (hai kênh đầu tiên của bản đồ đặc trưng đầu ra) với luồng này thì sử dụng hàm kích hoạt là softmax, luồng thứ hai phục vụ cho việc tính toán các thông số để tạo ra ma trận chuyển đổi hướng nhìn chính diện, với luồng này không sử dụng hàm kích hoạt hay nói cách khác hàm kích hoạt cho luồng này là hàm $f(x) = x$. Kết quả của hai luồng này sau đó sẽ được gộp với nhau để đưa ra kết quả như trong Hình 2.18. Từ những ưu điểm WPOD, đề tài đã ứng dụng mạng nơ-ron này vào mô hình phát hiện biển số xe trên thiết bị biến.

2.3 Theo vết đối tượng

Sau khi đã phát hiện được vật trong một khung hình, việc cần làm tiếp theo là theo dõi vị trí của vật đó theo thời gian. Phát hiện vật thể chỉ có chức năng xác định vị trí và phân loại đối tượng trong một khung hình duy nhất. Tuy nhiên, để xác định



Hình 2.18: Kiến trúc mạng WPOD [42]

đối tượng xuất hiện trong hai khung hình liên tiếp nhau là của cùng một vật thì chỉ bài toán phát hiện vật thể là không đủ, lúc đó, cần có thêm một thuật toán để theo dõi vật thể. Nói cách khác, bài toán theo vết đối tượng là bài toán ở mức độ cao hơn so với phát hiện đối tượng. Đối tượng đầu vào của bài toán không chỉ là vật thể chỉ xuất hiện một khung hình riêng lẻ mà là chuỗi các khung hình liên tiếp nhau. Trong bài toán này, bên cạnh việc xác định vị trí đối tượng, ta còn cần phải lưu ý đến các yếu tố khác như:

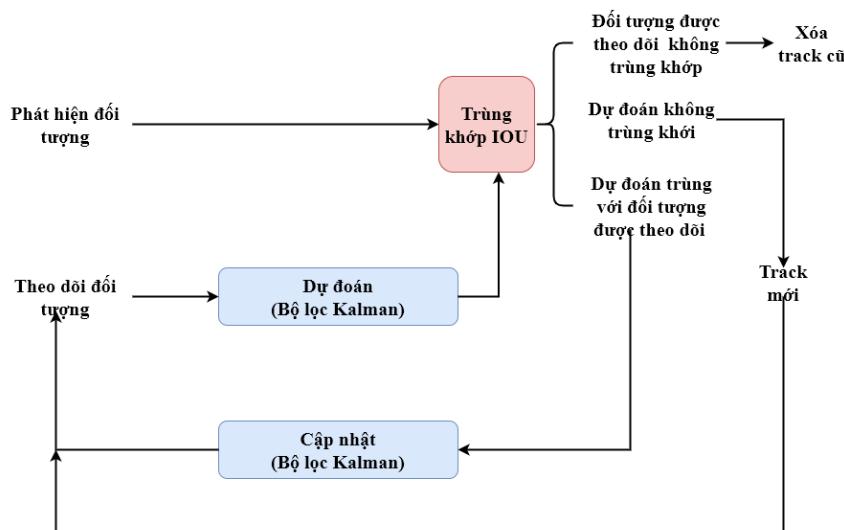
- Danh tính của vật thể được thể hiện qua ID và ID này cần đảm bảo luôn không đổi trong suốt quá trình vật thể đó xuất hiện trong một loại các khung hình.
- Hệ thống cần đảm bảo nhận diện được đúng ID của vật thể sau khi bị che khuất hoặc biến mất trong một vài khung hình.
- Cần đảm bảo tốc độ xử lý thời gian thực.

Theo dõi đối tượng là quá trình theo dõi các chuyển động của đối tượng và bảo tồn nhận dạng của nó, ngay cả khi ngoại hình hoặc chuyển động của nó thay đổi. Các phương pháp truyền thống cho theo dõi đối tượng, chẳng hạn như Multiple Hypothesis Tracking (MHT) [43], tạo ra nhiều đường theo dõi cho mỗi đối tượng và sử dụng một thuật toán kết hợp dữ liệu để chọn ra đường theo dõi có khả năng nhất ở mỗi bước thời gian. Tuy nhiên, những tiến bộ gần đây trong học sâu và mô hình phát hiện đối tượng với mạng nơ-ron đã dẫn đến sự phát triển nhanh chóng của các thuật toán theo dõi hiện đại. Những thuật toán này là sự kết hợp giữa mô hình phát hiện đối tượng và mô hình chuyển động. Mô hình chuyển động sử dụng các khung giới hạn được dự đoán của mô hình phát hiện ở một khoảng thời gian nhất định làm đầu vào để ước lượng các chuyển động của các đối tượng và xây dựng các thực thể. Ví dụ, SORT [44] là một thuật toán đơn giản sử dụng bộ lọc Kalman để dự đoán trạng thái của mỗi đối tượng trong mỗi khung hình và thuật

toán Hungarian để kết hợp các trạng thái được dự đoán với các đối tượng được phát hiện trong khung hình hiện tại để quyết định xem đối tượng nào thuộc về đối tượng nào được theo dõi. Ngoài ra, các phương pháp khác dựa trên ngoại hình, chẳng hạn như bộ lọc tương quan [45], [46], và các bộ theo dõi dựa trên biểu đồ màu cũng [47], [48] đã được sử dụng rộng rãi cho việc theo dõi đối tượng. Cũng có thể kể đến DeepSORT [49], nó kết hợp một mạng Siamese đã được tiền huấn luyện để phân biệt giữa các cá nhân và thêm thông tin thị giác vào quá trình ước lượng dẫn đến những cải tiến ấn tượng trong trường hợp che khuất nơi SORT không hoạt động được.

Tuy nhiên, đối với thiết bị có cấu hình thấp, cần hạn chế khối lượng tính toán và các phép tính phức tạp, do đó, SORT là sự lựa chọn tốt sử dụng để theo dõi đối tượng trong trường hợp này.

SORT là một thuật toán theo dõi dựa trên phát hiện. SORT bao gồm bốn thành phần chính: phát hiện, truyền trạng thái của các đối tượng được theo dõi sang các khung hình tương lai, kết hợp các đối tượng với bounding box được phát hiện ở khung hình hiện tại, và quản lý vòng đời của các đối tượng. Hình 2.19 mô tả luồng



Hình 2.19: Luồng xử lý của thuật toán SORT

xử lý của SORT. Tổng quan với mỗi khung hình mới, khung hình xử lý của thuật toán sẽ bao gồm các bước sau:

- Phát hiện: phát hiện vật thể trong khung hình đó.
- Dự đoán: dự đoán vị trí mới của vật thể dựa vào các khung hình trước đó.
- Liên kết: liên kết các vị trí đã được phát hiện với các vị trí được dự đoán để gán ID tương ứng.

SORT lấy kết quả đầu ra của mô hình phát hiện làm đầu vào của nó. Chất lượng dự đoán của mô hình phát hiện ảnh hưởng rất nhiều đến độ chính xác của SORT. Do đó, sự tiến bộ của học sâu với CNN đã cải thiện trực tiếp mô hình phát hiện đối tượng và gián tiếp nâng cao hiệu suất của thuật toán theo dõi này. Phần phát hiện đã được trình bày ở mục trước, ở phần này sẽ tập trung tìm hiểu hai pha sau.

a, Pha dự đoán với bộ lọc Kalman

SORT sử dụng một mô hình ước lượng để truyền trạng thái của các đối tượng được theo dõi sang các khung hình tương lai bằng cách sử dụng mô hình vận tốc không đổi tuyến tính. Mô hình này sẽ ước lượng vị trí và vận tốc của mỗi thực thể, tiến hành cập nhật lại sau khi phát hiện vị trí thực tế. Toàn bộ quy trình này được thực hiện thông qua bộ lọc Kalman [50]. Nếu không có phát hiện nào được kết hợp với đối tượng được theo dõi này, trạng thái của nó sẽ được ước lượng trực tiếp bởi mô hình vận tốc không đổi tuyến tính mà không có bất kỳ sự điều chỉnh nào.

Bộ lọc Kalman [20] là mô hình được giới thiệu lần đầu năm 1960. Hiện tại, ứng dụng của bộ lọc Kalman rất đa dạng: xe tự lái, thực tế ảo, kinh tế lượng, theo vết, điều khiển tối ưu. Đối với bài toán truy vết đối tượng, bộ lọc Kalman được sử dụng để dự đoán các trạng thái của đối tượng hiện tại dựa vào các theo dõi trong quá khứ và cập nhật lại các đối tượng sau khi đã được liên kết với các theo dõi trước đó. Bộ lọc Kalman mô hình hóa quá trình ngẫu nhiên với các trạng thái, mô hình được xác định từ trước như trong phương trình dưới:

$$x_k = \theta_{k-1}(x_{k-1}, u_{k-1}), x_k \in \mathbb{R}^n \quad (2.10)$$

$$z_k = h_{k-1}(x_{k-1}, \omega_{k-1}), z_k \in \mathbb{R}^m \quad (2.11)$$

Trong đó,

- x_k : là trạng thái thực sự của quá trình nhưng không quan sát được (do tác động của nhiễu), là biến trạng thái ẩn.
- z_k : là trạng thái của quá trình thu được thông qua sự quan sát, đo đạc, là biến quan sát được.
- θ_k, h_k : là các mô hình được định nghĩa trước.
- u_k, ω_k : là nhiễu của quá trình (khách quan) và nhiễu đo đạc(chủ quan).

Có nhiều loại bộ lọc Kalman như bộ lọc Linear Kalman, Extended Kalman, Unscented Kalman. Đối với thuật toán SORT, thuật toán giả thiết rằng các đối tượng di chuyển với vận tốc không đổi, từ đó thuật toán áp dụng bộ lọc Linear Kalman để mô hình

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

hóa quá trình chuyển động của các đối tượng. Bộ lọc Linear Kalman giả định mô hình của các quá trình θ_k , h_k đều là các mô hình tuyến tính:

$$x_k = F_{k-1}x_{k-1} + u_{k-1} \quad (2.12)$$

$$z_k = H_{k-1}x_k + u_k \quad (2.13)$$

trong đó:

- $x_k \sim N(\hat{x}_k, \Sigma_k)$ là trạng thái thực sự của quá trình tại thời điểm k .
- F_k, H_k lần lượt là ma trận chuyển trạng thái và ma trận đo đạc.
- $u_k \sim N(0, Q_k)$, $w_k \sim N(0, R_k)$ biểu thị nhiễu quá trình và nhiễu đo đạc.

Nhiệm vụ bây giờ là cần xác định được \hat{x}_k, Σ_k tại thời điểm k , từ đó chúng ta có thể đưa ra một ước lượng về thông tin của x_k . Bộ lọc Kalman thực hiện điều đó thông qua hai bước sau:

- Bước 1: Dự đoán

$$\hat{x}_k = E(x_k) = E(F_{k-1}x_{k-1} + u_{k-1}) = F_{k-1}\hat{x}_{k-1} \quad (2.14)$$

$$\Sigma_k = V(F_{k-1}x_{k-1} + u_{k-1}) = F_{k-1}\Sigma_{k-1}F_{k-1}^T + Q_k \quad (2.15)$$

- Bước 2: Hiệu chỉnh: Công thức ở trên chưa phải công thức cuối cùng vì vẫn chưa xét đến sự ảnh hưởng của nhiễu đo đạc. Đó là lí do cần thêm bước hiệu chỉnh. Từ $z_k = H_{k-1}x_k + u_k$, bộ lọc Kalman đưa ra công thức cho phép hiệu chỉnh như sau:

$$\hat{x}_k^F = \hat{x}_k + K_k(z_k - H_k\hat{x}_k) \quad (2.16)$$

$$\Sigma_k^F = \Sigma_k(I - K_kH_k) \quad (2.17)$$

trong đó, $K_k = \Sigma_k H_k^T (R_k + H_k \Sigma_k H_k^T)^{-1}$ gọi là hệ số Kalman tại thời điểm k .

Để áp dụng bộ lọc Kalman Filter vào bài toán truy vết, thuật toán SORT mô hình hóa trạng thái ban đầu của mỗi đối tượng qua một vec-to $X = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}]$ trong đó:

- X chỉ trạng thái của quá trình, có ma trận hiệp phương sai ban đầu được khởi tạo với giá trị lớn để thể hiện sự không chắc chắn của trạng thái.
- x, y lần lượt là tọa độ của tâm đối tượng (ở đây là tâm bounding box).
- s là diện tích của bounding box.

- r là tỉ lệ của hai cạnh bounding box.
- $\dot{x}, \dot{y}, \dot{s}$ lần lượt là các giá trị vận tốc tương ứng của x, y, s .

Do giả định các đối tượng chuyển động đều, ta có $X_k = F_{k-1}X_{k-1} + u_{k-1}$ trong đó:

$$F = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.18)$$

trong đó u_k được khởi tạo tuân theo phân phối chuẩn có $mean = 0$ và $covariance Q$ không đổi. Sau khi dự đoán, bước tiếp theo là hiệu chỉnh để tìm ra các thông số của bounding box. Công thức hiệu chỉnh như sau:

$$\begin{bmatrix} x'_k \\ y'_k \\ s'_k \\ r'_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} * \begin{bmatrix} x_k \\ y_k \\ s_k \\ r_k \\ \dot{x}_k \\ \dot{y}_k \\ \dot{s}_k \end{bmatrix} + w_k \quad (2.19)$$

Từ đó, với đầu vào là trạng thái của khung hình trước, bộ lọc Kalman sẽ cho ra dự đoán của trạng thái của khung hình tiếp theo.

b, Liên kết với thuật toán Hungarian

Để gán các phát hiện cho các mục tiêu đã được xác định trước, SORT sẽ ước lượng trước trạng thái của mỗi mục tiêu trong khung hình hiện tại là khung giới hạn của nó. Sau đó, SORT tính toán chỉ số khoảng cách giữa các khung giới hạn được ước lượng và các phát hiện từ mô hình phát hiện đối tượng. Chỉ số này là IoU và việc gán sẽ được giải quyết bởi thuật toán Hungarian. Quá trình này giúp thuật toán xác định phát hiện nào nên được gán cho mục tiêu nào dựa trên hình học của chúng khung giới hạn tương ứng. Hơn nữa, một ngưỡng tối thiểu được đặt cho việc gán để không ghép các đối tượng được theo dõi với các phát hiện khác biệt

so với các trạng thái được ước lượng. Điều này có thể ngăn chặn lỗi do che khuất trong thời gian ngắn.

Thuật toán Hungarian được phát triển và công bố vào năm 1955, đề xuất để giải bài toán phân công công việc. Bài toán có thể được phát biểu như sau: Có n người ($i = 1, 2, 3, \dots, n$) và n công việc ($j = 1, 2, \dots, n$). Để giao cho người i thực hiện một công việc j cần một chi phí c_{ij} . Bài toán đặt ra là cần giao cho người nào làm việc gì sao cho mỗi người chỉ làm một việc, mỗi việc chỉ do một người làm, và tổng chi phí là nhỏ nhất. Bài toán có thể được mô hình hóa lại như sau:

$$\max \sum_{i,j} x_{ij} c_{ij} \quad (2.20)$$

trong đó:

$$\sum_{i=1}^n x_{ij} = \sum_{j=1}^n x_{ij} = 1 \quad (2.21)$$

có nghĩa mỗi người chỉ làm một việc và mỗi việc chỉ có duy nhất một người làm, $x_{ij} = 1$ nếu người i làm việc j và bằng 0 nếu ngược lại. Nhiệm vụ của bài toán là xác định $x_{ij}, i = 1, 2, \dots, n; j = 1, 2, \dots, n$. Thuật toán Hungarian sử dụng các phép biến đổi trên ma trận chi phí $C = [c_{ij}]$ để tìm ra được lời giải tối ưu cho bài toán này với các bước như sau:

- Bước 1: Trừ các phần tử trên mỗi hàng của C cho phần tử nhỏ nhất trên hàng đó, tiếp theo trừ các phần tử trên mỗi cột cho phần tử nhỏ nhất trên cột đó. Kết quả nhận được ma trận C' có tính chất: trên mỗi hàng, cột có ít nhất một phần tử 0 và bài toán giao việc với ma trận C' có cùng lời giải như bài toán với ma trận C .
- Bước 2: Vẽ một số tối thiểu các đường thẳng trên dòng và cột để đảm bảo mọi phần tử 0 đều được đi qua.
- Bước 3: Nếu có n đường thẳng được vẽ, kết thúc thuật toán và tiến hành phân công việc. Nếu số đường thẳng được vẽ nhỏ hơn n , vẫn chưa tìm được phương án phân công tối ưu, tiến hành bước tiếp theo.
- Bước 4: Mỗi hàng (hoặc cột) có đường thẳng vẽ qua, ta gọi các hàng (cột) đó là các hàng (cột) thiết yếu. Các hàng (cột) còn lại là các hàng (cột) không thiết yếu. Tìm phần tử nhỏ nhất không nằm trong các hàng (cột) thiết yếu, tiến hành trừ mỗi hàng không thiết yếu cho phần tử nhỏ nhất đó, và cộng giá trị nhỏ nhất ấy cho cột thiết yếu. Ta được ma trận C'' có cùng lời giải với ma trận C' . Sau

đó quay lại Bước 2.

Quay lại bài toán truy vết, sau khi mô hình phát hiện đối tượng phát hiện được vật thể, ta cần phải kết hợp được chúng với những truy vết được bộ lọc Kalman dự đoán, để từ đó, thuật toán có thể gán được ID chính xác cho đối tượng đang theo vết. Thuật toán SORT xây dựng ma trận chi phí C bằng việc tính toán chỉ số IoU giữa các bounding box được dự đoán bởi bộ lọc Kalman với những bounding box của bộ phát hiện vật thể. Sau đó, SORT sử dụng thuật toán Hungarian để thực hiện kết hợp.

Áp dụng hai phương pháp nói trên, thuật toán SORT thực hiện truy vết theo quy trình các bước sau:

- Bước 1: SORT tiến hành sử dụng bộ lọc Kalman để dự đoán các trạng thái truy vết mới dựa trên các truy vết trong quá khứ.
- Bước 2: Sử dụng những truy vết vừa dự đoán được, kết hợp với các phát hiện thu được từ bộ phát hiện đối tượng, xây dựng ma trận chi phí C . Chi phí được sử dụng để đánh giá ở đây là giá trị IOU giữa các bounding box được dự đoán và bounding box được phát hiện bởi bộ phát hiện đối tượng.
- Bước 3: Sử dụng giải thuật Hungarian giải bài toán chia công việc với ma trận chi phí C vừa lập.
- Bước 4: Xử lý, phân loại các bounding box thu được từ các bộ phát hiện.
- Bước 5: Sử dụng bộ lọc Kalman để cập nhật những vật thể đã được liên kết với những vết của nó.

Khi các đối tượng xuất hiện hoặc biến mất khỏi hình ảnh, thuật toán tạo ra hoặc loại bỏ các nhận dạng theo yêu cầu. Một bộ theo dõi mới được tạo khi có một phát hiện không thể khớp với bất kỳ đối tượng hiện có nào. Bộ theo dõi này sau đó được khởi tạo sử dụng bounding box được phát hiện và vận tốc được đặt bằng không. Hơn nữa, đối tượng mới được theo dõi sẽ được giám sát trong một vài khung hình liên tiếp theo nơi nó cần phải được khớp với các phát hiện từ mô hình phát hiện đối tượng để tích lũy sự tự tin và trở thành một nhận dạng mới chính thức.

2.4 Thiết bị AI tại biên

Các mạng nơ-ron học sâu cho kết quả tốt nhất khi được xử lý trên môi trường GPU mạnh mẽ trong các máy chủ hoặc máy tính cấu hình cao. Ngược lại, các thiết bị IoT đặt tại biên thường khó có thể chạy các mạng học sâu hoặc cho kết quả rất thấp khi chạy do không được trang bị cấu hình đầy đủ như GPU, CPU, RAM. Nhận

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

Bảng 2.1: So sánh cấu hình phần cứng các máy tính nhúng thuộc dòng Jetson của NVIDIA [51]–[54]

Jetson NANO		Jetson TX2		Jetson AGX Xavier		
		4GB	8GB	Công nghiệp	8GB	16GB
GPU	128-Core Maxwell GPU với CUDA Cores		256-Core Pascal GPU với CUDA Cores		384 Core Volta + NVLDA	512 Core Volta + NVLDA
CPU	Quad-core ARM Cortex-A57	Dual-Core NVIDIA Denver 2 Quad-Core ARM Cortex®-A57		6-core Carmel ARM CPU 1.3 GHZ	8-core Carmel ARM CPU 2.26 GHz	
RAM	4 GB 64-bit LPDDR4	4 GB 128-bit LPDDR4	8 GB 128-bit LPDDR4	8 GB 256-bit LPDDR4x	16 GB 256-bit LPDDR4x	
Ổ cứng	16 GB eMMC 5.1	16 GB eMMC 5.1	32 GB eMMC 5.1		32 GB eMMC 5.1	
Mã hóa video	4K @ 30 (H.264/H.265)		2 × 4K @ 30 (HEVC)	2×4K @ 60 (HEVC)	4×4K @ 60 (HEVC)	
Giải mã video	4K @ 60 (H.264/H.265)		2 × 4K @ 30, 12-bit support	2 × 4K @ 30 (HEVC)	2 × 8K @ 30 (HEVC)	
Camera	12 lanes (3 × 4 or 4 × 2) MIPI CSI-2 DPHY 1.1 (1.5 Gbps)		12 lanes MIPI CSI-2, D-PHY 1.2 (30 Gbps)	16 lanes MIPI CSI-2 D-PHY 1.2 (40 Gbps)		
Kết nối	Gigabit Ethernet		Gigabit Ethernet		Gigabit Ethernet	
Hiển thị	HDMI 2.0 DP 1.2 eDP 1.4 DSI (1 × 2) 2		HDMI 2.0 DP 1.2 eDP 1.4 DSI (1 × 2) 2		HDMI 2.0 eDP 1.4, DP HBR3	
UPHY	1 × 1/2/4 PCIE, 1 × USB 3.0, 3 × USB 2.0		Gen2 1 × 4 + 1 × 1 hoặc 2 × 1 + 1 × 2, USB 3.0 + USB 2.0		(8×) PCIe Gen4 (8×) SLVS-EC (3×) USB 3.1	
Ngoại vi	SDIO, SPI, I2C, I2S, GPIOs		CAN, UART, SPI, I2C, I2S, GPIOs		UART, SPI, CAN, I2C, I2S, DMIC, GPIOs	
Kích thước	69.6 mm × 45 mm		87 mm × 50 mm		87 mm × 100 mm	
Số pin	260-pin Connector		400-pin connector		699 pin Connector	
Năng lực tính toán	472 GFLOPs	472 GFLOPs	1.3 TFLOPs	10 TFLOPs	16 TFLOPs	
Công suất tiêu thụ	5/10 W	7.5/15 W	10/20 W	10/20 W	10/15/30 W	

thấy tiềm năng của việc phát triển các ứng dụng IoT kết hợp học máy, NVIDIA đã cho ra đời nền tảng máy tính nhúng Jetson với ba sản phẩm chính: Nano, TX2 và AGX Xavier [51]–[54]. Ba thiết bị này có kích thước nhỏ gọn, trang bị đầy đủ CPU, GPU, RAM và ổ cứng với thông số kỹ thuật khác nhau tùy theo từng mức giá. Thông số kỹ thuật của các sản phẩm phần cứng thuộc dòng Jetson được mô tả như trong Bảng 2.1.

Jetson TX2 mang lại tốc độ và hiệu suất năng lượng vượt trội trong số các thiết bị máy tính nhúng, cho phép thực hiện các phép tính trong mạng nơ-ron học sâu. TX2 có ba phiên bản đó là TX2 4GB, TX2 8GB và TX2i dành cho công nghiệp.

Jetson AGX Xavier là một máy tính nhúng được thiết kế đặc biệt cho các hệ thống tự động kết hợp trí thông minh nhân tạo. Nó được thiết kế với cấu hình phần cứng tốt nhất trên thị trường hiện nay. Nó có thể dễ dàng xử lý dữ liệu cảm biến và các mạng học sâu với hiệu suất cao, tiêu thụ ít năng lượng.

Jetson Nano là một máy tính nhúng nhỏ gọn nhưng vô cùng mạnh mẽ, cho phép người dùng chạy song song nhiều mạng nơ-ron học sâu cho các ứng dụng xử lý hình ảnh và dữ liệu từ cảm biến. Đây là nền tảng lý tưởng để tích hợp các ứng dụng trí tuệ nhân tạo tiên tiến vào các sản phẩm nhúng, được sử dụng rộng rãi trong các ứng dụng IoT, bao gồm các thiết bị thu hình, robot, và gateway thông minh. Hình 2.20 là thiết bị Jetson Nano.



Hình 2.20: Máy tính nhúng Jetson Nano của NVIDIA

So với Jetson Nano, Jetson TX2 và AGX Xavier được trang bị phần cứng cao cấp hơn hẳn đặc biệt là bộ xử lý CPU và GPU mạnh mẽ, do đó, chúng có khả năng thực thi các phép tính song song trong mô hình học sâu có kiến trúc phức tạp với tốc độ cao. Tuy nhiên, xét theo góc độ kinh tế, thiết bị Jetson Nano có giá thành hợp lý đối với đại đa số người dùng và trong triển khai số lượng lớn thiết bị. Vì vậy, đề tài lựa chọn thiết bị Jetson Nano để xây dựng các ứng dụng học sâu nhằm mục đích có thể triển khai áp dụng thực tế.

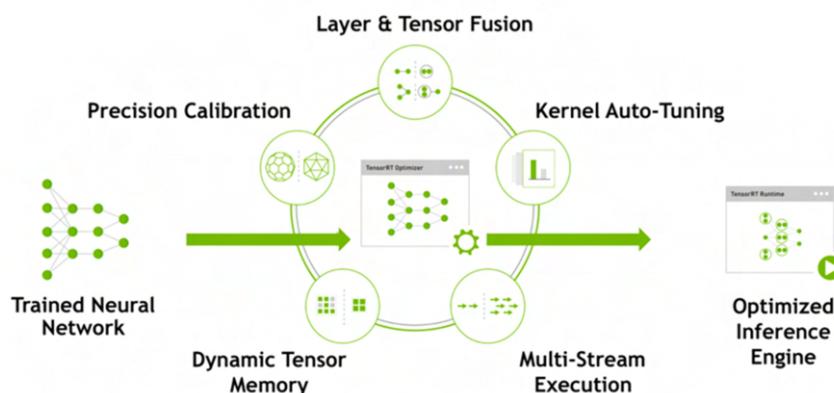
2.5 Thư viện hỗ trợ

2.5.1 TensorRT

TensorRT [55] là một bộ công cụ phát triển phần mềm (SDK) do công ty NVIDIA cung cấp nhằm tối ưu hóa các mô hình học sâu đã được đào tạo từ trước, TensorRT hỗ trợ trên nhiều loại mô hình khác nhau như: TensorFlow, Tflite, YOLO, SSD, Mobilnet. TensorRT giúp người dùng chuyển định dạng các mô hình đã được đào tạo từ trước về mô hình TensorRT có định dạng .trt nhằm tối ưu hóa các thông

số: FPS, bộ nhớ RAM, GPU, CPU, từ đó khi mô hình chạy trên các thiết bị nhúng của hãng NVIDIA sẽ đạt được hiệu suất cao nhất. Hiện nay, TensorRT đã được tích hợp sẵn vào hệ điều hành Jetpack 4.5 của Jetson Nano. Năm loại tối ưu để tăng hiệu suất suy luận được TensorRT thực hiện như Hình 2.21:

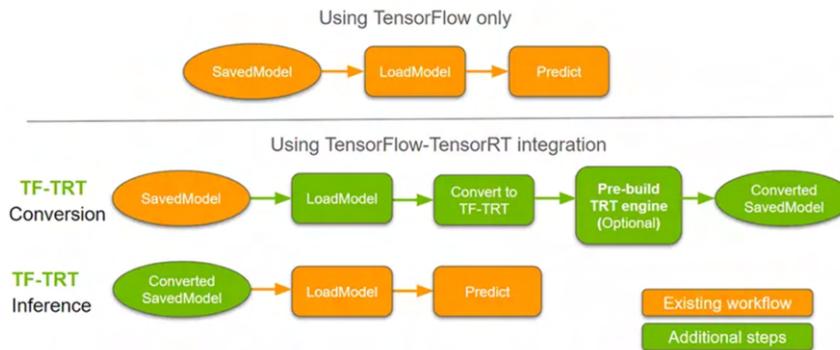
- Precision Calibration: trong suốt quá trình training, các tham số và hàm kích hoạt activations trong độ chính xác FP32 sẽ được chuyển đổi về độ chính xác FP16 hoặc INT8. Việc tối ưu nó sẽ giảm độ trì truệ và tăng tốc độ suy luận nhưng phải trả một cái giá là phải giảm độ chính xác của mô hình mặc dù không đáng kể. Trong nhận diện theo thời gian thực thì đôi khi việc đánh đổi độ chính xác so với tốc độ suy luận là cần thiết.
 - Layer & Tensor Fusion: TensorRT sẽ gộp các lớp và các tensor để tối ưu hóa bộ nhớ GPU cùng bằng thông qua việc gộp các mô hình theo chiều dọc, chiều ngang hoặc cả hai.
 - Kernel auto-tuning: một vài kernel riêng dành cho việc tối ưu sẽ thực thi trong suốt tiến trình.
 - Dynamic Tensor Memory: phân bổ bộ nhớ cần thiết cho mỗi tensor trong thời gian sử dụng giúp giảm thiểu lượng bộ nhớ và cải thiện khả năng sử dụng lại bộ nhớ.
 - Multiple Stream Execution: cho phép nhận nhiều luồng dữ liệu đầu vào một cách song song.



Hình 2.21: Năm loại tối ưu trên TensorRT [55]

Để chuyển đổi về định dạng TensorRT, mô hình phải có bước chuyển đổi trung gian về định dạng *.uff* (thích hợp với các mô hình TensorFlow) hoặc *.onnx* (thích hợp với nhiều loại mô hình khách nhau). Ngoài hai định dạng này ra, hiện nay TensorRT chưa hỗ trợ định dạng nào khác. Sau đó, sử dụng hàm chuyển đổi

của TensorRT SDK để chuyển đổi từ *.onnx* hoặc *.uff* sang định dạng TensorRT. Hình 2.22 thể hiện các bước chuyển đổi từ TensorFlow sang TensorRT.



Hình 2.22: Luồng chuyển đổi mô hình từ TensorFlow sang TensorRT [55]

2.5.2 Darknet

Darknet [56] là một framework mã nguồn mở cho việc xây dựng và triển khai các mạng nơ-ron với hiệu suất cao [56]. Nó được xây dựng dựa trên ngôn ngữ C và CUDA, do đó có thể được tích hợp cả với CPU và GPU. Một số mạng được nơ-ron được triển khai trên darknet bao gồm YOLOv1 đến YOLOv4, các mạng phân loại hình ảnh ResNet, ResNext, và các mạng RNN. Đặc biệt, Darknet được phát triển Josheph Redmon, cũng là người phát triển mạng YOLO các phiên bản từ một đến bản bốn. Do đó, đây là framework phù hợp nhất để huấn luyện và chạy mô hình YOLOv4 sử dụng trong nghiên cứu.

2.5.3 PyTorch

PyTorch là framework được phát triển bởi Facebook. PyTorch cùng với TensorFlow và Keras là một trong những framework phổ biến được sử dụng trong các bài toán về học sâu hiện nay[57]. Đặc biệt, trong các lĩnh vực nghiên cứu, hầu như các tác giả đều sử dụng PyTorch để triển khai bài toán của mình. PyTorch có một số đặc điểm sau:

- PyTorch được viết bằng C++ và CUDA, với cơ chế song song trên GPUs. Kèm với đó, PyTorch cung cấp khả năng chạy trực tiếp từ C++ hỗ trợ tốc độ cho việc triển khai mạng học sâu trên sản phẩm.
- PyTorch có thể kết hợp với các thư viện khác trong python.
- PyTorch cung cấp mảng nhiều chiều Tensors, hỗ trợ nhiều kiểu dữ liệu, các phép toán trên tensor có thể tính toán trên cả CPU lẫn GPU và có thể chuyển đổi qua lại.
- Engine autograd của PyTorch đi kèm trong Tensors giúp truy vết được vi phân

của bất kì kết quả nào theo đầu vào tương ứng. Việc này cực kỳ hữu ích trong sử dụng những kỹ thuật để phân tích các lớp trong mạng học sâu.

- PyTorch cũng cung cấp sẵn một lượng tương đối các mạng học sâu đã được huấn luyện sẵn để người dùng có thể xây dựng nhanh các ứng dụng.

2.5.4 OpenCV

OpenCV [58] là thư viện mã nguồn mở hàng đầu cho thị giác máy tính, xử lý ảnh và máy học, và các tính năng tăng tốc GPU trong hoạt động thời gian thực. OpenCV có các thư viện hỗ trợ lập trình bằng ngôn ngữ C++, C, Python, Java. OpenCV cũng hỗ trợ Windows, Linux, Mac OS, iOS và Android. OpenCV được thiết kế để tính toán hiệu quả và với sự tập trung nhiều vào các ứng dụng thời gian thực. Được viết bằng tối ưu hóa C/C++, thư viện có thể tận dụng lợi thế của xử lý đa lõi. Thư viện OpenCV đã được sử dụng trong nhiều ứng dụng khác nhau từ nghệ thuật tương tác, cho đến lĩnh vực khai thác mỏ, bản đồ trên web hoặc công nghệ robot. Một số mô-đun phổ biến có sẵn của OpenCV như sau:

- Core: mô-đun nhỏ gọn để xác định cấu trúc dữ liệu cơ bản, bao gồm mảng đa chiều dày đặc và nhiều chức năng cơ bản được sử dụng bởi tất cả các mô-đun khác.
- Mô-đun xử lý ảnh (*imgproc*): các thuật toán lọc hình ảnh tuyến tính và phi tuyến, phép biến đổi hình học, chuyển đổi không gian màu, biểu đồ.
- Mô-đun phân tích video (*video*): các tính năng ước tính chuyển động, tách nền, và các thuật toán theo dõi vật thể.
- Mô-đun chỉnh sửa ảnh và tái tạo hình ảnh 3D(*calib3d*): thuật toán hình học đa chiều cơ bản, hiệu chuẩn hình ảnh đơn kênh và đa kênh, dự đoán kiểu dáng của đối tượng, và các yếu tố tái tạo 3D.
- Mô-đun đặc trưng trong không gian 2 chiều (*features2d*): phát hiện các đặc tính nổi bật của bộ nhận diện, bộ truy xuất thông số, thông số đối chọi.
- Mô-đun phát hiện đối tượng (*objdetect*): phát hiện các đối tượng và mô phỏng của các hàm được định nghĩa sẵn.
- Mô-đun xây dựng giao diện (*highgui*): để thực hiện việc thiết kế các giao diện người-máy giao tiếp đơn giản.
- Mô-đun xử giải mã, mã hóa video (*videoio*): các thuật toán giải mã và mã hóa luồng video thu và xuất ra.
- Mô-đun GPU: Các thuật toán tăng tốc GPU từ các mô-đun OpenCV.

2.5.5 ZMQ

Để truyền hình ảnh và thông tin theo dõi của một người về máy chủ, nghiên cứu lựa chọn sử dụng thư viện ZMQ. Đây là một thư viện mã nguồn mở hỗ trợ truyền thông điệp để xây dựng các ứng dụng phân tán với tên gọi chính thức là ZeroMQ [59]. Mục tiêu của ZeroMQ là giảm thiểu sự phức tạp và chi phí của việc phát triển và triển khai ứng dụng mạng phân tán, cho phép các ứng dụng truyền tải thông điệp một cách tin cậy, nhanh chóng, bảo mật và hiệu quả.

Triết lý của ZMQ là không có trung gian, không có độ trễ, chi phí bằng không và không quản trị. ZMQ sử dụng dịch vụ giao vận TCP. ZMQ xây dựng và khai thác một giao thức ứng dụng riêng có tên gọi là ZeroMQ Message Transport Protocol (ZMTP). ZMTP sử dụng giao thức giao vận TCP. ZMTP là giao thức được tối ưu cho tốc độ truyền thông.

ZMQ hỗ trợ nhiều mô hình truyền thông điệp trong nhiều loại ứng dụng khác nhau:

- Mô hình publish - subscribe: đây cũng là mô hình được sử dụng bởi giao thức MQTT. Tuy nhiên, mô hình publish - subscribe trong ZMQ không sử dụng một broker riêng. Một trong số các ứng dụng sẽ đóng vai trò publisher. Các ứng dụng khác đóng vai trò subscriber và kết nối đến publisher. ZMQ cũng hỗ trợ mô hình mở rộng của publish-subscribe gọi là Xpub-Xsub với nhiều publisher, subscriber.
- Mô hình client - server: đây là mô hình thường gặp trong giao thức HTTP.
- Mô hình pipeline: còn gọi là mô hình push - pull. Trong mô hình này, các ứng dụng được tổ chức theo một chiều liên tiếp từ nguồn đến đích.

Để áp dụng ZMQ, đề tài đã sử dụng thư viện ImageZMQ, một thư viện mã nguồn mở chuyên dùng để truyền các hình ảnh OpenCV giữa các máy tính với nhau. ImageZMQ cho phép máy trung tâm nhận và xử lý hình ảnh từ nhiều nguồn gửi cùng một lúc. ImageZMQ là một giao thức điểm đến điểm, bỏ qua broker ở giữa, do đó, giảm thiểu việc phải xử lý một hình ảnh hai lần. Đặc biệt là đối với hình ảnh trong OpenCV thường lớn hơn các tin nhắn văn bản đơn giản, vì vậy, tăng tốc độ truyền dữ liệu cao giữa các nguồn gửi hình ảnh và trung tâm xử lý hình ảnh.

Trong chương này, đề tài đã trình bày các nghiên cứu và lý thuyết nền tảng. Trong bài toán phát hiện đối tượng, các bộ phát hiện một pha mang lại tốc độ xử lý cao với độ chính xác tương đương các bộ hai pha, nổi bật trong số đó là mạng YOLO. Đề tài lựa chọn YOLO để áp dụng cho các bài toán về phát hiện xe cũng

CHƯƠNG 2. NỀN TẢNG LÝ THUYẾT

nhiều phát hiện người. Bên cạnh đó, lý thuyết về bộ phát hiện biển số xe cũng đã được đề cập tại chương này, trong đó, mạng WPOD đã khắc phục được nhược điểm về góc chụp của biển số xe so với các phương pháp khác, cho ra hình ảnh biển số với ít các chi tiết thừa và góc chính diện. Trong số các thiết bị máy tính nhúng AI thuộc dòng Jetson của NVIDIA, Jetson Nano cân bằng giữa khả năng xử lý mô hình học sâu và giá thành cho mục đích thương mại. Chương 3 sẽ trình bày ứng dụng trích xuất hình ảnh biển số trên thiết bị Jetson Nano.

CHƯƠNG 3. BÀI TOÁN PHÁT HIỆN BIỂN SỐ XE TRÊN THIẾT BỊ BIÊN

Phần này nghiên cứu giải pháp cho bài toán phát hiện phương tiện và biển số xe trên thiết bị biên Jetson Nano, ứng dụng đối với giao thông tại Việt Nam. Luồng xử lý bao gồm hai mô-đun đó là phát hiện xe trên đường phố và phát hiện biển số. Đối với mô-đun phát hiện phương tiện, đề tài thực hiện nghiên cứu và thực nghiệm hai mạng học sâu đó là YOLOv4-tiny và YOLOv7-tiny. Trong khi đó, mạng WPOD được sử dụng cho mô-đun phát hiện biển số xe. Chương này có cấu trúc như sau: phần đầu mô tả các công trình liên quan về giám sát giao thông trên thiết bị biên. Các giải pháp được trình bày trong phần hai. Tiếp theo là mô tả thực nghiệm và kết quả đạt được, cuối cùng là kết luận chương.

Các kết quả nghiên cứu đạt được trong chương 3 đã bao hàm bài báo “Lightweight Model’s Performances on a Resource-Constrained Device for Traffic Application” được trình tại hội thảo quốc tế AICI2023 và đã được đưa vào ký yếu của hội nghị [60].

3.1 Các nghiên cứu liên quan

Phát hiện biển số xe là bài toán định vị biển số xe trong hình ảnh, video chứa các phương tiện giao thông. Các hệ thống phát hiện biển số xe tự động đang được nhiều quốc gia đưa vào trong các ứng dụng quản lý đô thị thông minh dựa trên AI như phát hiện vi phạm giao thông, quản lý bãi đỗ xe thông minh hay giám sát tốc độ phương tiện [41]. Trong một hệ thống phát hiện biển số xe tự động có thể gồm một hoặc nhiều pha, kết hợp của nhiều mô hình học sâu khác nhau. Pha đầu tiên đó là phát hiện biển số xe, sau đó là tách ký tự và cuối cùng là nhận dạng ký tự trên biển số [61]. Tại khâu phát hiện biển số, các phương pháp sử dụng mô hình học sâu được đánh giá cao bởi sự ổn định và độ chính xác khi điều kiện ánh sáng, bối cảnh thay đổi [41].

Đã có một số nghiên cứu về bài toán phát hiện biển số chạy trên thiết bị biên, có thể kể đến như nghiên cứu của Cheng-Jian Lin [41]. Trong đó, các tác giả sử dụng thiết bị phần cứng có cấu hình cao là Jetson AGX XAVIER và hai mô hình YOLOv4-tiny nối tiếp để phát hiện và nhận diện ký tự trên biển số xe. Nghiên cứu [62] lại sử dụng thiết bị Raspberry Pi 4 để phát hiện phương tiện và biển số xe tại các trạm thu phí. Mạng sử dụng trong nghiên cứu này là YOLOv3-tiny, YOLOv4-tiny, và Faster R-CNN cho kết quả lên đến 97% mAP@0,5 đối với YOLOv4-tiny. Ngoài ra, một nghiên cứu kết hợp giữa Raspberry Pi và Intel Neural

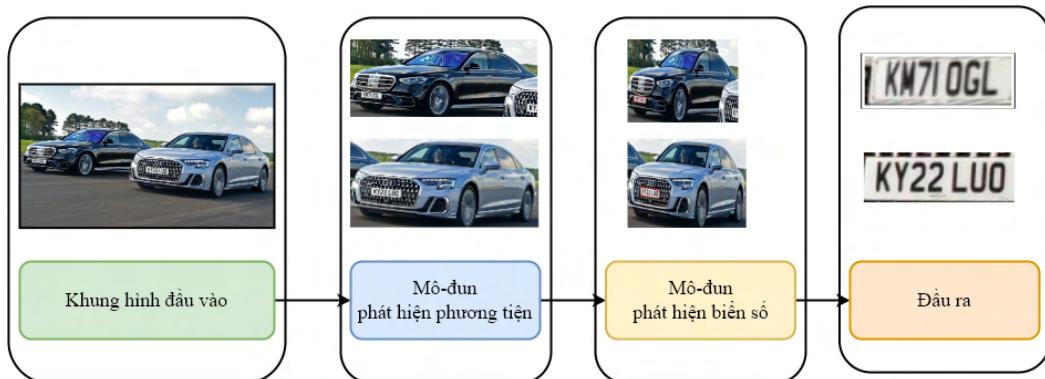
Compute Stick cũng được thử nghiệm triển khai mạng RPNet, TE2E, và YOLOv3 trong ứng dụng phát hiện biển số và vật thể [63].

Các nghiên cứu trên đều sử dụng hình ảnh trích xuất từ các camera được đặt ở góc chính diện, gần với phương tiện, do đó kết quả của mô hình phát hiện phương tiện cũng như nhận diện biển số ít bị ảnh hưởng bởi yếu tố góc chụp, khoảng cách, cũng như độ sáng. Bên cạnh đó, các tập dữ liệu được sử dụng để huấn luyện mô hình và kiểm tra độ chính xác chỉ bao gồm một loại phương tiện là xe ô tô, điều này chưa phù hợp khi đem áp dụng tại Việt Nam, nơi có sự tham gia của nhiều loại phương tiện trên đường.

Từ đó, nghiên cứu đề xuất phương pháp để phát hiện biển số dành cho phương tiện thông dụng tại Việt Nam. Cùng với đó là góc máy quay và thời gian, bối cảnh thay đổi đa dạng để sát với điều kiện thực tế tại các điểm đặt máy quay giám sát giao thông. Thiết bị biên được sử dụng trong luận văn có thể dễ dàng mua với giá cả phải chăng, phù hợp với việc triển khai mở rộng số lượng thiết bị.

3.2 Mô hình đề xuất

3.2.1 Tổng quan giải pháp



Hình 3.1: Mô hình tổng quan của giải pháp phát hiện biển số xe

Hình 3.1 mô tả tổng quan toàn trình mô hình phát hiện biển số xe đề xuất. Có hai thành phần chính trong hệ thống. Thành phần đầu tiên là mô-đun phát hiện phương tiện có chức năng xác định vị trí và phân loại phương tiện trong khung hình. Tiếp theo, ROI của mỗi phương tiện sẽ được cắt dựa trên bounding box của mỗi xe và được đưa vào thành phần thứ hai là mô-đun phát hiện biển số. Mô-đun này sẽ thực hiện xác định vị trí biển số xe và tự chỉnh sửa góc nhìn biển số theo hướng vuông góc. Kết quả đầu ra của toàn bộ hai mô-đun là hình ảnh biển số xe đã được hiệu chỉnh góc nhìn. Khi không có xe nào được xuất hiện trong khung hình, hình ảnh sẽ được bỏ qua, và luồng xử lý lại tiếp tục nhận các hình tiếp theo. Các mô hình này sẽ được cài đặt và thực thi trên thiết bị biên đó là Jetson Nano.

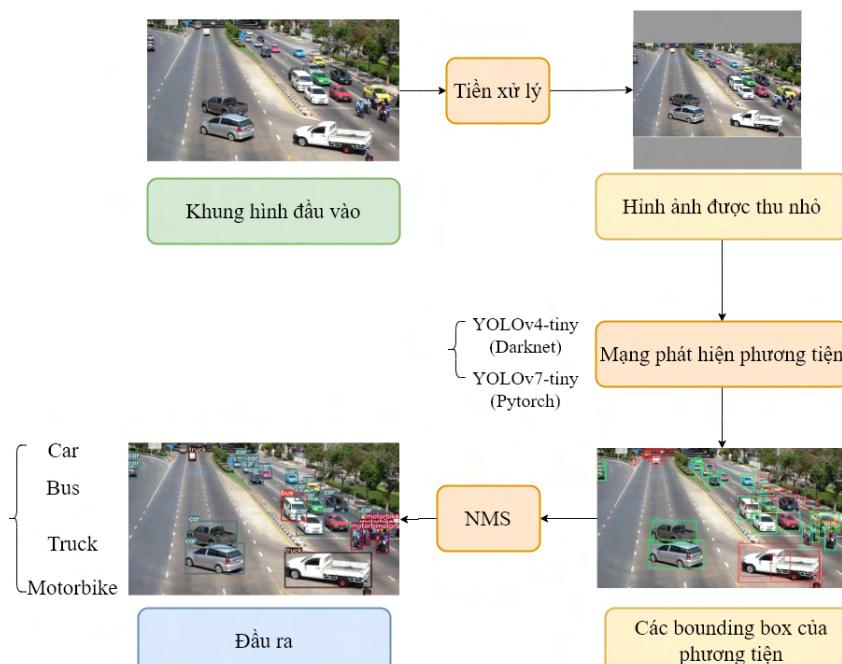
Đầu vào của các mô hình bao gồm:

- Hình ảnh của phương tiện giao thông.
- Luồng các khung hình chạy liên tục từ camera hoặc đoạn video.

Đầu ra của mô hình đề xuất bao gồm:

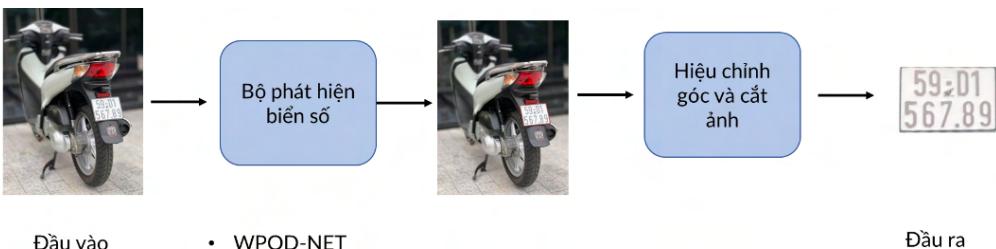
- Bounding box có chứa biển số xe.
- Hình ảnh biển số được hiệu chỉnh theo hướng chính diện với phương nhìn.

Mô-đun phát hiện phương tiện có luồng xử lý như Hình 3.2, đầu vào sẽ là các khung hình từ camera gắn vào thiết bị biên, đầu ra của mô-đun là bounding box và loại phương tiện. Đề tài lựa chọn bốn loại xe chính thường xuyên tham gia giao thông đó là xe ô tô con, xe máy, xe buýt, và xe tải để phân loại. Hai bộ phát hiện xe được sử dụng đó là phiên bản thu gọn của YOLOv4 và YOLOv7, phiên bản tiny này có cấu trúc gọn, nhẹ, phù hợp với triển khai trên thiết bị biên có cấu hình thấp.



Hình 3.2: Mô-đun phát hiện phương tiện tham gia giao thông

Hình ảnh đầu ra của mô-đun phát hiện xe được xử lý để cắt lấy vùng chỉ chứa xe, đây cũng là hình ảnh đầu vào của mô-đun thứ hai, phát hiện biển số xe. Tại đây, mô hình WPOD được sử dụng để nhận nhận diện biển số xe và chuyển hình ảnh về góc nhìn chính diện. Để phù hợp với các loại biển số tại Việt Nam, mô hình WPOD đã được chỉnh kiến trúc để có thể phân biệt được cả hai dạng biển số một dòng và hai dòng. Luồng xử lý của mô-đun thứ hai được trình bày như trong Hình 3.2.



Hình 3.3: Mô-đun phát hiện biển số xe

3.2.2 Mô-đun phát hiện phương tiện tham gia giao thông

Trong phần trước đã trình bày về tổng quan luồng xử lý của bài toán, phần này sẽ đi sâu hơn về những kỹ thuật được áp dụng trên mô-đun phát hiện phương tiện.

Mô-đun phát hiện phương tiện có đầu vào là khung hình và phát hiện các phương tiện trong khung hình đó. Hình 3.2 mô tả luồng xử lý của mô-đun phát hiện xe. Trước khi được đưa vào bộ phát hiện một giai đoạn, hình ảnh đầu vào sẽ được tiền xử lý để điều chỉnh kích thước ảnh đầu vào về kích thước 416×416 thay vì kích thước mặc định là 640×640 . Điều này giúp giảm áp lực tính toán và giảm độ trễ khi mô hình chạy trên thiết bị biên. Sau khi đã xác định được các bounding box, kết quả sẽ được lọc bớt các bounding box bằng NMS và giữ lại các kết quả tốt nhất.

Để thực hiện bài phát hiện vật đối với các loại xe trên đường phố, đề tài sử dụng hai mạng YOLOv4-tiny và YOLOv7-tiny trên tương ứng hai framework học sâu phổ biến là Darknet và PyTorch. Qua đó, ta có thể đánh giá được khả năng của thiết bị biên với từng loại mô hình cũng như các framework khác nhau.

Đầu ra của bounding box sẽ bao gồm tọa độ bốn điểm x_{min} , y_{min} , x_{max} , y_{max} , lớp của vật thể và độ tin cậy của dự đoán $confidence$.

Mặc dù hai phiên bản YOLOv4-tiny và YOLOv7-tiny đã được thiết kế với mô hình nhỏ gọn, chúng vẫn không thể đạt được tốc độ xử lý thời gian thực khi triển khai trên Jetson Nano. Nguyên nhân chính là do tài nguyên phần cứng hạn chế của thiết bị và mô hình cũng chưa được tối ưu để cho thiết bị này. Do vậy, sau khi huấn luyện, mô hình cần phải chuyển sang định dạng TensorRT để có thể tối ưu cho phần cứng của NVIDIA.

a, YOLOv4-tiny

Luồng thực hiện biến đổi mạng YOLOv4-tiny để chạy trên thiết bị biên được trình bày như Hình 3.4. Có thể thấy, không thể biến đổi một cách trực tiếp từ Darknet sang TensorRT mà phải thông qua bước biến đổi sang định dạng trung gian ONNX rồi chuyển sang định dạng TensorRT. Trong quá trình biến đổi, mạng

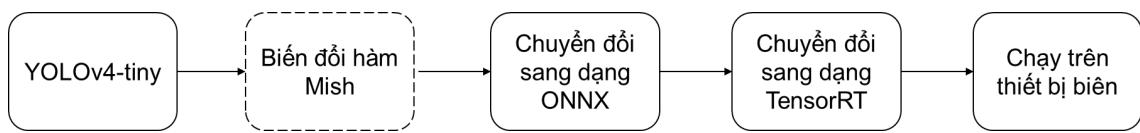
YOLOv4-tiny sử dụng hàm kích hoạt *Mish*, mà hàm này không có sẵn trong thư viện của ONNX hay TensorRT. Do đó, để có thể chuyển đổi từ mạng YOLOv4-tiny, phải thực hiện viết chương trình để chuyển từ hàm *Mish* sang hàm khác tương đương có sẵn trong ONNX và TensorRT. Từ công thức định nghĩa của hàm *Mish* như trong phương trình 3.1.

$$f(x) = x \star \tanh(\text{softplus}(x)) \quad (3.1)$$

Trong đó, hàm *softplus* được định nghĩa như sau

$$\text{softplus}(x) = \ln(1 + e^x) \quad (3.2)$$

Từ phương trình 3.1 và 3.2, ta nhận thấy có thể thay thế hàm *Mish* bằng các hàm tương đương được hỗ trợ bởi cả ONNX và TensorRT đó là hàm *Softplus*, *Tanh*, *Mul*. Để tài thực hiện viết chương trình con để chuyển đổi từ YOLOv4-tiny sang dạng ONNX trong đó, thay thế hàm *Mish* bằng *Softplus*, *Tanh*, *Mul*.



Hình 3.4: Luồng biến đổi từ YOLOv4-tiny sang TensorRT

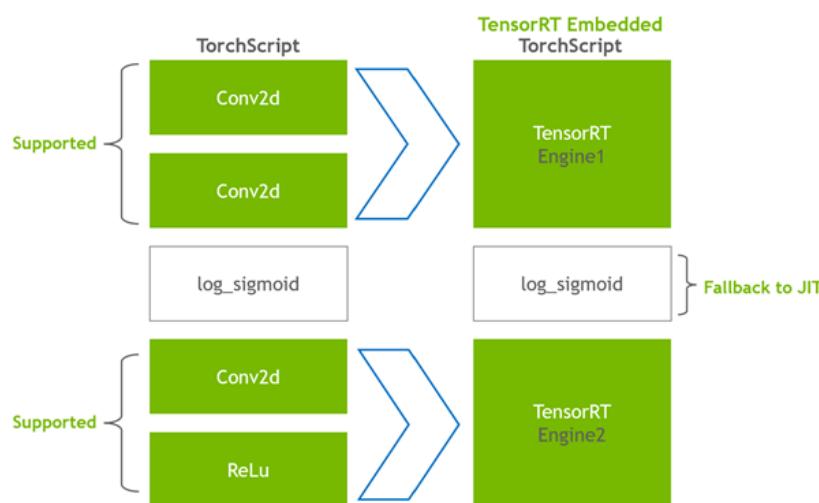
b, YOLOv7-tiny

YOLOv7-tiny là bản rút gọn của YOLOv7 được tối ưu hóa cho việc sử dụng trên GPU của thiết bị biên. So với các phiên bản khác, phiên bản YOLOv7-tiny sử dụng hàm leaky ReLU làm hàm kích hoạt, trong khi các mạng YOLOv7 khác sử dụng SiLU làm hàm kích hoạt. Mạng YOLOv7-tiny được thiết kế và huấn luyện trên framework PyTorch. Tương tự như YOLOv4-tiny, mặc dù đã được thu gọn về mặt kích thước mạng nơ-ron, YOLOv7-tiny vẫn cần phải chuyển đổi về dạng TensorRT để có thể tối đa hóa tốc độ xử lý trên thiết bị Jetson Nano.

Để chuyển đổi từ mô hình YOLOv7-tiny từ PyTorch sang TensorRT gồm ba bước:

- Bước 1-Giảm độ phức tạp của mô-đun TorchScript: đơn giản hóa các toán tử phức tạp hoặc các hàm toán ghép ở trong PyTorch sang các toán tử đơn giản hơn, từ đó, có thể chuyển đổi trực tiếp sang TensorRT ở bước sau. Điều quan trọng là việc giảm mức độ phức tạp này không làm ảnh hưởng đến chức năng của mạng học sâu.

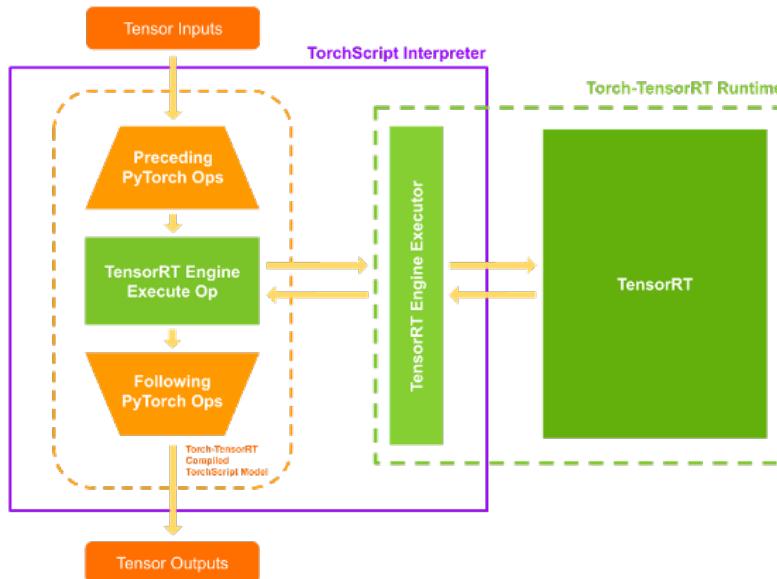
- Bước 2-Biến đổi trực tiếp: trong pha này, các thành phần con trong cấu trúc mạng học sâu tương thích với TensorRT được xác định tự động và chuyển chúng thành các phép tính TensorRT. Trong đó, các node với giá trị tĩnh được đánh giá và ánh xạ thành các hằng số. Các node mô tả tính toán trên tensor được chuyển đổi thành một hoặc nhiều lớp TensorRT. Các node còn lại không biến đổi được trong TorchScript sẽ tạo thành một đồ thị lai và được trả về dưới dạng một mô-đun TorchScript tiêu chuẩn. Mô-đun đã biến đổi sẽ được trả lại với TensorRT engine được nhúng vào đó, điều này có nghĩa là toàn bộ mô hình, bao gồm mã PyTorch, trọng số mô hình và các engine TensorRT, đều được đóng gói thành một thể thống nhất như Hình 3.5.



Hình 3.5: lớp Conv2d được chuyển đổi thành một engine TensorRT, trong khi log_sigmoid sẽ sử dụng TorchScript JIT để thực thi [64]

- Bước 3-Thực thi: Khi thực thi mô-đun đã được biến đổi, trình thông dịch TorchScript sẽ gọi engine TensorRT và truyền các dữ liệu đầu vào qua TensorRT. Engine này thực thi và đẩy kết quả trở lại trình thông dịch như một mô-đun TorchScript bình thường như Hình 3.6.

Trong quá trình biến đổi, mô hình được tối ưu hóa sang định dạng số nguyên INT8 với hai kỹ thuật chính: post-training quantization và quantization-aware training. Kỹ thuật post-training quantization là một phương pháp tối ưu hóa mô hình máy học sâu khi nó đã được huấn luyện. Nó giảm độ chính xác của các tham số trong mô hình từ dạng số thực *FP32* sang dạng số nguyên *INT8*, để giảm kích thước của mô hình, giúp tăng tốc độ thực thi trên thiết bị có tài nguyên hạn chế. Kỹ thuật quantization-aware training giúp ánh xạ các toán tử/hàm liên quan đến lượng tự hóa từ PyTorch sang TensorRT.



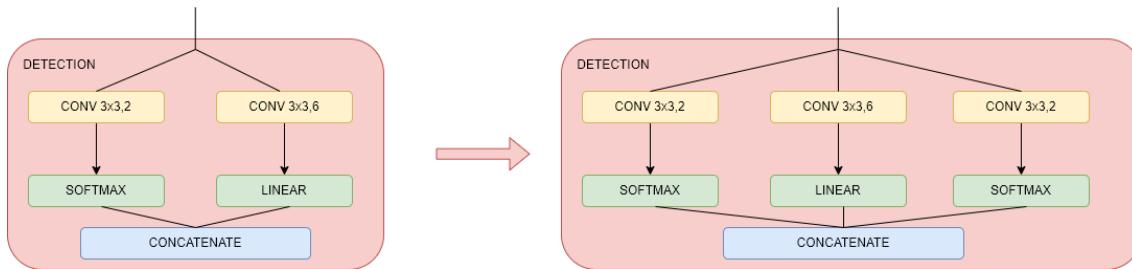
Hình 3.6: Thực thi các toán tử PyTorch và TensorRT [64]

3.2.3 Mô-đun phát hiện biển số xe

Mạng WPOD được sử dụng làm mô-đun phát hiện biển số, để tài giữ lại tất cả các thành phần cốt lõi và chỉnh sửa đầu ra của mạng để phù hợp tốt hơn với bộ dữ liệu thực nghiệm của chúng tôi. Ban đầu, mô hình được huấn luyện trên các hình ảnh chứa biển số xe dài, chỉ có một dòng, vì vậy các tác giả chỉ sử dụng một kích thước đầu ra duy nhất để giải mã vùng biển số xe. Nói cách khác, mô hình gốc không thể phân biệt xem một biển số xe có tỷ lệ khía cạnh giống hình vuông hay tỷ lệ khía cạnh dài hình chữ nhật. Do đó, tất cả các biển số vuông, hai dòng sẽ bị thay đổi tỷ lệ khía cạnh sai, làm cho chúng trông bị méo và không tự nhiên. Mặc dù tỷ lệ khía cạnh có thể được suy ra bằng cách tính toán chiều rộng và chiều cao của biển số từ bốn góc đầu ra, phương pháp này không ổn định do hình dạng đa giác của đầu ra và có hiệu suất khá kém trong các thử nghiệm.

Để giải quyết vấn đề này, trong đề tài đã thử nghiệm với hai phương pháp. Cách thứ nhất đó là dựa vào tỷ lệ giữa hai cạnh của biển số đầu ra để kết luận đó là biển số loại vuông hay hình chữ nhật. Nếu tỷ lệ giữa hai cạnh này lớn hơn 1,7 thì khẳng định đó là hình chữ nhật và ngược lại. Cách thứ hai đó là sửa đổi kiến trúc gốc bằng cách thêm hai kẽm vào bản đồ đặc trưng đầu ra của mạng, đại diện cho xác suất của biển số xe trong hình ảnh đó là một dòng hoặc hai dòng. Hình 3.7 thể hiện kiến trúc mạng WPOD trước và sau khi thay đổi. Kích thước của đầu vào của lớp này là 3×3 , đầu ra là hai ma trận hai phần tử đại diện cho loại biển số, hàm kích hoạt được sử dụng là *Softmax*. Đề tài đã thử nghiệm cả hai phương pháp để xem liệu

việc sửa đổi kiến trúc có giúp phân loại loại biển số xe tốt hơn so với phương pháp dựa vào tỷ lệ chiều dài cạnh của đa giác phát hiện được hay không.



Hình 3.7: Kiến trúc của khôi phát hiện trong mạng WPOD trước và sau khi được sửa đổi

Mạng được huấn luyện trên framework TensorFlow do đó, nó được tối ưu hóa thông qua chuyển đổi từ TensorFlow sang TensorRT. Tương tự như với Darknet, đầu tiên mô hình được biến đổi qua dạng ONNX sau đó, chuyển từ ONNX sang TRT.

3.3 Thực nghiệm và kết quả

Sau khi đã trình bày các luồng xử lý, mô hình và phương pháp biến đổi ở mục trước, tại mục này sẽ trình bày quá trình thực nghiệm và các kết quả đạt được.

3.3.1 Thiết lập môi trường thực nghiệm

Với mô-đun phát hiện phương tiện với tập dữ liệu được đề xuất, các tham số được cài đặt như sau: kích thước đầu vào là 416×416 , cả hai mạng đều sử dụng một GPU để huấn luyện với kích thước tập mẫu (batch size) là 64. Với từng mạng YOLOv4-tiny, đề tài thực hiện một số thay đổi với các siêu tham số trong quá trình huấn luyện như sau: tham số *filter* là 27, tham số *maxbatches* là 8000, và *step* là 6400, 7200. Còn với YOLOv7-tiny sẽ giữ nguyên siêu tham số mặc định.

Với mô-đun phát hiện biển số xe, mạng WPOD được tiến hành huấn luyện với kích thước tập mẫu là 32, số vòng lặp là 300000, thuật toán Adam được sử dụng để tối ưu, tốc độ học (learning rate) được cài đặt là 0,01.

Nghiên cứu được thực nghiệm trên thiết bị biên là NVIDIA Jetson Nano 4GB và môi trường Google Colab với GPU NVIDIA T4. Đầu vào của các mô hình là dữ liệu dạng ảnh đã được gán nhãn. Từ đó, có thể dễ dàng đánh giá độ chính xác của các mô hình được sử dụng. Để chứng minh năng lực của thiết bị Jetson Nano hoàn toàn có thể đáp ứng tương đương với phần cứng của máy chủ được trang bị GPU, đề tài đã thực hiện chạy mô hình trên cả nền tảng đám mây Google Colab. Đây là một sản phẩm từ Google Research, nó cho phép thực thi mã lệnh Python thông qua trình duyệt. GPU được sử dụng trên Colab trong thí nghiệm là NVIDIA Tesla T4.

3.3.2 Bộ dữ liệu

a, Bộ dữ liệu huấn luyện mô hình phát hiện phương tiện giao thông

Bộ dữ liệu dùng để huấn luyện và kiểm tra mô hình phát hiện xe sử dụng trong đề tài được kết hợp từ tập dữ liệu phương tiện Việt Nam (VVD - Vietnamese Vehicle Dataset) [65] và tập dữ liệu phương tiện OpenImage (VOID - Vehicle Open Image dataset) [66]. VVD là tập dữ liệu các phương tiện phổ biến được thu thập tại thành phố Hồ Chí Minh vào các khoảng thời gian khác nhau. Các bức ảnh, như trong Hình 3.8 được trích xuất từ dữ liệu camera ở các vị trí khác nhau, thời điểm khác nhau, thời tiết khác nhau để đảm bảo tính đa dạng của bộ dữ liệu. Tất cả các hình ảnh đều được gán nhãn với các bounding box chứa tương ứng với một trong bốn lớp: xe máy, ô tô, xe buýt hoặc xe tải. Tổng cộng có 15391 hình ảnh, được thu thập từ 25 camera giao thông. Tuy nhiên, các hình ảnh trong bộ dữ liệu VVD chủ yếu là hình ảnh từ các góc nhìn ở trên cao, nhiều vật thể ở trong cùng một khung hình. Vì vậy, để bổ sung tính đa dạng cho bộ dữ liệu với các hình ảnh ở khoảng cách gần hơn, góc nhìn đa dạng hơn, và có ít đối tượng trong ảnh, đề tài đã sử dụng thêm bộ dữ liệu VOID.



Hình 3.8: Hình ảnh từ tập dữ liệu VVD

VOID là một tập dữ liệu gồm 627 xe được lấy từ thư viện thị giác máy tính mã nguồn mở OpenImages. Tập dữ liệu này chứa các loại xe thuộc năm lớp: xe máy, ô tô, xe buýt, xe tải và xe cứu thương, với 1194 chú thích khung chứa (trung bình 1,9 cho mỗi hình ảnh) và tỷ lệ trung vị của hình ảnh là 1024×571 . Hình 3.9 lấy một số

hình ảnh trong tập dữ liệu VOID thể hiện sự đa dạng về góc chụp của các phương tiện và số lượng phương tiện trong một mẫu.



Hình 3.9: Hình ảnh từ tập dữ liệu VOID

Để kết hợp hai tập dữ liệu, đề tài xem tất cả các hình ảnh xe cứu thương trong tập dữ liệu thứ hai là ô tô. Điều đó có nghĩa là tập dữ liệu kết hợp chỉ có bốn lớp: xe máy, ô tô, xe buýt và xe tải. Sau đó, mỗi tập dữ liệu được chia thành ba tập con là tập huấn luyện, tập xác nhận và tập kiểm tra theo tỷ lệ tương ứng xấp xỉ là 8 : 1 : 1 như trong Bảng 3.1

Bảng 3.1: Bộ dữ liệu phương tiện giao thông

	Huấn luyện	Xác nhận	Kiểm tra
Vietnamese Vehicle Dataset	12221 ảnh	1132 ảnh	2038 ảnh
OpenImages Vehicle Dataset	439 ảnh	125 ảnh	63 ảnh
Tổng	12660 ảnh	1257 ảnh	2101 ảnh

b, Bộ dữ liệu huấn luyện mô hình biến số xe

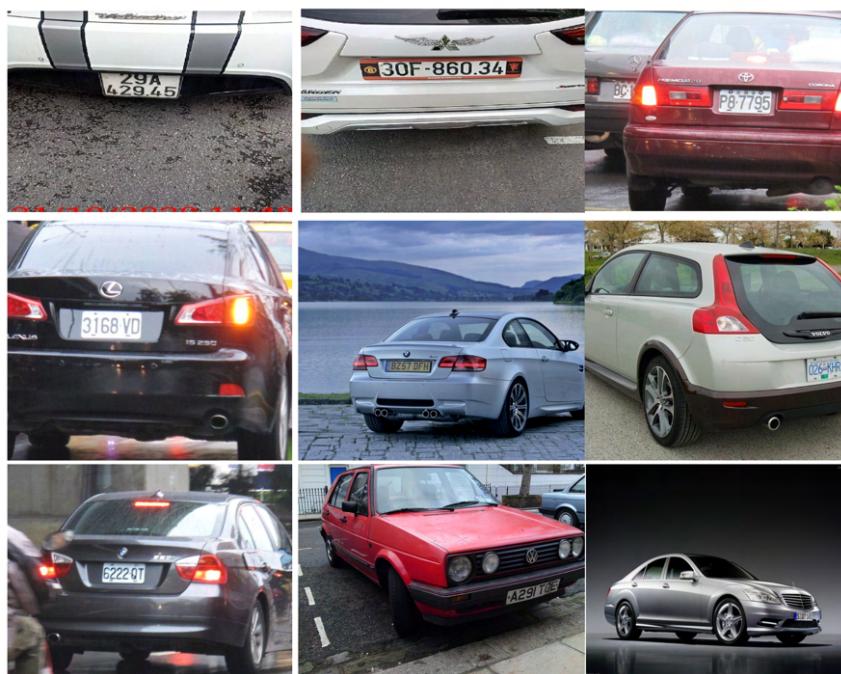
Tập dữ liệu phát hiện biển số có kích thước nhỏ hơn nhiều so với tập dữ liệu phát hiện xe, với tổng cộng 421 hình ảnh, và là sự kết hợp của ba nguồn: 189 hình ảnh từ tập dữ liệu xe ô tô [67], 51 hình ảnh từ tập dữ liệu AOLP [68], và 181 hình ảnh từ tập dữ liệu CarTGMT [69]. Hình ảnh được lấy mẫu từ ba nguồn để đảm bảo rằng biển số xe xuất hiện trong các nền, góc nhìn, quốc gia và điều kiện khác nhau. Cả biển số dài loại một dòng và biển số vuông loại hai dòng đều có mặt trong bộ dữ liệu. Trong đó, chỉ có tập AOLP là đã được gán nhãn cho biển số, tuy nhiên số

CHƯƠNG 3. BÀI TOÁN PHÁT HIỆN BIỂN SỐ XE TRÊN THIẾT BỊ BIÊN

lượng mẫu như vậy chưa đa dạng về các loại biển số và góc chụp của biển số. Do đó, nhóm nghiên cứu đã bổ sung thêm các bộ dữ liệu tự gán nhãn, tất cả các hình ảnh trong tập dữ liệu CarTGMT và 189 hình ảnh trong tập dữ liệu xe đều được gán nhãn thủ công trong quá trình thực hiện đề tài, đây là một đóng góp của đề tài trong xây dựng bộ dữ liệu biển số xe. Bảng 3.2 cho thấy cách chia dữ liệu biển số xe. Hình 3.10 thể hiện bộ dữ liệu biển số xe trong đề tài.

Bảng 3.2: Bộ dữ liệu biển số xe

	Huấn luyện	Xác nhận	Kiểm thử
Cars Dataset	149 ảnh	20 ảnh	20 ảnh
AOLP Dataset	45 ảnh	3 ảnh	3 ảnh
CarTGMT Dataset	100 ảnh	40 ảnh	41 ảnh
Tổng	294 ảnh	63 ảnh	64 ảnh



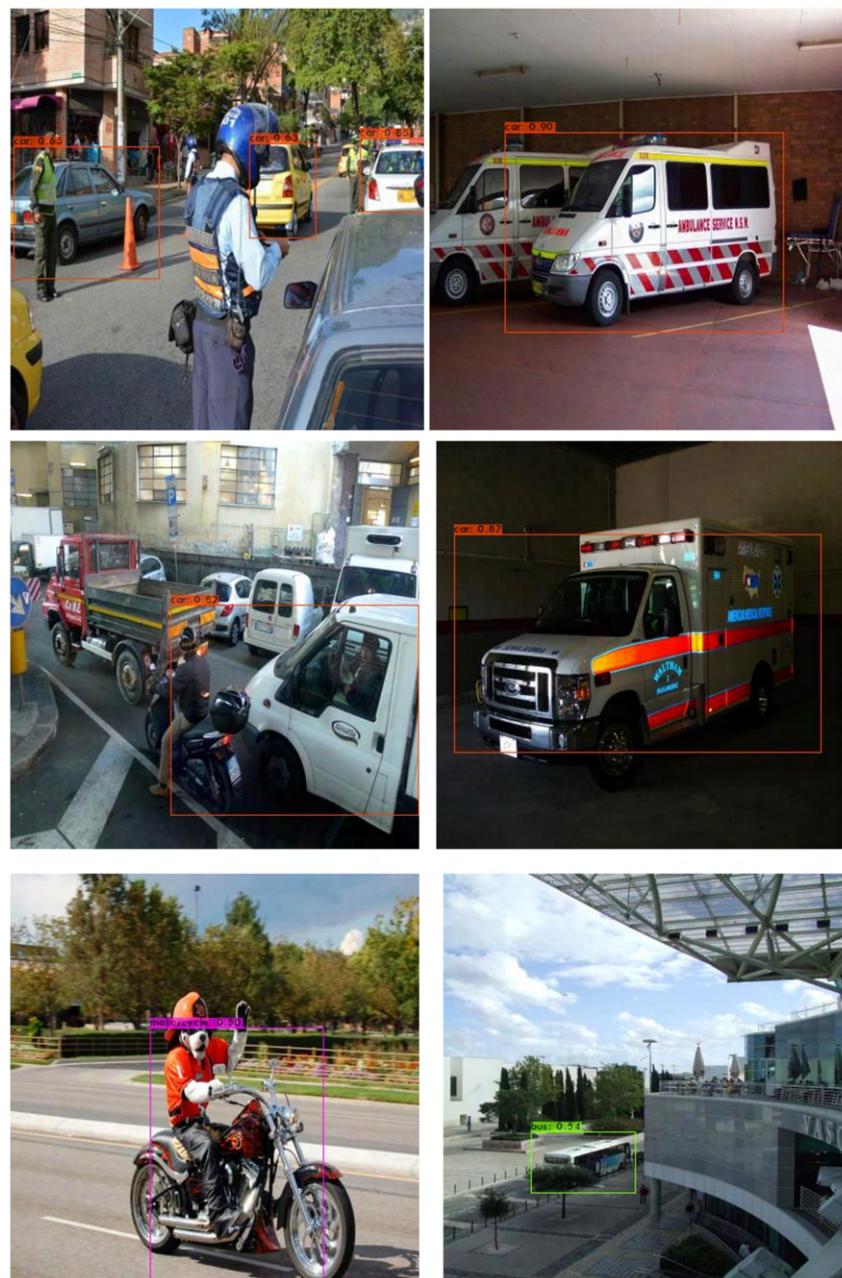
Hình 3.10: Hình ảnh từ tập dữ liệu biển số xe

3.3.3 Kết quả thực nghiệm

a, Mô-đun phát hiện phương tiện

Để kiểm tra độ chính xác của mô-đun phát hiện phương tiện, nghiên cứu đã sử dụng hình ảnh với những môi trường ánh sáng khác nhau, tại các thời điểm khác nhau trong ngày, đặc biệt, đối với hình ảnh từ trích xuất từ đường phố Việt Nam, địa điểm của hình ảnh hoàn toàn không nằm trong tập huấn luyện để đảm bảo khách quan. Hình 3.11 mô tả kết quả phát hiện phương tiện với tập OVD. Có thể thấy, dù là điều kiện thiếu sáng hay đầy đủ ánh sáng thì mô-đun đều cho kết quả tương đối

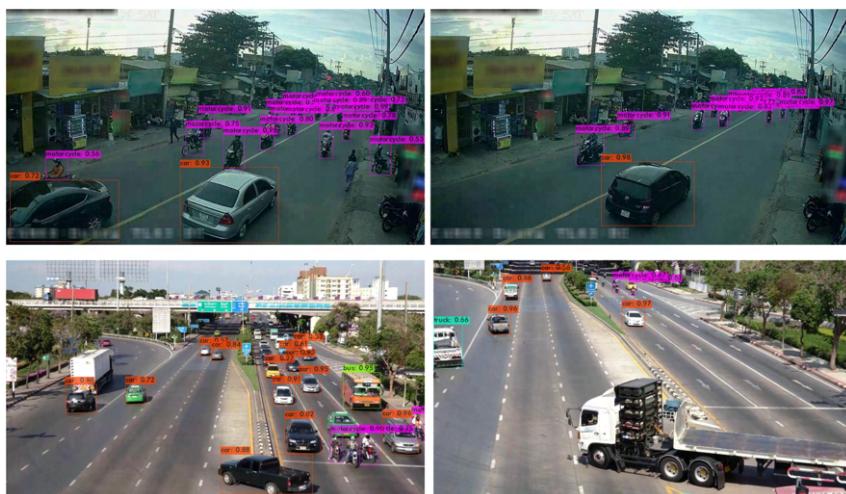
chính xác.



Hình 3.11: Kết quả phát hiện phương tiện trên tập OVD với các điều kiện ánh sáng khác nhau

Hình 3.12 thể hiện kết quả khi sử dụng hình ảnh đầu vào từ tập VVD. Trong điều kiện có nhiều phương tiện xuất hiện cùng trong một khung hình, bộ phát hiện có thể phát hiện được các lớp như đã được huấn luyện. Bên cạnh đó, với điều kiện ánh sáng thay đổi giữa sáng và chiều tối, bộ phát hiện vẫn cho kết quả nhận diện tốt.

Kết quả của hai mô hình phát hiện đối tượng trong nhiệm vụ phát hiện xe được thể hiện trong Bảng 3.3. YOLOv7-tiny cho kết quả tốt nhất về cả AP và tốc độ với



Hình 3.12: Kết quả phát hiện phương tiện trên tập VVD với các địa điểm và thời gian khác nhau

75,500% AP@0,5 và tốc độ xử lý 42 FPS trên Jetson Nano. YOLOv4-tiny có số lượng tham số lớn hơn so với YOLOv7-tiny, cùng với đó độ chính xác và tốc độ xử lý cũng thấp hơn khá nhiều, với 71,600%AP@0,5, tốc độ xử lý 25 FPS.

Bảng 3.3: Kết quả của mô hình phát hiện phương tiện trên nền tảng Colab và Jetson Nano

Mô hình	Framework	Parameters	Google Colab			Jetson Nano		
			AP@0,5	AP@0,5:0,95	FPS	AP@0,5	AP@0,5:0,95	FPS
YOLOv7-tiny	PyTorch	6,0M	0,747	0,454	143	0,755	0,460	42
YOLOv4-tiny	Darknet	6,1M	0,745	0,396	151	0,716	0,386	25

Kết quả trong Bảng 3.3 cho thấy với mô hình YOLOv7-tiny chạy trên thiết bị Jetson Nano tương đương, thậm chí có phần tốt hơn về độ chính xác.Thêm vào đó, tốc độ 42 FPS hoàn toàn đáp ứng cho việc xử lý theo thời gian thực, không làm cho hình ảnh bị ngắt quãng, đạt được mục tiêu đề ra ban đầu. Ngay kể cả với mô hình YOLOv4-tiny trên framework Darknet cũng cho thấy khả năng xử lý chính xác gần bằng với khi chạy trên Google Colab, trong khi tốc độ lại đáp ứng lớn hơn 24 FPS.

b, Mô-đun phát hiện biển số xe

Kết quả phát hiện biển số của mô hình được thể hiện trong Bảng 3.4.

Bảng 3.4: Kết quả của mô hình phát hiện biển số

Model	Framework	Parameters	Google Colab		Jetson Nano	
			mIoU	FPS	mIoU	FPS
WPOD-NET	TensorFlow	1,6M	0,762	260	0,758	204

Kết quả thí nghiệm cho thấy mạng WPOD có hiệu năng cao về kích thước, tốc độ và IoU. Với chỉ khoảng 1,6 triệu tham số, mô hình hoạt động tốt ở mức khoảng

0,760 mIoU trong khi đạt được tốc độ rất cao trên cả Google Colab và Jetson Nano. FPS trên Jetson Nano đạt mức 204 FPS chậm hơn không đáng kể so với 260 FPS trên Google Colab, điều này rất ấn tượng khi xét đến phần cứng thấp của thiết bị. Kết quả lại cho thấy rằng TensorRT có thể mang lại cho mô hình một sự tăng tốc đáng ngạc nhiên với ít hoặc không có sự đánh đổi về độ chính xác. Đối với việc phân loại loại biển số, Bảng 3.5 thông kê kết quả phát hiện biển số với hai phương pháp: sử dụng quy tắc cố định và sửa đổi kiến trúc mô hình. Ngưỡng cho tỷ lệ của cạnh dài nhất và cạnh ngắn thứ hai của vùng biển số được phát hiện được đặt là 1,7. Nếu tỷ lệ đó vượt quá ngưỡng đã chọn, biển số xe sẽ được phân loại là vuông, hai dòng. Ngược lại, nó sẽ được phân loại là biển số dài, một dòng. Các thí nghiệm cho thấy rằng, với phương pháp sửa đổi kiến trúc mạng WPOD, mô hình có thể nhận biết loại biển số trình bày trong tập dữ liệu tốt hơn, đồng thời cho phép chỉnh sửa tốt hơn bằng cách sử dụng biến đổi phối cảnh.

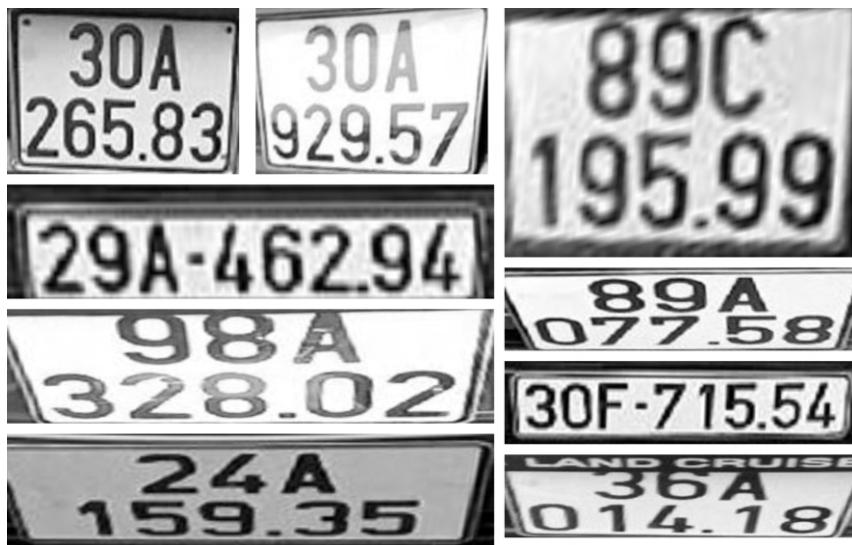
Bảng 3.5: Kết quả phân loại biển số một dòng và hai dòng

	Phương pháp thay đổi kiến trúc mạng	Phương pháp sử dụng tỷ lệ
TP	55	37
TP+FP	61	61
Precision	90,100%	60,600%

Hình 3.13 cho thấy các kết quả biển số được nhận diện và biến đổi góc nhìn sau khi sử dụng mạng WPOD. Có thể thấy, các biển số thuộc loại hai dòng hay một dòng đều cho kết quả phát hiện tốt, được xoay sang góc chính diện, dễ nhìn. Một số kết quả có phần bị méo, chưa vuông góc, tuy nhiên vẫn đảm bảo nhìn rõ các ký tự trên biển số. Điều này sẽ hỗ trợ cho hệ thống nhận trích xuất thông tin biển số ở các pha sau.

3.4 Kết luận

Chương 3 đã nghiên cứu và thử nghiệm các mô hình học sâu cho các nhiệm vụ liên quan đến giao thông. Hai trong số đó là cho nhiệm vụ phát hiện xe, và còn lại là cho nhiệm vụ phát hiện biển số. Các mô hình YOLOv4-tiny và YOLOv7-tiny được sử dụng trực tiếp, trong khi các mô hình WPOD-NET được tùy biến để phù hợp hơn với bài toán và đảm bảo tốc độ tính toán. Đề tài đã triển khai, đánh giá hiệu năng và tốc độ của tất cả các mô hình trên tập dữ liệu tùy chỉnh của cho mỗi nhiệm vụ. Kết quả cho thấy rằng, đối với một GPU trung bình như Tesla T4, tốc độ suy luận rất cao, và độ chính xác khá tốt. Khi triển khai trên thiết bị biên, cụ thể là Jetson Nano, tốc độ bị giảm khá nhiều, nhưng vẫn có thể đáp ứng chạy thời gian thực, và độ chính xác gần như không thay đổi, thậm chí tốt hơn trong một số



Hình 3.13: Kết quả phát hiện biển số bằng WPOD

trường hợp. Có thể thấy rằng Jetson Nano có khả năng triển khai các mô hình học sâu và có thể được sử dụng hiệu quả để giải quyết nhiều vấn đề thị giác máy tính. Một hướng cải tiến có thể trong tương lai là kiểm tra thêm nhiều mô hình khác và cố gắng tùy chỉnh chúng để giảm số lượng tham số và chi phí tính toán.

Các phương pháp thực hiện và kết quả trong chương này đã được trình bày tại hội nghị AICI2023 và đã được đưa vào tập san của hội nghị [60].

CHƯƠNG 4. BÀI TOÁN TÁI ĐỊNH DANH NGƯỜI

Chương 3 đã trình bày về bài toán phát hiện biển số chạy độc lập trên thiết bị biên Jetson Nano. Thực tế, không phải lúc nào cũng có thể phụ thuộc hoàn toàn vào các thiết bị này, đặc biệt là đối với những bài toán cần chạy nhiều mô hình phức tạp hoặc các bài toán cần phải phối hợp luồng thông tin từ nhiều thiết bị khác nhau. Chương này, đề tài sẽ đề xuất giải pháp đối với bài toán tái định danh người theo mô hình kết hợp giữa phần xử lý tại biên và tại máy chủ. Mục tiêu của bài toán là có thể xác định cùng một người trong các luồng video từ nhiều camera khác nhau với tốc độ xử lý lớn hơn 24 FPS và độ chính xác trên 70%. Chương 4 có cấu trúc như sau: mục đầu tiên sẽ trình bày về các nghiên cứu liên quan trên thiết bị biên. Sau đó, giải pháp thiết kế được đề xuất, mô tả thực nghiệm, và cuối cùng là kết luận.

Kết quả của nghiên cứu cho bài toán tái định danh này đã được tác giả và nhóm nghiên cứu gửi lên tạp chí Multimedia Tools and Applications và đang trong quá trình phản biện.

4.1 Nghiên cứu liên quan

Mục tiêu chính của bài toán tái định danh đó là xác định chính xác cùng một người trong luồng video tại nhiều thời điểm khác nhau, vị trí khác nhau từ nhiều camera khác nhau. Hệ thống này có thể được sử dụng trong các hệ thống giám sát, theo dõi một đối tượng đặc biệt khi di chuyển qua nhiều khu vực khác nhau. Nhiều nghiên cứu đã sử dụng các đặc điểm về ngoại hình để nhận diện một người. Phương pháp này đối mặt với các thách thức như biến đổi về ánh sáng, góc quan sát [70], bối cảnh và trang phục mà người đó đang mặc [71].

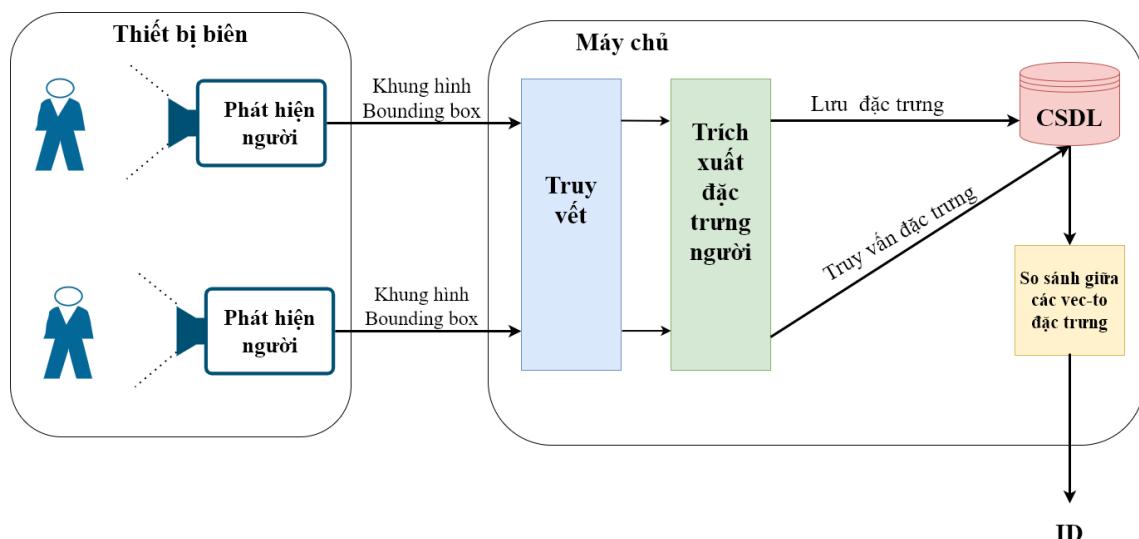
Đã có những thành tựu đáng chú ý gần đây về các phương pháp person re-ID cho các thiết bị phần cứng. Ví dụ, một phương pháp sử dụng độ chính xác hỗn hợp bằng cách sử dụng các mô hình nhẹ (các mô hình như ResNet50 và MobileNet-v2) đã được triển khai trên một hệ thống nhúng NVIDIA AGX Xavier, để tạo thành một phương pháp tái định danh từ đầu đến cuối [72]. Một nghiên cứu khác đã xây dựng giải pháp kết hợp của các thiết bị nhúng, máy chủ cận biên và máy chủ đám mây dựa trên một mô hình segmented binarized Resnet [73]. Cuối cùng, phương pháp liên quan đến việc học các đặc điểm tương đồng giữa các cặp hình ảnh con người khác nhau [74]. Các nghiên cứu trên mới tập trung vào giải pháp chứ chưa tập trung vào việc áp dụng thực tế với các thiết bị có tài nguyên hạn chế. Do vậy,

đề tài sẽ tiến hành giải quyết bài toán tái định danh người áp dụng mô hình học sâu trên thiết bị biên cấu hình thấp.

4.2 Mô hình đề xuất

4.2.1 Tổng quan mô hình

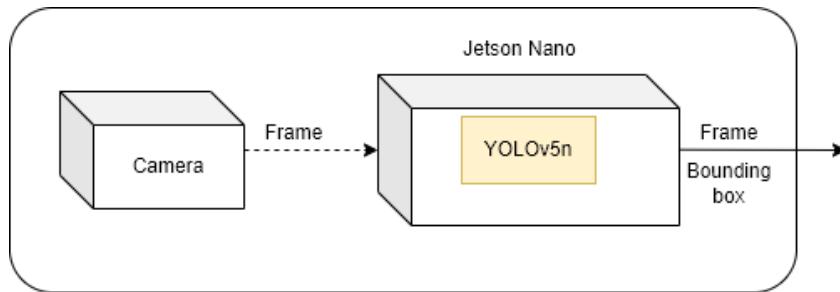
Hình 4.1 thể hiện tổng quan về một hệ thống giám sát người toàn trình hoàn chỉnh trong một căn phòng hoặc tòa nhà. Hệ thống này có các thiết bị ngoại vi như camera để chụp khung hình, một máy tính nhúng nhỏ gọn, Jetson Nano đảm nhiệm tính toán tại biên. Nó nhận luồng từ camera kết nối và thực hiện tất cả các tính toán các tác vụ liên quan đến AI, sau đó gửi thông tin được xử lý đến máy chủ để lưu trữ. Có hai thiết bị Jetson Nano được đặt ở cửa vào và cửa ra. Hệ thống này cũng có một máy chủ để lưu trữ dữ liệu được gửi từ Jetson Nano và xử lý các yêu cầu từ ứng dụng người dùng. Máy chủ chạy các thuật toán theo dõi và trích xuất đặc trưng. Đặc trưng của từng người đi vào được phát hiện bởi thiết bị ở cửa vào, sau đó sẽ được lưu trữ trong cơ sở dữ liệu. Các đặc trưng của những người đi ra cũng sẽ được thu thập bởi thiết bị ở cửa ra và so sánh với những đặc trưng có sẵn trong cơ sở dữ liệu bằng cách ước tính khoảng cách giữa mỗi cặp vector đặc trưng. Từ đó, máy chủ có khả năng tái định danh các thực thể này miễn là họ đã được phát hiện bởi camera ở cửa vào.



Hình 4.1: Mô hình đề xuất

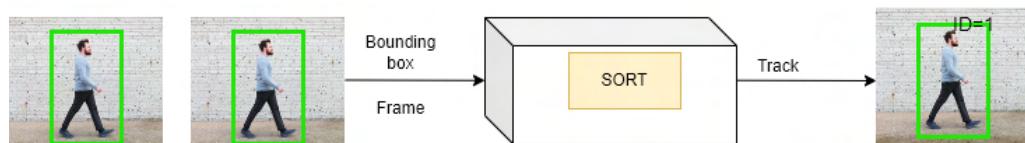
Từ tổng quan mô hình đề xuất có thể thấy, luồng xử lý cho bài toán tái định danh người bao gồm ba mô-đun nối tiếp nhau: mô-đun phát hiện người trên thiết bị biên, mô-đun truy vết và mô-đun trích xuất đặc trưng trên máy chủ. Phạm vi của đề tài chỉ tập trung vào phát triển mô-đun AI trên thiết bị biên và mô-đun truy vết trên máy chủ, còn lại các mô-đun trích xuất đặc trưng được phát triển bởi thành

viên khác trong nhóm. Jetson Nano nhận các khung hình đầu vào từ camera, sau đó mô hình phát hiện đối tượng sẽ nhận diện người trên các khung hình này và đưa ra bounding box cho mỗi người (Hình 4.2). Mạng YOLOv5 được sử dụng cho mô-đun phát hiện người bởi kích thước nhỏ gọn và độ chính xác cao.



Hình 4.2: Mô-đun phát hiện người trên thiết bị biên

Tiếp theo, người đó phải được truy vết để đảm bảo máy chủ xác định được người xuất hiện trong các khung hình liên tiếp là một. Đầu vào của mô hình truy vết là hình ảnh và bounding box của người đó. Đầu ra là danh sách các đối tượng đã được theo dõi trong một chuỗi khung hình hoặc video. Mỗi đối tượng trong danh sách này thường được biểu thị bằng một bounding box hoặc các thông tin liên quan khác như tọa độ của bounding box, vận tốc, ID duy nhất của đối tượng, và thời gian tiến triển từ khi được theo dõi. SORT giúp theo dõi các đối tượng qua các khung hình liên tiếp và cung cấp thông tin về vị trí và di chuyển của chúng theo thời gian thực (Hình 4.3).



Hình 4.3: Mô-đun truy vết người trên server

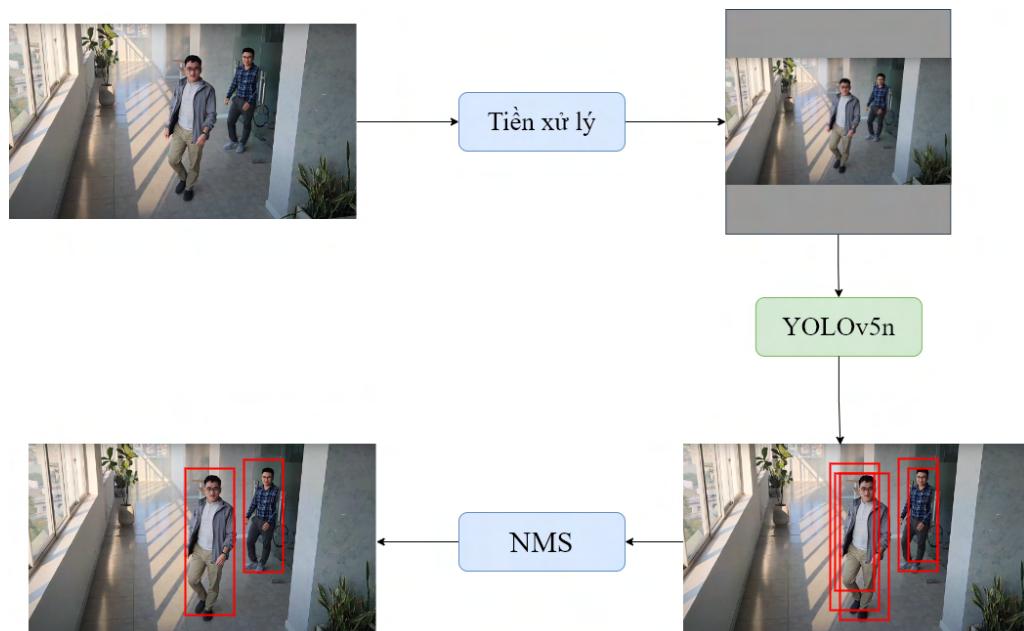
4.2.2 Mô-đun phát hiện người

Như đã đề cập ở phần trước, YOLO là bộ phát hiện một pha có tốc độ và độ chính xác cao. Trong bài toán này, mô-đun phát hiện người phải cho kết quả chính xác mà vẫn đảm bảo thời gian xử lý nhanh nhất có thể, do vậy, đề tài lựa chọn phiên bản thu gọn của YOLOv5 là YOLOv5n. YOLOv5n được thiết kế tối giản hơn bằng cách tích hợp ít lớp tích chập và anchor box hơn so với phiên bản gốc. Sự giảm thiểu này trong các thành phần dẫn đến giảm số lượng tham số và FLOPS, do đó hiệu quả hơn. Điều này đặc biệt quan trọng khi mô-đun phát hiện được sử dụng trên các thiết bị biên, nơi tài nguyên tính toán bị giới hạn. Để rút ngắn thời gian

thực hiện, nghiên cứu này sử dụng mạng YOLOv5n đã được huấn luyện trên tập dữ liệu COCO, do đó, nó có sẵn khả năng phát hiện người mà không cần phải huấn luyện lại.

Tương tự như ở bài toán phát hiện biển số xe, nếu thực hiện chạy mạng YOLOv5n với dạng mặc định trên framework PyTorch thì sẽ không thể đạt được tốc độ xử lý thời gian thực. Do đó, cần chuyển đổi mô hình thành định dạng được tối ưu hóa phù hợp với từng thiết bị, cụ thể là định dạng TensorRT cho các thiết bị Jetson Nano.

YOLOv5n có thể dễ dàng được chuyển đổi sang định dạng ONNX, sau đó chuyển sang định dạng TensorRT. Các tham số của mô hình được lượng tử hóa xuống dạng số thực 16 bit, giúp tăng tốc độ xử lý trên thiết bị biên.



Hình 4.4: Quá trình phát hiện người

Trước khi được đưa vào mạng YOLOv5n để phát hiện người, các khung hình phải đi qua bước tiền xử lý như trong Hình 4.4. Sau khi đã có được các bounding box của từng đối tượng, thuật toán NMS được sử dụng để chọn ra bounding box tốt nhất cho mỗi người.

Mỗi bounding box đầu ra sẽ bao gồm các thành phần sau đây:

- Tọa độ của các khung ở trong hình ảnh. Tọa độ này chứa thành phần đó là $xmin, ymin, xmax, ymax$.
 - $xmin$ là giới hạn cạnh bên trái của khung.
 - $ymin$ là giới hạn cạnh bên trên của khung.
 - $xmax$ là giới hạn cạnh bên phải của khung.

- x_{max} là giới hạn cạnh bên dưới của khung.
- Phân loại lớp của đối tượng trong khung giới hạn. Trong trường hợp này chỉ có một lớp đó là lớp người.
- Chỉ số tin cậy. Chỉ số này có giá trị trong khoảng từ 0 đến 1. Chỉ số tin cậy càng cao, mô hình càng chắc chắn trong dự đoán.

a, Tiền xử lý

Trước khi đưa các hình ảnh vào mô hình phát hiện, chúng cần được tiền xử lý để chuyển đổi tất cả các hình ảnh có độ phân giải khác nhau thành cùng một định dạng. Phương pháp tiền xử lý phải hoàn toàn giống như trong quá trình huấn luyện mô hình. Với kích thước mong muốn là 384×384 nên hình ảnh gốc được thay đổi kích thước thành độ phân giải này. Để tài lựa chọn độ phân giải này 384×384 thay vì độ phân giải ban đầu là 640×640 của YOLOv5n để giảm chi phí tính toán và giảm độ trễ. Tính toán trên độ phân giải cao hơn sẽ gây ra quá tải năng lực phần cứng đối với thiết bị biên.

Với các khung hình đầu vào có định dạng chữ nhật, cạnh dài của ảnh gốc sẽ được biến đổi về độ dài 384, và tất nhiên, cạnh còn lại cũng sẽ phải được thu về với cùng tỷ lệ. Điều này khiến cho độ dài của cạnh ngắn sẽ nhỏ hơn kích thước 384. Do đó, để đạt được độ phân giải mong muốn là 384, cả hai bên của cạnh ngắn sẽ được thêm vào khung hình khoảng xám với giá trị là (114, 114, 114). Tiếp theo, các giá trị pixel sẽ bị chia cho 255 để chuẩn hóa về giá trị $[0, 1]$.

b, Phát hiện người với mô hình YOLOv5n

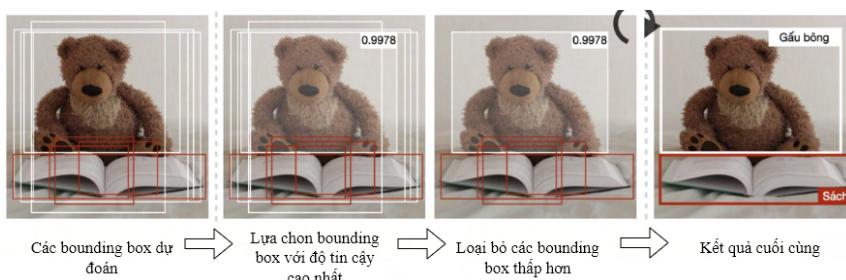
Nghiên cứu đã sử dụng trọng số được huấn luyện trước của YOLOv5n. Những trọng số này đã được huấn luyện trước bằng bộ dữ liệu COCO. Tôi chỉ sử dụng lớp “person” và bỏ qua tất cả các lớp khác. Phiên bản này của trọng số được huấn luyện trước đã được huấn luyện với kích thước đầu vào là 640×640 . Mặc dù kích thước đầu vào này không phải là kích thước mong muốn của tôi là 384×384 , mô hình được huấn luyện này vẫn có thể được sử dụng tốt do kiến trúc pyramid của YOLOv5n cho phép nó hoạt động ổn định với các đối tượng có kích thước khác nhau. Hơn nữa, kiến trúc mạng tích chập cho phép tính toán với hình ảnh đầu vào có độ phân giải tùy ý. Do tài nguyên tính toán hạn chế trên các thiết bị biên, đề tài chọn thực hiện mô hình trên độ phân giải nhỏ hơn để giảm chi phí tính toán. Hơn nữa, mô hình này sau đó được chuyển đổi thành một TensorRT engine để thực hiện tính toán hiệu suất cao liên quan đến học sâu trên Jetson Nano. Trọng số của nó đã được chuyển từ số thực 32-bit sang số thực 16-bit để tăng tốc độ tính toán.

Sau khi đưa dữ liệu đầu vào đã được xử lý vào mô hình, kết quả đầu ra có nhiều bounding box chồng lấn lên nhau. Do đó, cần có một thuật toán hậu xử lý để loại bỏ các bounding box dư thừa và gọi là thuật toán NMS sẽ được trình bày trong phần tiếp theo.

c, Non-maximum suppression

Mô hình phát hiện đối tượng tạo ra rất nhiều sự chồng lấn kết quả. Một cá nhân duy nhất trong hình ảnh có thể có nhiều bounding box. Một đối tượng với nhiều bounding box tương ứng có thể gây khó khăn ở giai đoạn sau. Tất cả các bounding box không cần thiết phải được lọc ra và chỉ giữ lại một bounding box tốt nhất. Bounding box còn lại này được chọn bằng thuật toán NMS. Nó giúp chọn ra một bounding box tốt nhất trong số nhiều bounding box chồng lấn nhau. Nó chọn một cặp bounding box trùng lấn bằng ngưỡng IoU và loại bỏ bounding box có độ tin cậy thấp hơn. Quá trình này được lặp lại cho đến khi không còn cặp phát hiện trùng lấn nào còn lại như trong Hình 4.5.

Thông qua các thử nghiệm, điểm IoU được chọn là 0,45 vì ngưỡng này cho thấy kết quả tốt nhất. Một ngưỡng quá cao có thể làm cho các bounding box không cần thiết vẫn tồn tại, trong khi một ngưỡng quá thấp có thể dẫn đến việc bounding box của hai cá nhân khác nhau đứng cạnh nhau bị hợp nhất. Ngoài ra, trước khi chạy NMS, tất cả các bounding box có độ tin cậy thấp hơn 0,45 đều được lọc ra trước để giảm công việc cho NMS.

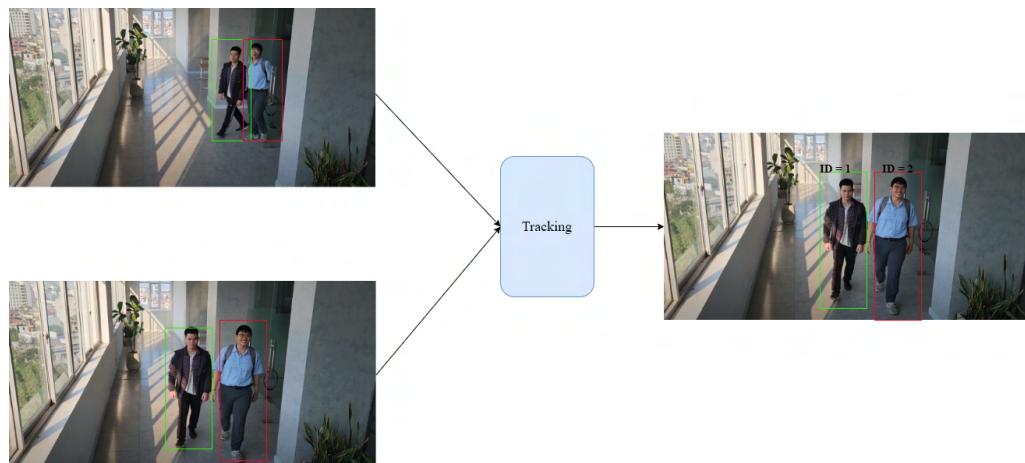


Hình 4.5: Quá trình loại bỏ bounding box bằng NMS

4.2.3 Mô-đun theo vết người

Thuật toán theo vết giúp theo dõi những người di chuyển trong các khung hình, nó giúp duy trì cùng một định danh cho một người khi di chuyển trong một camera. Hình 4.6 thể hiện tổng quan thuật toán theo vết, nó giữ nguyên ID của mỗi di chuyển trong camera. Thuật toán theo vết người có đầu vào là các bounding box từ mô hình phát hiện người ở mục 4.2.2 và đối tượng đã được theo vết ở khung hình trước. Đầu ra của giải thuật này là các vật thể được đã định danh cố định trong xuyên suốt toàn

thời gian của luồng video đầu vào. Mỗi đối tượng sẽ được gán một ID duy nhất, cùng một người xuất hiện ở nhiều khung hình liên tiếp nhau sẽ được định danh bằng một ID, kết hợp với đó là bounding box của mô hình phát hiện đối tượng. Thông tin này sẽ giúp hệ thống phân biệt nhiều người khác nhau xuất hiện trong cùng một đoạn video từ đầu cho đến cuối.



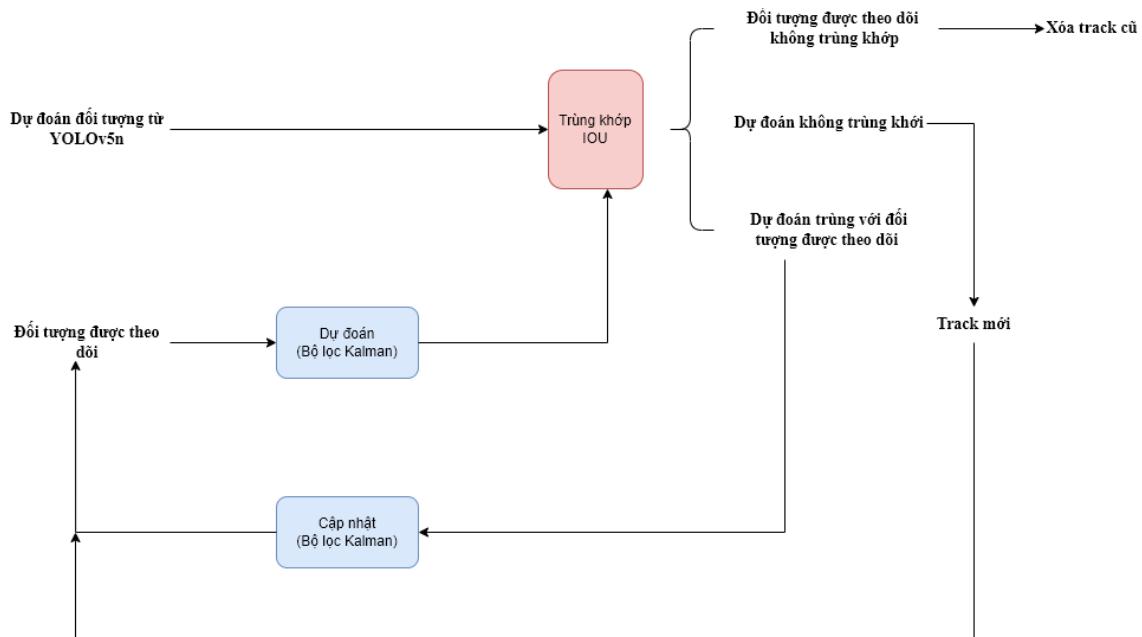
Hình 4.6: Tổng quan thuật toán truy vết

Để có thể theo vết một vật thể bất kỳ, luồng xử lý mỗi khung hình sẽ gồm các bước sau:

- Detect: phát hiện vị trí các đối tượng trong khung hình.
- Predict: Dự đoán vị trí mới của các đối tượng dựa vào các khung hình trước đó.
- Associate: Liên kết các vị trí đã phát hiện với các vị trí dự đoán được để gán ID tương ứng.

Thuật toán SORT [44] đề xuất giải pháp cho bài toán theo vết, nó đồng thời giải quyết cả 2 vấn đề theo vết đa đối tượng và thời gian thực. Trong đó, phần phát hiện được thực hiện bởi mô hình YOLOv5n, phần dự đoán và liên kết các vị trí được thực hiện bởi SORT.

Luồng xử lý của thuật toán SORT bao gồm các bước chính như trong Hình 4.7: Thuật toán bắt đầu bằng việc khởi tạo danh sách trống để lưu trữ các đối tượng đang được theo dõi và đặt một số biến cố định, chẳng hạn như ngưỡng IoU để xem xét sự khớp giữa các dự đoán và đối tượng đã theo dõi. Khi nhận được dự đoán từ mô hình phát hiện đối tượng, SORT sẽ đưa dự đoán vào thuật toán theo vết. Tiếp theo, Đối với mỗi đối tượng đã được theo dõi, thuật toán thực hiện dự đoán vị trí tiếp theo dựa trên thông tin vị trí hiện tại và tốc độ di chuyển ước tính. Sau đó, nó



Hình 4.7: Thuật toán truy vết SORT

so khớp các dự đoán này với các đối tượng đã được theo dõi dựa trên ngưỡng IoU. Dựa trên sự so khớp và tính nhất quán, thuật toán cập nhật danh sách các đối tượng đang được theo dõi. Đối tượng nào không có sự so khớp với dự đoán nào hoặc đã vượt qua một thời gian xác định mà không có sự cập nhật sẽ bị loại bỏ. Cùng với đó, Các dự đoán mới mà không có sự so khớp với bất kỳ đối tượng nào trong danh sách sẽ được xem xét là đối tượng mới. Chúng sẽ được thêm vào danh sách để theo dõi. Nếu một đối tượng biến mất (không có dự đoán hoặc dự đoán không phù hợp trong một khoảng thời gian), nó có thể được xem xét là đã biến mất và loại bỏ khỏi danh sách. Cuối cùng, kết quả là danh sách các đối tượng đang được theo dõi cùng với thông tin vị trí của họ trong từng khung hình. Quá trình này tiếp tục cho đến khi không còn dự đoán nào còn lại hoặc kết thúc việc xử lý các khung hình.

Hai thuật toán cốt lõi của SORT là bộ lọc Kalman Filter và giải thuật Hungarian. Trong đó, thuật toán Hungarian được áp dụng để tối ưu hóa việc gán các dự đoán mới cho các đối tượng đã được theo dõi. Còn bộ lọc Kalman thường sử dụng để dự đoán và cập nhật vị trí của các đối tượng đã được theo dõi qua các khung hình liên tiếp. Để ứng dụng được Bộ lọc Kalman, việc xác định được các dạng biến cũng như mô hình ban đầu của quá trình là điều bắt buộc cần có. Với giả định các đối tượng chuyển động đều, và độc lập với các đối tượng khác, một dự đoán Detection xác định bởi YOLOv5n, một đối tượng được theo dõi được xác định bằng:

$$z = [x, y, s, r, \dot{x}, \dot{y}, \dot{s}]^T \quad (4.1)$$

Với:

- z có ma trận hiệp phương sai ban đầu được khởi tạo với giá trị lớn để thể hiện sự không chắc chắn của trạng thái.
- x, y lần lượt là tọa độ của tâm đối tượng.
- s là diện tích của bounding box.
- r là tỷ lệ các cạnh của bounding box.
- $\dot{x}, \dot{y}, \dot{s}$ lần lượt là các giá trị vận tốc tương ứng của x, y, s .

4.3 Thực nghiệm và kết quả

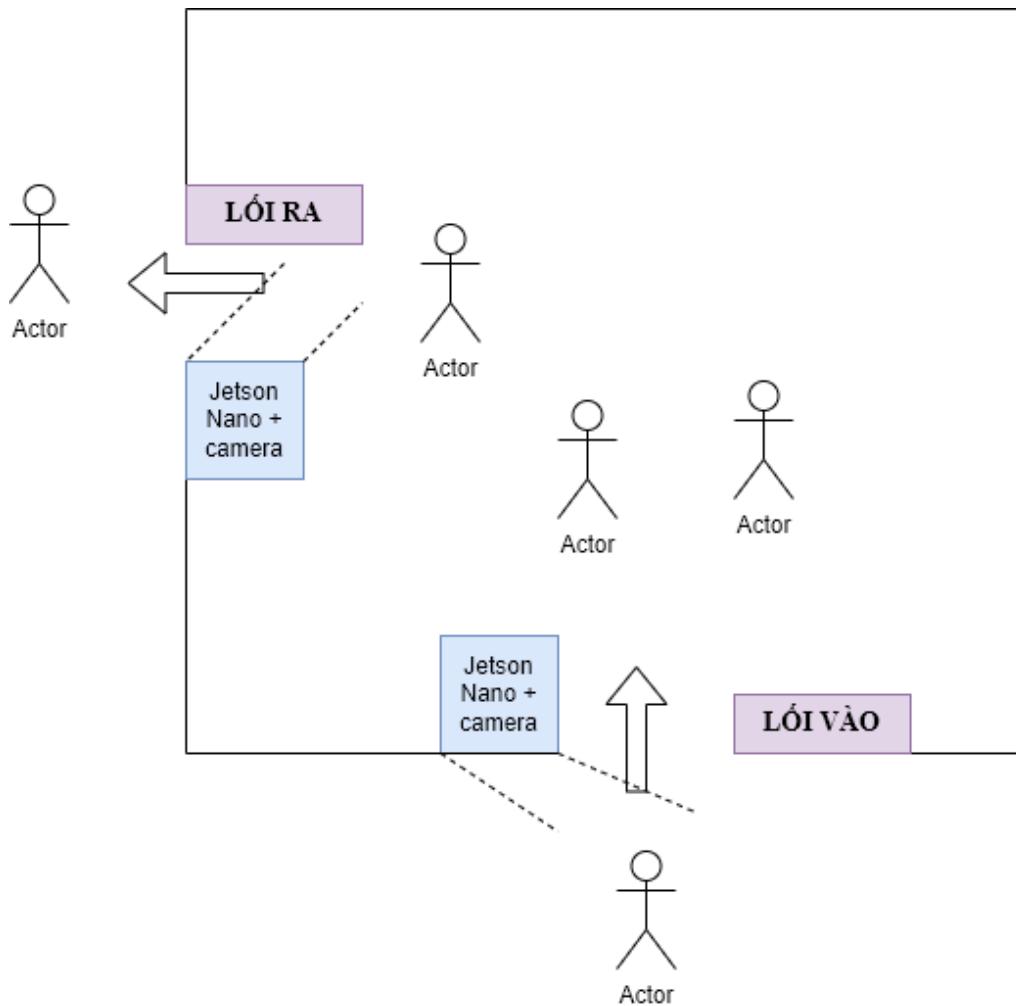
4.3.1 Thiết lập môi trường thực nghiệm

Để thực nghiệm đánh giá ứng dụng thiết bị biên cho bài toán tái định danh, đề tài đã sử dụng tầng 10 của tòa nhà B1 thuộc Đại học Bách Khoa Hà Nội để tiến hành lắp đặt thiết bị. Sẽ có hai nút thiết bị biên được đặt ở hai góc khác nhau thay cho lối vào và lối ra của một tòa nhà như Hình 4.8. Mỗi nút bao gồm một camera được kết nối với một Jetson Nano. Khi có người đi đến lối vào, camera sẽ ghi hình người này và gửi cho thiết bị biên để tiến hành xử lý hình ảnh. Sau đó, khi người này đi qua lối cửa ra thì cũng sẽ bị ghi hình bởi nút ở cửa ra. Các thông tin về hình ảnh và ID trên từng nút sẽ được gửi về máy chủ trung tâm, tại đây sẽ có chương trình để trích xuất đặc trưng và so sánh các đặc trưng người ở hai camera. Từ đó, thu được kết quả đó có phải là một người hay không.

Hình 4.9 và Hình 4.10 mô tả hai vị trí thực nghiệm thực tế tại tòa B1, với hai vị trí này, nghiên cứu sẽ kiểm chứng được với hai trường hợp: đầu tiên là môi trường hành lang, với ánh sáng đầy đủ mô phỏng điều kiện đặt camera ngoài trời; trường hợp hai là lắp camera trong phòng với ánh sáng yếu hơn mô phỏng trường hợp trong các tòa nhà, siêu thị.

Máy chủ được sử dụng trong thực nghiệm là một máy tính cá nhân có trang bị CPU AMD Ryzen 5 5600H, GPU NVIDIA GeForce RTX3050 Ti 4GB, RAM 8GB. Các thiết bị được dùng trong thực nghiệm đều có chi phí đầu tư không quá cao, phù hợp với người dùng cá nhân hoặc triển khai số lượng lớn.

Để hiển thị trực quan hình ảnh và kết quả xử lý máy chủ, một ứng dụng đơn giản đã được nhóm nghiên cứu viết trên hệ điều hành Window như Hình 4.11. Qua đó, người dùng có thể kiểm tra hình ảnh camera cũng như nhận được các thông tin ID của những người trong camera.



Hình 4.8: Vị trí lắp các thiết bị biên



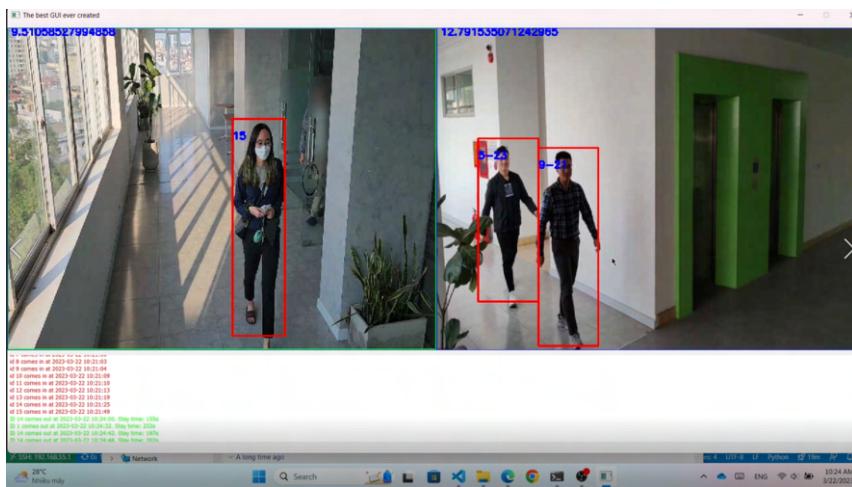
Hình 4.9: Vị trí thực nghiệm số 1

4.3.2 Bộ dữ liệu

Để triển khai và đánh giá, ba tập dữ liệu đã được sử dụng đó là Penn-Fudan [75], CUHK03 [76] và một tập dữ liệu do chính nhóm thực hiện thu thập là BKREID. Trong đó, Penn-Fudan và CUHK03 là hai tập dữ liệu đã được công bố.



Hình 4.10: Vị trí thực nghiệm số 2



Hình 4.11: Ứng dụng giám sát camera trên máy chủ

a, Bộ dữ liệu Penn-Fudan

Bộ dữ liệu chứa hình ảnh, các bounding box và mặt nạ phân đoạn (segmentation masks) của những người được chụp tại các khuôn viên trường học và đường phố đô thị. Số lượng người đi bộ trên mỗi hình ảnh dao động từ một đến tối đa tám, với tổng cộng 170 hình ảnh, 345 người được ghi nhãn và 423 bounding box. Hình 4.12 mô tả một số mẫu dữ liệu lấy từ bộ Penn-Fudan, có thể thấy các dữ liệu được lấy từ nhiều địa điểm ngoài trời, số lượng người trong ảnh đa dạng từ một cho đến mười người.

b, Bộ dữ liệu CUHK03

Bộ dữ liệu này bao gồm 14097 bức ảnh của 1467 cá nhân [76]. Sáu máy ảnh tại trường học đã được sử dụng để thu thập hình ảnh, mỗi cá nhân xuất hiện trong ít nhất trong hai máy. Trong bộ dữ liệu này, có hai loại nhãn bounding box: nhãn thủ công và nhãn tự động bằng cách sử dụng một trình phát hiện. Bộ dữ liệu này cũng cung cấp 20 phân vùng huấn luyện và thử nghiệm, trong đó có 100 cá nhân được



Hình 4.12: Một số hình ảnh từ bộ dữ liệu Penn-Fudan

dành riêng cho thử nghiệm và các mẫu còn lại được coi là mẫu huấn luyện. Trong thực nghiệm, tôi chỉ xem xét phần được ghi nhãn thủ công để huấn luyện, đánh giá và so sánh. Hình 4.13 trích xuất một số hình ảnh từ bộ dữ liệu CUHK03.

c, Bộ dữ liệu BKREID

Bộ dữ liệu đề xuất có tên là BKREID, nó được sử dụng để phát triển các mô hình tái định danh và đóng góp vào việc cải thiện hiệu suất của hệ thống. Bộ dữ liệu này được chụp trong nhà với ánh sáng ổn định, vì vậy nó sẽ không bị ảnh hưởng bởi các yếu tố thời tiết và ánh sáng ngoại trời, khác với bộ dữ liệu CUHK03.

Bộ dữ liệu đề xuất bao gồm bốn cặp video, mỗi cặp bao gồm hai video tương ứng với hai máy quay được đặt ở hai vị trí: cửa vào và cửa ra. Máy quay đầu tiên ghi hình ảnh của những người vào cửa vào; máy ảnh này được sử dụng chủ yếu để thu thập hình ảnh. Máy thứ hai quay những người ra cửa ra; máy ảnh này chủ yếu được sử dụng để xác định danh tính của họ. Bộ dữ liệu chứa 23160 hình ảnh của 37 người và được đánh dấu cẩn thận với sự hỗ trợ của một trình phát hiện đối tượng. Bộ dữ liệu này được sử dụng cho mục đích đào tạo và đánh giá. Hình 4.14 trích xuất một số hình ảnh từ bộ dữ liệu BKREID.



Hình 4.13: Một số hình ảnh từ bộ dữ liệu CUHK03

4.3.3 Kết quả thực nghiệm

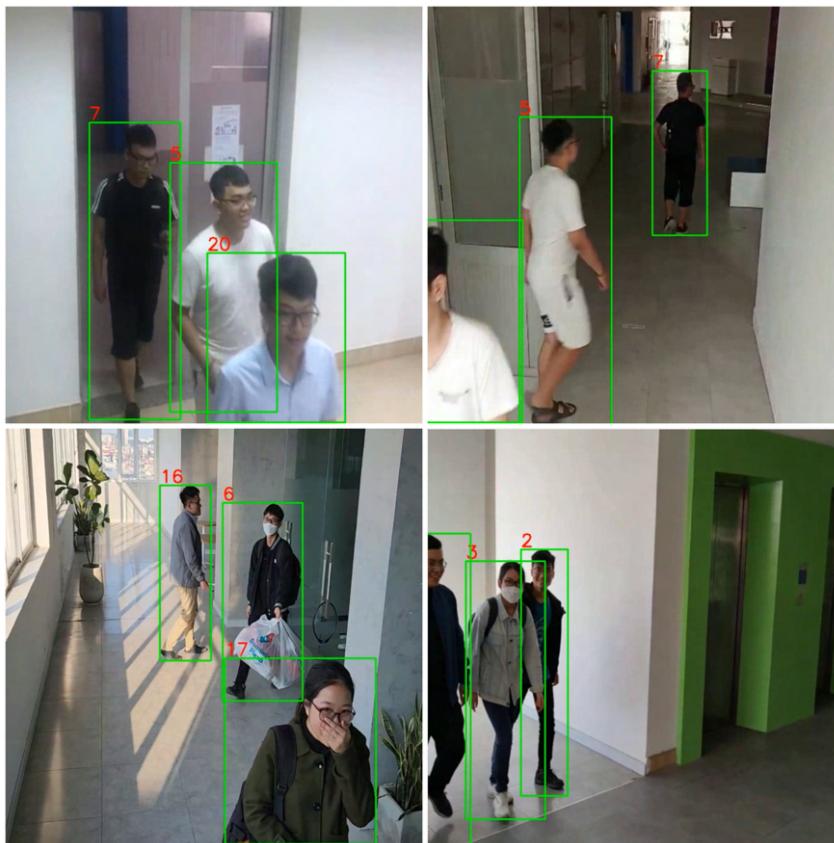
a, Kết quả của mô hình phát hiện người

Nghiên cứu đã tiến hành đánh giá hiệu suất của mô-đun phát hiện trên các thiết bị biên thông qua hai kịch bản khác nhau. Trong kịch bản đầu tiên, mô-đun được thực thi trong môi trường PyTorch nguyên bản, trong khi kịch bản thứ hai yêu cầu chuyển đổi mô-đun thành dạng TensorRT trước khi thực thi.

Tất cả các hình ảnh đều được đưa vào mô hình để nhận thô kết quả ban đầu, sau đó một bước tiền xử lý được thực hiện bằng thuật toán NMS để loại bỏ các bounding box trùng lặp và thu được dự đoán cuối cùng về bounding box. Các chỉ số hiệu suất được tính toán sau khi qua bước NMS.

Cả hai thông số quan trọng, gồm tỷ lệ IoU và ngưỡng độ tin cậy, đã được đặt là 0,45 và 0,25 tương ứng trong cả hai kịch bản. Điều này có nghĩa rằng chỉ có các bounding box có độ tin cậy bằng hoặc cao hơn 0,25 mới được xem xét trong quá trình đánh giá, và hai bounding box được xem xét là trùng lặp nếu chỉ số IoU giữa chúng lớn hơn hoặc bằng 0,45.

Kết quả từ Bảng 4.1 cho thấy rằng trong khi sự suy giảm hiệu suất là không



Hình 4.14: Hình ảnh từ bộ dữ liệu BKREID

Bảng 4.1: Kết quả của mô-đun phát hiện người trước và sau khi biến đổi sang TensorRT

Bộ dữ liệu	Trước khi biến đổi			Sau khi biến đổi		
	AP@0,5	mAP	FPS	AP@0,5	mAP	FPS
Bộ dữ liệu Penn-Fudan	0,956	0,963	13	0,956	0,962	29
Bộ dữ liệu BKREID	0,962	0,965	14	0,958	0,962	29

đáng kể, thời gian dự đoán của mô hình đã tăng gần gấp đôi trên thiết bị Jetson Nano au khi chuyển đổi sang định dạng tối ưu TensorRT. YOLOv5n trên thiết bị Jetson Nano hoạt động tốt với mức trung bình chính xác trung bình *mAP* khoảng 0,96 điểm và đồng thời cung cấp tốc độ thời gian thực. Có một sự suy giảm nhỏ trong *mAP* do việc chuyển đổi từ trọng số dạng số thập phân 32-bit sang trọng số thực 16-bit. Tuy nhiên, tác động này vẫn là tối ưu và không đáng kể. Kết quả một lần nữa cho thấy rằng việc chuyển đổi mô hình đúng cách có thể tạo ra một sự tăng cường đáng kể về tốc độ, mà không có sự đánh đổi độ chính xác nhiều.

Hình 4.15 cho thấy mô hình phát hiện người chạy ổn định tại các môi trường thí nghiệm khác nhau, bao gồm: hành lang sáng, hành lang thiếu sáng, trong phòng, và ngoài phòng.

Hình 4.16 mô tả kết quả thực nghiệm với trường hợp thay đổi số lượng và vị trí



Hình 4.15: Kết quả thực nghiệm với những môi trường khác nhau

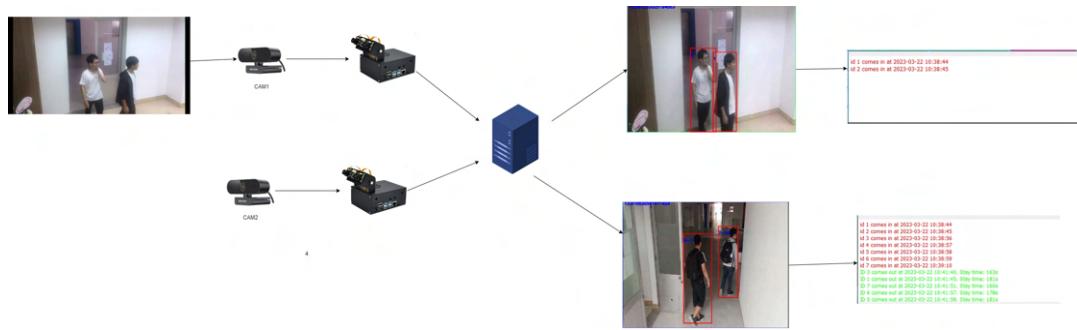
đứng trong khung hình, từ một người đến ba người. Ngoài ra, để kiểm tra độ chính xác của mô hình phát hiện và truy vết, đề tài đã sắp xếp người đi vào khung hình theo dạng hàng dọc, hàng ngang, và xen lấn nhau để mô phỏng như tình huống ngoài thực tế.



Hình 4.16: Kết quả thực nghiệm với các số lượng người khác nhau trong khung hình

b, Kết quả của giải pháp tái định danh người

Trong quá trình kiểm tra, kết quả cho thấy cả mô hình thử nghiệm hoạt động tốt và ổn định. Tốc độ hoạt động của hệ thống trung bình khoảng từ 13-15 FPS. Hình 4.17 mô hình thực nghiệm của mô hình tái định danh. Hình 4.18 biểu diễn



Hình 4.17: Mô hình thực nghiệm của hệ thống tái định danh người

kết quả tái định danh một người khi xuất hiện trên hai camera. Đầu tiên, người này xuất hiện ở camera một, hệ thống đã phát hiện và gán ID cho người đó là 19. Khi người này lọt vào trong vùng quan sát của camera hai, hình ảnh của anh ta đã được truyền về hệ thống máy chủ và sau khi so sánh vec-to đặc trưng của với cơ sở dữ liệu, hệ thống đã phát hiện ra ID của của người này là 19. Các kết quả khác của thực nghiệm được trình bày trong Hình 4.19



Hình 4.18: Kết quả thực nghiệm phát hiện và tái định danh người



Hình 4.19: Một số kết quả thực nghiệm phát hiện và tái định danh người

4.4 Kết luận

Đề tài đề xuất phương pháp tái định danh người toàn trình dành cho các thiết bị có tài nguyên hạn chế. Phương pháp này bao gồm việc nhận diện đối tượng trên thiết bị biên Jetson Nano kết hợp với mô hình theo vết và trích xuất đặc trưng trên máy chủ. Cụ thể, trong phạm vi nghiên cứu của luận văn, mô hình phát hiện đối tượng trên thiết bị biên đã đạt được mục tiêu đề ra bao gồm: tốc độ dự đoán nhanh và kết quả chính xác trên các tập dữ liệu được đánh giá. TensorRT đã được sử dụng để tối ưu hóa các mô hình của học sâu, giúp chúng có thể hoạt động hiệu quả trên các thiết bị biên như Jetson Nano. Kết quả đạt được từ thực nghiệm cho thấy mô hình hoạt động ổn định trong các bối cảnh khác nhau bao gồm số lượng hai đến ba người, không gian ánh sáng khác nhau. Tuy nhiên, bộ truy vết SORT vẫn gặp hiện tượng nhảy ID dẫn đến việc tái định danh vẫn chưa đạt độ chính xác cao.

CHƯƠNG 5. KẾT LUẬN

5.1 Kết luận

5.1.1 Kết quả

Luận văn đã trình bày các kết quả nghiên cứu và các nội dung công việc trong quá trình thực hiện luận văn tốt nghiệp với hai kết quả nổi bật, bao gồm:

Trong phần đầu tiên, luận văn nghiên cứu và đề xuất mô hình xử lý cho bài toán phát hiện biển số xe bao gồm hai mô-đun là nhận diện phương tiện và mô-đun nhận diện biển số từ phương tiện. Đồng thời, luận văn đã giải quyết được vấn đề tăng tốc độ xử lý trên thiết bị biên có cấu hình hạn chế. Bằng cách chuyển đổi các mạng học sâu đã được huấn luyện sang dạng TensorRT, tốc độ xử lý trên thiết bị Jetson Nano đạt 42 FPS với bộ phát hiện phương tiện và 204 FPS với mạng phát hiện biển số. Để huấn luyện cho hai mô-đun, hai bộ dữ liệu phương tiện giao thông và biển số xe được tạo bằng cách kết hợp nhiều bộ dữ liệu lại với nhau nhằm tạo ra sự đa dạng, cũng như gần với thực tế tại điều kiện giao thông của Việt Nam.

Trong phần thứ hai, luận văn xây dựng được chương trình cho bài toán tái định danh bao gồm bộ phát hiện người chạy trên thiết bị biên, kết hợp với mô-đun truy vết và trích xuất đặc trưng trên máy chủ tập trung. Nghiên cứu cũng đóng góp bộ dữ liệu BKREID để huấn luyện cho mạng trích suất đặc trưng. Riêng đối với mô-đun phát hiện người sử dụng mạng YOLOv5n trên thiết bị Jetson Nano cho độ chính xác lên đến 0,962 mAP và tốc độ 29 FPS. Việc xử lý tái định danh người trên server cho tốc độ thấp hơn do ảnh hưởng bởi quá trình truyền dữ liệu và xử lý của mô-đun trích xuất đặc trưng.

Hơn nữa, trong quá trình hoàn thành luận văn này, tôi đã thu được rất nhiều kiến thức bổ ích về thị giác máy tính, các thuật toán trí tuệ nhân tạo và cách triển khai chúng trên các thiết bị biên. Tôi thấy việc triển khai chúng trên các thiết bị thực tế rất thách thức. Tất cả những điều này đã trở thành một kinh nghiệm quý báu đối với tôi.

5.1.2 Hạn chế

Bên cạnh những kết quả khả quan mà nghiên cứu đạt được, vẫn còn một số tồn tại cần được khắc phục trong thời gian tới như sau:

- Trong bài toán phát hiện biển số xe, độ chính xác của mô hình thấp khi gặp phải các thay đổi về mặt thời tiết như mưa, trời âm u. Thứ hai, việc xử lý mới chỉ dừng lại ở phát hiện biển số xe, chưa thực hiện xây dựng được hệ thống

toàn trình bao gồm cả nhận diện ký tự biển số hoặc phối hợp với các hệ thống giao thông khác. Thứ ba, bài toán mới chỉ dừng lại ở mức phòng thí nghiệm, chưa được kiểm tra khi triển khai trên đường phố để đánh giá khả năng của mô hình đề xuất.

- Đối với bài toán tái định danh, khi gặp trường hợp có nhiều người xuất hiện trong cùng một khung hình thì mô hình hoạt động chưa được chính xác.
- Chưa có cơ chế tự động khôi phục chương trình cho thiết bị biên khi gặp phải trường hợp bị khởi động lại một cách bất ngờ.

5.2 Hướng phát triển trong tương lai

Trong tương lai, để hoàn thiện hơn nghiên cứu của mình, luận văn đưa ra một số hướng phát triển như sau. Đối với bài toán phát hiện biển số xe:

- Thủ nghiệm với phiên bản mới nhất của YOLO để lựa chọn ra mạng phù hợp cho bài toán.
- Tối ưu hóa các mạng học sâu để có thể chạy được toàn bộ quy trình trên thiết bị biên, bao gồm: phát hiện phương tiện, phát hiện biển số và nhận diện biển số.
- Bổ sung các khâu tiền xử lý ảnh đầu vào để tăng độ chính xác khi gặp thời tiết mưa, hay âm u.
- Thực nghiệm mô hình đề xuất tại đường phố Việt Nam.

Đối với bài toán tái định danh, một số công việc có thể tiếp tục làm trong tương lai như sau:

- Tiếp tục tối ưu hóa các mạng học sâu để có thể triển khai toàn bộ bài toán tái định danh trên một thiết bị biên từ phát hiện người, truy vết cho đến trích xuất đặc trưng. Khi đó, máy chủ tập trung sẽ chỉ nhận các kết quả dạng vec-tơ đặc trưng và ID mỗi người để thực hiện tái định danh.
- Mở rộng bài toán với nhiều thiết bị biên hơn, đa dạng về chủng loại hơn như Raspberry Pi hay Google Coral.
- Thiết kế thêm cơ chế phục hồi cho thiết bị biên.
- Thực hiện nhiều triển khai thực tế hơn để phát hiện ra những điểm yếu cần được khắc phục.

TÀI LIỆU THAM KHẢO

- [1] M. Merenda, C. Porcaro **and** D. Iero, “Edge machine learning for ai-enabled iot devices: A review,” *Sensors*, **jourvol** 20, **number** 9, **page** 2533, 2020.
- [2] R. Zhang, W. Li **and** T. Mo, “Review of deep learning,” *arXiv preprint arXiv:1804.01653*, 2018.
- [3] J. Chen **and** X. Ran, “Deep learning with edge computing: A review,” *Proceedings of the IEEE*, **jourvol** 107, **number** 8, **pages** 1655–1674, 2019.
- [4] M. Zhang, F. Zhang, N. D. Lane **and others**, “Deep learning in the era of edge computing: Challenges and opportunities,” *Fog Computing: Theory and Practice*, **pages** 67–78, 2020.
- [5] J. Shashirangana, H. Padmasiri, D. Meedeniya **and** C. Perera, “Automated license plate recognition: A survey on methods and techniques,” *IEEE Access*, **jourvol** 9, **pages** 11 203–11 225, 2020.
- [6] Y. Guo, Z. Liu, H. Luo, H. Pu **and** J. Tan, “Multi-person multi-camera tracking for live stream videos based on improved motion model and matching cascade,” *Neurocomputing*, **jourvol** 492, **pages** 561–571, 2022.
- [7] B. Gaikwad **and** A. Karmakar, “Smart surveillance system for real-time multi-person multi-camera tracking at the edge,” *Journal of Real-Time Image Processing*, **jourvol** 18, **number** 6, **pages** 1993–2007, 2021.
- [8] O. Elharrouss, N. Almaadeed **and** S. Al-Maadeed, “A review of video surveillance systems,” *Journal of Visual Communication and Image Representation*, **jourvol** 77, **page** 103 116, 2021.
- [9] Z.-Q. Zhao, P. Zheng, S.-t. Xu **and** X. Wu, “Object detection with deep learning: A review,” *IEEE transactions on neural networks and learning systems*, **jourvol** 30, **number** 11, **pages** 3212–3232, 2019.
- [10] N. O’Mahony, S. Campbell, A. Carvalho **and others**, “Deep learning vs. traditional computer vision,” *inAdvances in Computer Vision: Proceedings of the 2019 Computer Vision Conference (CVC), Volume 1* Springer, 2020, **pages** 128–144.
- [11] N. Dalal **and** B. Triggs, “Histograms of oriented gradients for human detection,” *in2005 IEEE computer society conference on computer vision and pattern recognition (CVPR’05) Ieee*, **volume** 1, 2005, **pages** 886–893.
- [12] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, **jourvol** 60, **pages** 91–110, 2004.

- [13] H. Bay, T. Tuytelaars **and** L. Van Gool, “Surf: Speeded up robust features,” **in***Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7–13, 2006. Proceedings, Part I* 9 Springer, 2006, **pages** 404–417.
- [14] C. Cortes **and** V. Vapnik, “Support vector machine,” *Machine learning*, **jourvol** 20, **number** 3, **pages** 273–297, 1995.
- [15] H. Harzallah, C. Schmid, F. Jurie **and** A. Gaidon, “Classification aided two stage localization,” **in***PASCAL Visual Object Classes Challenge Workshop, in conjunction with ECCV 2008*.
- [16] Y. Freund **and** R. E. Schapire, “A desicion-theoretic generalization of on-line learning and an application to boosting,” **in***European conference on computational learning theory* Springer, 1995, **pages** 23–37.
- [17] D.-L. Dinh, H.-N. Nguyen, H.-T. Thai **and** K.-H. Le, “Towards ai-based traffic counting system with edge computing,” *Journal of Advanced Transportation*, **jourvol** 2021, **pages** 1–15, 2021.
- [18] A. Krizhevsky, I. Sutskever **and** G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” *Advances in neural information processing systems*, **jourvol** 25, 2012.
- [19] H. Ghahremannezhad, H. Shi **and** C. Liu, “Object detection in traffic videos: A survey,” *IEEE Transactions on Intelligent Transportation Systems*, 2023.
- [20] R. Girshick, J. Donahue, T. Darrell **and** J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2014, **pages** 580–587.
- [21] L. Jiao, F. Zhang, F. Liu **and others**, “A survey of deep learning-based object detection,” *IEEE access*, **jourvol** 7, **pages** 128 837–128 868, 2019.
- [22] S. Ren, K. He, R. Girshick **and** J. Sun, “Faster r-cnn: Towards real-time object detection with region proposal networks,” *Advances in neural information processing systems*, **jourvol** 28, 2015.
- [23] R. Girshick, “Fast r-cnn,” **in***Proceedings of the IEEE international conference on computer vision* 2015, **pages** 1440–1448.
- [24] W. Liu, D. Anguelov, D. Erhan **and others**, “Ssd: Single shot multibox detector,” **in***Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I* 14 Springer, 2016, **pages** 21–37.

- [25] J. Redmon, S. Divvala, R. Girshick **and** A. Farhadi, “You only look once: Unified, real-time object detection,” *inProceedings of the IEEE conference on computer vision and pattern recognition 2016*, **pages** 779–788.
- [26] T.-Y. Lin, P. Goyal, R. Girshick, K. He **and** P. Dollár, “Focal loss for dense object detection,” *inProceedings of the IEEE international conference on computer vision 2017*, **pages** 2980–2988.
- [27] S. Zhang, L. Wen, X. Bian, Z. Lei **and** S. Z. Li, “Single-shot refinement neural network for object detection,” *inProceedings of the IEEE conference on computer vision and pattern recognition 2018*, **pages** 4203–4212.
- [28] T. Diwan, G Anirudh **and** J. V. Tembhurne, “Object detection using yolo: Challenges, architectural successors, datasets and applications,” *multimedia Tools and Applications*, **jourvol** 82, **number** 6, **pages** 9243–9275, 2023.
- [29] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. Asghar **and** B. Lee, “A survey of modern deep learning based object detection models,” *Digital Signal Processing*, **jourvol** 126, **page** 103 514, 2022.
- [30] R. Padilla, W. L. Passos, T. L. Dias, S. L. Netto **and** E. A. Da Silva, “A comparative analysis of object detection metrics with a companion open-source toolkit,” *Electronics*, **jourvol** 10, **number** 3, **page** 279, 2021.
- [31] J. Redmon **and** A. Farhadi, “Yolo9000: Better, faster, stronger,” *inProceedings of the IEEE conference on computer vision and pattern recognition 2017*, **pages** 7263–7271.
- [32] A. Bochkovskiy, C.-Y. Wang **and** H.-Y. M. Liao, “Yolov4: Optimal speed and accuracy of object detection,” *arXiv preprint arXiv:2004.10934*, 2020.
- [33] *Yolov5*, <https://github.com/ultralytics/yolov5/>, [Online; accessed 19-April-2021].
- [34] C. Li, L. Li, H. Jiang **and others**, “Yolov6: A single-stage object detection framework for industrial applications,” *arXiv preprint arXiv:2209.02976*, 2022.
- [35] C.-Y. Wang, A. Bochkovskiy **and** H.-Y. M. Liao, “Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors,” *inProceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition 2023*, **pages** 7464–7475.
- [36] D. Reis, J. Kupec, J. Hong **and** A. Daoudi, “Real-time flying object detection with yolov8,” *arXiv preprint arXiv:2305.09972*, 2023.
- [37] A. M. Roy, R. Bose **and** J. Bhaduri, “A fast accurate fine-grain object detection model based on yolov4 deep neural network,” *Neural Computing and Applications*, **pages** 1–27, 2022.

- [38] S. Saponara, A. Elhanashi **and** Q. Zheng, “Developing a real-time social distancing detection system based on yolov4-tiny and bird-eye view for covid-19,” *Journal of Real-Time Image Processing*, **jourvol** 19, **number** 3, **pages** 551–563, 2022.
- [39] M. M. Islam, A. A. R. Newaz **and** A. Karimoddini, “A pedestrian detection and tracking framework for autonomous cars: Efficient fusion of camera and lidar data,” **in**2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC) IEEE, 2021, **pages** 1287–1292.
- [40] Ö. Kaya, M. Y. Çodur **and** E. Mustafaraj, “Automatic detection of pedestrian crosswalk with faster r-cnn and yolov7,” *Buildings*, **jourvol** 13, **number** 4, **page** 1070, 2023.
- [41] C.-J. Lin, C.-C. Chuang **and** H.-Y. Lin, “Edge-ai-based real-time automated license plate recognition system,” *Applied Sciences*, **jourvol** 12, **number** 3, **page** 1445, 2022.
- [42] S. M. Silva **and** C. R. Jung, “License plate detection and recognition in unconstrained scenarios,” **in**Proceedings of the European conference on computer vision (ECCV) 2018, **pages** 580–596.
- [43] D. Reid, “An algorithm for tracking multiple targets,” *IEEE transactions on Automatic Control*, **jourvol** 24, **number** 6, **pages** 843–854, 1979.
- [44] A. Bewley, Z. Ge, L. Ott, F. Ramos **and** B. Upcroft, “Simple online and realtime tracking,” **in**2016 IEEE international conference on image processing (ICIP) IEEE, 2016, **pages** 3464–3468.
- [45] L. Mekkayil **and** H. Ramasangu, “Object tracking with correlation filters using selective single background patch,” *arXiv preprint arXiv:1805.03453*, 2018.
- [46] D. S. Bolme, J. R. Beveridge, B. A. Draper **and** Y. M. Lui, “Visual object tracking using adaptive correlation filters,” **in**2010 IEEE computer society conference on computer vision and pattern recognition IEEE, 2010, **pages** 2544–2550.
- [47] J. Vergés-Llahí, J. Aranda, A. Sanfeliu **and**others, “Object tracking system using colour histograms,” **in**Proceedings of the 9th Spanish Symposium on Pattern Recognition and Image Analysis 2001, **pages** 225–230.
- [48] Z. Zivkovic **and** B. Krose, “An em-like algorithm for color-histogram-based object tracking,” **in**Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004. IEEE, **volume** 1, 2004, **pages** I–I.

- [49] N. Wojke, A. Bewley **and** D. Paulus, “Simple online and realtime tracking with a deep association metric,” **in***2017 IEEE international conference on image processing (ICIP)* IEEE, 2017, **pages** 3645–3649.
- [50] R. E. Kalman, “A new approach to linear filtering and prediction problems,” 1960.
- [51] Nvidia, *Embedded Systems with Jetson*, <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/>, [Online; accessed 20-August-2023].
- [52] NVIDIA, *NVIDIA Jetson Nano*, <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-nano/>, [Online; accessed 20-August-2023].
- [53] NVIDIA, *NVIDIA Jetson TX2*, <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/>, [Online; accessed 20-August-2023].
- [54] NVIDIA, *NVIDIA Jetson Xavier*, <https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-xavier-series/>, [Online; accessed 20-August-2023].
- [55] NVIDIA, *NVIDIA TensorRT*, <https://developer.nvidia.com/tensorrt>, [Online; accessed 20-August-2023].
- [56] J. Redmon, *Darknet: Open source neural networks in c*, <http://pjreddie.com/darknet/>, 2013–2016.
- [57] A. Paszke, S. Gross, F. Massa **and others**, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, **jourvol** 32, 2019.
- [58] G. Bradski, “The opencv library.” *Dr. Dobb’s Journal: Software Tools for the Professional Programmer*, **jourvol** 25, **number** 11, **pages** 120–123, 2000.
- [59] P. Hintjens, *ZeroMQ: messaging for many applications.* " O'Reilly Media, Inc.", 2013.
- [60] T. L. Dang, D. L. Le, T. H. Pham **and** X. T. Tran, “Lightweight models’ performances on a resource-constrained device for traffic application,” **in***Deep Learning and Other Soft Computing Techniques: Biomedical and Related Applications* N. H. Phuong **and** V. Kreinovich, **editors**. Cham: Springer Nature Switzerland, 2023, **pages** 183–194, ISBN: 978-3-031-29447-1. DOI: 10.1007/978-3-031-29447-1_16. **url:** https://doi.org/10.1007/978-3-031-29447-1_16.

- [61] Y. Wen, Y. Lu, J. Yan, Z. Zhou, K. M. von Deneen **and** P. Shi, “An algorithm for license plate recognition applied to intelligent transportation system,” *IEEE Transactions on intelligent transportation systems*, **jourvol** 12, **number** 3, **pages** 830–845, 2011.
- [62] M. Usama, H. Anwar, A. Anwar **and** S. Anwar, “Vehicle and license plate recognition with novel dataset for toll collection,” *arXiv preprint arXiv:2202.05631*, 2022.
- [63] J. Shashirangana, H. Padmasiri, D. Meedeniya **and others**, “License plate recognition using neural architecture search for edge devices,” *International Journal of Intelligent Systems*, **may** 2021. DOI: 10.1002/int.22471.
- [64] NVIDIA, *Accelerating Inference Up to 6x Faster in PyTorch with Torch-TensorRT*, <https://developer.nvidia.com/blog/accelerating-inference-up-to-6x-faster-in-pytorch-with-torch-tensorrt/>, [Online; accessed 22-August-2023].
- [65] Vietnamese vehicles dataset, <https://www.kaggle.com/datasets/duongtran1909/vietnamese-vehicles-dataset>, Accessed: 2022-08-06.
- [66] Vehicles-openimages dataset, <https://public.roboflow.com/object-detection/vehicles-openimages>, Accessed: 2022-08-07.
- [67] Cars dataset, standford ai hub, http://ai.stanford.edu/~jkrause/cars/car_dataset.html, Accessed: 2022-08-07.
- [68] Application-oriented license plate recognition dataset, <https://github.com/AvLab-CV/AOLP>, Accessed: 2022-08-07.
- [69] Cartgmt dataset, https://drive.google.com/file/d/1U5ebTzW2c_sVVTCSX1QH-ZJFpLijMdUv/view, Accessed: 2022-08-07.
- [70] S. Karanam, Y. Li **and** R. J. Radke, “Person re-identification with discriminatively trained viewpoint invariant dictionaries,” **inProceedings of the IEEE international conference on computer vision 2015**, **pages** 4516–4524.
- [71] M. Ye, J. Shen, G. Lin, T. Xiang, L. Shao **and** S. C. Hoi, “Deep learning for person re-identification: A survey and outlook,” *IEEE transactions on pattern analysis and machine intelligence*, **jourvol** 44, **number** 6, **pages** 2872–2893, 2021.
- [72] M. Baharani, S. Mohan **and** H. Tabkhi, “Real-time person re-identification at the edge: A mixed precision approach,” **inInternational Conference on Image Analysis and Recognition** Springer, 2019, **pages** 27–39.

- [73] Y. Chen, T. Yang, C. Li **and** Y. Zhang, “A binarized segmented resnet based on edge computing for re-identification,” *Sensors*, **jourvol** 20, **number** 23, **page** 6902, 2020.
- [74] X. Chen, Z. Li, S. Xiao **and** Y. Chen, “Deep square similarity learning for person re-identification in the edge computing system,” **in***2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)* IEEE, 2018, **pages** 561–567.
- [75] L. Wang, J. Shi, G. Song **and** I.-f. Shen, “Object detection combining recognition and segmentation,” **in***Asian conference on computer vision* Springer, 2007, **pages** 189–199.
- [76] W. Li, R. Zhao, T. Xiao **and** X. Wang, “Deepreid: Deep filter pairing neural network for person re-identification,” **in***Proceedings of the IEEE conference on computer vision and pattern recognition* 2014, **pages** 152–159.