BỘ GIÁO DỤC VÀ ĐÀO TẠO
TRƯỜNG ĐẠI HỌC CẦN THƠ

CỔ BẢO HIẾU
MÃ SỐ HV: M3718007

XÂY DỰNG CÔNG CỤ PHÁT TRIỂN ỨNG DỤNG CHO CẢM BIẾN MÁY ẢNH TỰ ĐỘNG

LUẬN VĂN THẠC SĨ
NGÀNH KHOA HỌC MÁY TÍNH
MÃ SỐ 8 48 01 01

NGƯỜI HƯỚNG DẪN
TS. LÂM HOÀI BẢO

NĂM 2022

CHẤP THUẬN CỦA HỘI ĐỒNG

Luận văn này, với đề tựa là "Xây dựng công cụ phát triển ứng dụng cho cảm biến máy ảnh tự động", do học viên Cổ Bảo Hiếu thực hiện theo sự hướng dẫn của TS. Lâm Hoài Bảo. Luận văn đã báo cáo và được Hội đồng chấm luận văn thông qua ngày 25/06/2022.

Thành viên đọc luận văn sau khi chỉnh sửa
(ký tên)

TS. Thái Minh Tuấn

Chủ tịch Hội đồng
(ký tên)

PGS.TS. Phạm Nguyên Khang

Thư ký
(ký tên)

TS. Trần Việt Châu

Người hướng dẫn
(ký tên)

TS. Lâm Hoài Bảo

LỜI CẢM ƠN

Tôi xin chân thành gửi lời cảm ơn đến các quý Thầy, quý Cô trong Khoa Công Nghệ Thông Tin - Truyền Thông của trường Đại học Cần Thơ. Tôi cảm ơn vì những bài giảng đầy ấp kiến thức cùng những trái tim tràn đầy tâm huyết mà quý Thầy và quý Cô đã truyền đạt những kiến thức đến cho tôi khi tôi còn ngồi trên giảng đường trong thời gian qua. Tôi cũng không thể nào quên được các thông điệp mà các quý Thầy và quý Cô đã nhắn nhủ, cũng như gửi gấm đến cho tôi, và đó cũng sẽ là hành trang cho tôi bước đến sự thành công trên con đường tương lai.

Tôi cũng xin gửi lời biết ơn sâu sắc đến Thầy Lâm Hoài Bảo. Người Thầy đã nhận tôi về đội nghiên cứu của Thầy. Thầy mang đến tôi rất nhiều bài học, động lực, góp ý để tôi có thể thực hiện đề tài nghiên cứu luận văn "Xây dựng công cụ phát triển ứng dụng cho cảm biến máy ảnh tự động" và hoàn thành đề tài nghiên cứu một cách hoàn thiện nhất.

Tôi cũng muốn gửi cảm ơn đặc biệt đến các vị giảng viên, cán bộ, nhân viên của khoa Công Nghệ Thông Tin Truyền Thông và trường Đại học Cần Thơ, các anh chị em tập thể lớp cao học Khóa học máy tính khóa K25, cùng các anh chị em đồng nghiệp công ty DEK Technologies và công ty Bosch Global Software Technologies Vietnam đã động viên, giúp đỡ cho tôi trong suốt quá trình học tập cũng như quá trình thực hiện đề tài nghiên cứu luận văn.

Một lần nữa, Tôi xin chân thành cảm ơn và biết ơn!

Cần Thơ, ngày 03 tháng 02 năm 2022

Người viết

Cổ Bảo Hiếu

TÓM TẮT

Nghiên cứu này nhằm tạo ra một tập hợp các nghiên cứu xử lý liên quan đến con mắt của Robot đó là camera. Nghiên cứu sử dụng các phần cứng tương thích với lập trình nhúng từ đó tạo điều kiện cho những phần cứng với giá cả thấp nhưng phần mềm đi kèm vô cùng ấn tượng và hợp lý hơn không kém cạnh các hệ thống nghiên cứu cao cấp của các phần mềm với phần cứng có giá thành cao. Và dễ dàng sử dụng, dễ dàng cập nhật, dễ dàng gỡ lỗi, nhỏ gọn, tiện lợi mà với công nghệ xử lý các thuật toán AI mạnh mẽ bên trong.

Nghiên cứu sử dụng thư viện OpenCV là chủ đạo để giải quyết các vấn đề liên quan đến nghiên cứu nhập xuất hình ảnh. Nghiên cứu phát hiện đối tượng ở thời gian thực thì tất nhiên không thể bỏ qua được các mô hình mạng họ YOLO để xử lý các vấn đề liên quan đến các bài toán về phát hiện các khuôn mặt có mang khẩu trang hay không về việc tuân thủ V2K của Bộ Y tế, cũng như phát hiện các đối tượng là rau củ quả để thuận tiện cho việc xây dựng đầu bếp AI sau này. Song song đó, nghiên cứu còn dùng các thư viện hỗ trợ phát hiện khuôn mặt như Dlib, face_recognition để phát hiện và định danh các khuôn mặt nhằm phục vụ cho các hệ thống điểm danh học sinh, sinh viên trong trường học hay các hệ thống chấm công các công nhân viên hoặc những hệ thống mở cửa ít bảo mật trong nông trại.

Nghiên cứu còn cung cấp mã nguồn của chương trình nghiên cứu trên trang github với mong muốn xây dựng một chương trình tích hợp nhiều chức năng hơn nữa bên cạnh những chức năng như xử lý hình ảnh, phát hiện màu sắc, phát hiện mã QR, tăng tốc độ khung hình hay những chức năng AI đã kể ở trên. Qua đó, nghiên cứu hi vọng sẽ được cộng đồng đóng góp mã nguồn để tạo dựng sự ổn định cho chương trình và được cập nhật thường xuyên để sửa chữa khi chương trình gặp sự cố hay lỗi.

ABSTRACT

This research is aimed at creating a set of programming systems that relate to the robot's eye, which is the camera's primary visual processing system. Research on the use of embedded program-compatible hardware, thereby enabling lower-cost hardware and the accompanying software, is impressive and more affordable than systems with harder-cost hardware. Additionally, the program should be easy to use, easy to update, and easy to debug. It's also compact, convenient, and full of quality and powerful AI technology.

Research using the OpenCV library is mainstream for resolving issues related to image import and export systems. Regarding object detection in real-time, this research cannot ignore the YOLO family of network models that are designed to handle the problems of detecting faces with masks and not with COVID19. As well as detecting objects such as vegetables and fruits to facilitate the development of AI chefs. Simultaneously, the research also uses discovery support libraries such as Dlib and face_recognition to recognize and identify targets for grading systems, student catalogs in schools, dot systems for employees, or security window opening systems on the farm.

The source code for this research will be pushed onto the Github page with the desire to build a system that integrates more functionality, such as image processing, color detection, QR code detection, and frame rate increases that are impossible with the current system. The AI has been on the ground for a while now. The system will thus be contributed source code by the community to create stability for the system and regularly updated to fix when the system has an issue, error, or other issues that are not working.

LỜI CAM ĐOAN

Tôi xin cam đoan nghiên cứu luận văn tốt nghiệp "Xây dựng công cụ phát triển ứng dụng cho cảm biến máy ảnh tự động" được hoàn thành trên kết quả nghiên cứu của tôi với sự hướng dẫn của Thầy Lâm Hoài Bảo. Ngoài các trích dẫn, tài liệu tham khảo cũng như các số liệu, kết quả nêu trong luận văn là trung thực và các kết quả của nghiên cứu này chưa được dùng cho bất cứ luận văn cùng cấp nào khác. Nếu không đúng như đã nêu trên, tôi xin hoàn toàn chịu trách nhiệm về đề tài của mình dưới mọi hình thức.

Cần Thơ, ngày 03 tháng 02 năm 2022

Người hướng dẫn	**Tác giả thực hiện**
(ký tên)	*(ký tên)*
Lâm Hoài Bảo	**Cổ Bảo Hiếu**

MỤC LỤC

DANH MỤC BẢNG

DANH SÁCH HÌNH

DANH MỤC TỪ VIẾT TẮT

STT	TỪ VIẾT TẮT	TIẾNG ANH	TIẾNG VIỆT
1	1D	One dimensionality	Một chiều
2	2D	Two dimensionality	Hai chiều
3	3D	Three dimensionality	Ba chiều
4	AI	Artificial Intelligent	Trí tuệ nhân tạo
5	API	Application Programming Interface	Phương thức trung gian kết nối
6	AVC	Advance Video Coding	Mã hóa video cấp cao
7	BoF	Bag of Freebies	
8	BoS	Bag of Specials	
9	CBAM	Convolution Block Attention Module	
10	CBN	Cross-Iteration Batch Normalization	
11	CCD	Charge-Coupled Device	
12	CIoU-loss	Complete-IoU Loss	
13	CmBN	Cross mini-Batch Normalization	
14	CMOS	Complementary Metal-Oxide-Semiconductor	
15	CNN, CNNs	Convolutional Neural Network	Mạng nơ ron tích chập
16	CONV	Convolution	Tích chập
17	CRT	Cathode ray tube	Màn hình hoạt động theo nguyên lý ống phóng chùm điện tử
18	CSI	Camera Serial Interface	
19	CSP	Cross-Stage-Partial connections	
20	DL	Deep Learning	Học sâu
21	DIoU	Distance-IoU	
22	FC	Fully Connected	
23	FF		Bộ lọc
24	GioU	Generalized Intersection over Union	
25	GPU	Graphics Processing Unit	
26	H.264	MPEG-4 Part 10 AVC	Chuẩn nén H.264
27	H.265	H.265/HEVC	Chuẩn nén H.265
28	HEVC	High Efficiency Video Coding	Mã hóa video hiệu quả cao

29	HOG	Histogram of Oriented Gradients	
30	IOU	Intersection over Union	Chỉ số đánh giá
31	IR	Infrared	Hồng ngoại
32	KNN	K-nearest neighbors	
33	LCD	Liquid Crystal Display	Màn hình tinh thể lỏng
34	MIPI	Mobile Industry Processor Interface	
35	MTCNN	Multi-task Cascaded Convolutional Networks	Mạng đa năng xếp tầng đa tác vụ
36	OO		Đầu ra
37	PP-YOLOE	Paddle Paddle YOLOE	
38	QR	Quick Response	Phản hồi nhanh
39	RCNN, R-CNN	Region-based Convolutional Neural Networks	
40	ROI	Region of Interest	
41	RoIAlign	Region of interest feature alignment	
42	RPN	Region Proposal Network	
43	SDK	Software Development Kit	Các công cụ và phần mềm dùng để phát triển ứng dụng thông qua một nền tảng nhất định
44	SS		Độ trượt
45	SSD	Single Sort Detection	
46	SVM	Support Vector Machine	Máy vectơ hỗ trợ
47	TLS	Transport Layer Security	Giao thức mật mã
48	UI	User Interface	Giao diện người dùng
49	V2K		Vắc xin, khử khuẩn, khẩu trang
50	YOLO	You Only Look Once	

CHƯƠNG 1. GIỚI THIỆU

1.1 Giới thiệu và lý do chọn đề tài nghiên cứu

Cuộc sống của con người từ thời xa xưa cho đến hiện tại đều cần có những công cụ để giúp cho con người có thể thực hiện được các công việc hằng ngày trở nên dễ dàng, dễ dùng và cũng như dễ tiếp cận đến thị hiếu của mọi người với mức giá hợp lý để có thể phù hợp với túi tiền của nhà nhà, người người. Để làm cho điều đó ngày càng dễ dàng, con người luôn tìm cách học hỏi và không ngừng nâng cấp các công cụ đó dựa trên các kiến thức được tích trữ trong hàng nghìn năm qua.

Đến thời điểm hiện tại, thì công nghệ mới nhất có thể kể đến đó là sự ra đời của công nghệ 4.0. Công nghệ 4.0 ra đời đã tạo ra một cuộc sống cho con người ngày càng hiện đại, tiện lợi. Dù là người lớn hay trẻ nhỏ thì đều có thể dễ dàng sử dụng mà không cần phải thông qua các trường lớp đào tạo nào cả. Công nghệ 4.0 với đầy ấp sự hữu dụng đã làm cho khoa học tiến bộ không ngừng và vượt bậc đến nỗi tất cả mọi thứ trong phim viễn tưởng đều có thể xuất hiện ở ngoài đời thực.

Để có thể chế tạo ra các công cụ hữu ích thì cần phải tạo ra các thiết bị phần cứng và phần mềm tối ưu và hoàn thiện nhất. Dù là phần mềm hay phần cứng thì vai trò nào cũng quan trọng cả. Nhưng để đáp ứng tức thì các phản hồi của các khách hàng khi sử dụng, qua đó sửa lỗi hoạt động của các sản phẩm thì phần mềm luôn đóng vai trò đặc biệt quan trọng trong việc điều khiển chính xác các bo mạch hay các linh kiện bên trong để chúng ít hỏng hốc và tăng tuổi thọ của chúng lên mức lâu nhất có thể. Vì điều đó, phần mềm nhúng đã ra đời và có thể thực hiện những điều dường như không tưởng để biến chúng thành hiện thực chân thực.

Đi kèm với công nghệ hiện đại trong thời buổi dịch bệnh Covid-19 thì không thể không kể đến các thiết bị có phần mềm nhận dạng khuôn mặt, nhận dạng các vị khách hàng không mang khẩu trang hoặc có trang bị khẩu trang theo tiêu chuẩn V2K của Bộ Y tế khuyến cáo để có thể kịp thời phát hiện và nhắc nhở mọi người thực hiện đúng với Thông điệp V2K ưtrong việc chung sống an toàn với dịch bênh Covid-19. Nghiên cứu "Xây dựng công cụ phát triển ứng dụng cho cảm biến máy ảnh tự động" đã thiết kế mô hình nhận dạng khuôn mặt chứa tên của người dùng, mô hình nhận dạng những người không mang khẩu trang hoặc có mang khẩu trang, ngoài ra còn có mô hình nhận dạng rau củ để có thể tạo ra những món thức ăn đầy đủ dinh dưỡng trong mùa dịch. Công cụ cũng có những chức năng cơ bản như một ứng dụng chụp ảnh của một thiết bị ghi hình kỹ thuật số, máy ảnh kỹ thuật số được kết nối với máy vi tính.

Nghiên cứu này đã mang đến một phần mềm có thể chạy trên những mô hình hoạt động với kinh phí thấp mà vẫn có thể đạt được hiệu quả như mong đợi và có thể hữu dụng trong cuộc sống hiện tại trong văn phòng hay trong hộ gia đình. Công cụ này được nghiên cứu như một thiết bị định danh khuôn mặt an toàn ở cửa để có thể mở cửa khi gia chủ về đến nhà hay có thể tích hợp vào những công nghệ mới như nhà thông minh

(smart home), khóa cửa thông minh, hệ thống điểm danh các nhân viên, học sinh, sinh viên khi đến trường hay cơ quan cũng như phát hiện người nào có đeo khẩu trang hay không mang khẩu trang. Công nghệ này cũng có thể tích hợp vào việc nhận dạng các thiết bị rau củ quả, phân loại rau củ quả cho các mô hình trang trại, siêu thị hoặc hỗ trợ nấu ăn trong chính gian bếp của các đầu bếp tại gia. Công cụ có thể tích hợp thêm gợi ý món ăn để cho ra một quyển menu biết tuốt và có thể nói được cũng như không có cảm xúc. Nghiên cứu cũng giúp tạo ra những tấm hình xinh xắn làm kỉ niệm với các bộ lọc khác nhau hoặc có thể trở thành một camera giám sát ngay chính trong căn nhà hay bất cứ nơi nào cần quan sát.

1.2 Các nghiên cứu liên quan

Mở khóa điện thoại bằng khuôn mặt hay mở khóa máy tính cá nhân bằng khuôn mặt, hay chấm công bằng khuôn mặt đã không có gì đặc biệt trong kỷ nguyên số ngày nay.

Đã có rất nhiều bài báo liên quan đến nhận dạng khuôn mặt hay phát hiện đối tượng được xuất bản, thế nhưng những bài báo liên quan đến việc phát triển camera trên thiết bị NVIDIA Jetson Nano thì chỉ mới được công bố nhiều hơn trong 2 năm trở lại đây. Đại dịch đặc biệt COVID19 bùng phát đã tạo ra nhiều cơ hội cho giới nghiên cứu về thiết bị nhúng, cũng nhờ điều đó mà những bài báo phát hành liên quan các trí tuệ nhân tạo ngày càng nhiều.

Trước khi nghiên cứu được thực hiện, thì đã có rất nhiều những nghiên cứu xoay quanh giải quyết các vấn đề từ cơ bản đến nâng cao về các chức năng của máy ảnh và tích hợp các AI vào trong xử lý ảnh. Những nghiên cứu AI tiền đề về nhận dạng người mang khẩu trang hay không mang khẩu trang sử dụng giải thuật YOLO [1], xây dựng hệ thống nhận dạng khuôn mặt với giá 60$ với NVIDIA Jetson Nano 2GB với ngôn ngữ Python [2] làm tiền đề cho nghiên cứu bài luận văn này. Để luận văn này được hoàn thành thì nghiên cứu cũng đã tham khảo cho các bài báo sau:

- Bài báo với nội dung là Đánh giá hiệu suất và so sánh phần mềm nhận dạng khuôn mặt, dựa trên thư viện Dlib và Opencv được công bố vào năm 2018 của các tác giả như: Nataliya Boyko, Oleg Basystiuk, Nataliya Shakhovska [3].
- Bài báo với nội dung là Phân tích video thời gian thực để phát hiện đối tượng và nhận dạng khuôn mặt sử dụng học sâu được công bố vào năm 2019 của các tác giả như: Shrikant Jagannath Patro, Prof. Nisha V M [4].
- Bài báo với nội dung là Hiệu quả của hệ thống nhận dạng khuôn mặt với kỹ thuật học tập phát hiện khuôn mặt người cho sinh viên được công bố vào năm 2020 của tác giả Md. Rakibul Hasan [5].

- Bài báo với nội dung là Hệ thống chấm công và nhận dạng khuôn mặt sử dụng thư viện Dlib và face_recognition được công bố vào năm 2021 của tác giả Shashank Reddy Boyapally [6].
- Bài báo với nội dung là Thiết kế và triển khai hệ thống kiểm soát truy cập của khách truy cập sử dụng học sâu nhận dạng khuôn mặt được công bố vào năm 2021 của các tác giả như: Seok-Yeol Heo, Kang Min Kim và Wan-Jik Lee [7].
- Bài báo với nội dung là thuật toán phát hiện biển báo giao thông Trung Quốc theo thời gian thực dựa trên YOLOv2 đã được chỉnh sửa được công bố vào năm 2017 của các tác giả như: Jianming Zhang, Manting Huang, Xiaokang Jin và Xudong Li [8]. Nghiên cứu sử dụng YOLOv2.
- Bài báo với nội dung là YOLO-LITE: Thuật toán phát hiện đối tượng theo thời gian thực được tối ưu hóa cho máy tính không có GPU được công bố vào năm 2018 của các tác giả như: Rachel Huang, Jonathan Pedoeem và Cuixian Chen [9]. Nghiên cứu sử dụng YOLO-LITE.
- Bài báo với nội dung là phát triển các hệ thống AI để đếm số lượng khách truy cập và kiểm tra việc mang khẩu trang bằng các thuật toán học sâu được công bố vào năm 2020 của các tác giả như: 조원영O, 박승렬, 김현수, 윤태진 [10]. Nghiên cứu sử dụng YOLOv4, YOLOv3, YOLO-Tiny.
- Bài báo với nội dung là thuật toán phát hiện đeo mặt nạ dựa trên YOLO-v4 được cải tiến được công bố vào năm 2021 của các tác giả như: Jimin Yu và Wei Zhang [11]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là phát hiện lỗi học sâu tại chỗ trong thời gian thực để kiểm tra UAV tự hành được công bố vào năm 2021 của các tác giả như: Naeem Ayoub và Peter Schneider-Kamp [12]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là SORT-YM: Thuật toán theo dõi đa đối tượng với YOLOv4-Tiny và dự đoán chuyển động được công bố vào năm 2022 của các tác giả như: Han Wu, Chenjie Du, Zhongping Ji, Mingyu Gao và Zhiwei He [13]. Nghiên cứu sử dụng YOLOv4-Tiny.
- Bài báo với nội dung là triển khai hệ thống đo lường khoảng cách xã hội và phát hiện người theo thời gian thực, dựa trên AI cho Covid-19 được công bố vào năm 2021 của các tác giả như: Sergio Saponara, Abdussalam Elhanashi và Alessio Gagliardi [14]. Nghiên cứu sử dụng YOLOv2.
- Bài báo với nội dung là sử dụng mạng học sâu YOLOv4 được cải tiến để phát hiện chính xác ruồi trắng và bọ trĩ trên hình ảnh bẫy dính được công bố vào năm 2021 của các tác giả như: Dujin Wang, Yizhong Wang, Ming Li,

Xinting Yang, Jianwei Wu và Wenyong Li [15]. Nghiên cứu sử dụng YOLOv4.

- Bài báo với nội dung là phát triển mô hình phát hiện đối tượng nén dựa trên YOLOv4 để triển khai trên nền tảng GPU nhúng của hệ thống tự quản lý được công bố vào năm 2021 của các tác giả như: Issac Sim, Ju-Hyung Lim, Young-Wan Jang, JiHwan You, SeonTaek Oh và Young-Keun Kim [16]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là ETL-YOLO v4: Một thuật toán phát hiện mặt nạ trong thời đại đại dịch COVID-19 được công bố vào năm 2022 của các tác giả như: Akhil Kumar, Arvind Kalia và Aayushi Kalia [1]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là đánh giá hiệu suất khung học sâu sử dụng YOLO với nền tảng Nvidia Jetson được công bố vào năm 2022 của các tác giả như: Dong-Jin Shin và Jeong-Joon Kim [17]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là phân tích điểm chuẩn của Hiệu suất YOLO trên các thiết bị Edge Intelligence.được công bố vào năm 2022 của các tác giả như: Haogang Feng, Gaoze Mu, Shida Zhong, Peichang Zhang và Tao Yuan [18]. Nghiên cứu sử dụng YOLOv3, YOLOv3-Tiny, YOLOv4, YOLOv4-Tiny.
- Bài báo với nội dung là hệ thống robot tự động dựa trên nhận dạng bảng tên phòng sử dụng phương pháp YOLOv4 trên Jetson Nano 2GB được công bố vào năm 2022 của các tác giả như: Muhammad Pandu Dwi Cahyo, Fitri Utaminingrum [19]. Nghiên cứu sử dụng YOLOv4.
- Bài báo với nội dung là phát triển hệ thống phát hiện khoảng cách xã hội theo thời gian thực dựa trên chế độ xem YOLOv4-Tiny và mắt chim cho COVID-19 được công bố vào năm 2022 của các tác giả như: Sergio Saponara, Abdussalam Elhanashi, Qinghe Zheng [20]. Nghiên cứu sử dụng YOLOv4-Tiny.
- Bài báo với nội dung là xây dựng hệ thống quản lý và giám sát đám đông theo thời gian thực để phân loại khoảng cách xã hội và chăm sóc sức khỏe bằng cách sử dụng học sâu được công bố vào năm 2022 của các tác giả như: Sangeeta Yadav, Preeti Gulia, Nasib Singh Gill và Jyotir Moy Chatterjee [21]. Nghiên cứu sử dụng YOLOv4.

1.3 So sánh các nghiên cứu

Các nghiên cứu phát hiện đối tượng tham khảo ở trên đều có những nghiên cứu thành công về vấn đề tối ưu hiệu suất, phát hiện các khuôn mặt có trang bị khẩu trang khi ra đường hay có giữ khoảng cách trong dịch bệnh COVID19 hay không hoặc nhận dạng các rau củ quả. Tính đến thời điểm hiện tại thì vẫn chưa có nghiên cứu nào dùng ứng dụng công nghệ NVIDIA DeepStream vào nghiên cứu. Mặt khác, các nghiên cứu

nhận dạng khuôn mặt trên chỉ sử dụng các thư viện Dlib hay thư viện face_recognition cùng với OpenCV chỉ với mục đích nhận dạng khuôn mặt người và định danh khuôn mặt đó. Vẫn chưa có nghiên cứu nào tối ưu các khung hình về tốc độ truy xuất của nghiên cứu nhận diện khuôn mặt. Và cuối cùng là để quy tụ lại các chức năng trong một ứng dụng là điều chưa có nghiên cứu nào làm cho đến thời điểm hiện tại.

1.4. Mục tiêu nghiên cứu đề tài

Việc trang bị các thiết bị tích hợp trí tuệ nhân tạo vào trong cơ quan, ngôi nhà của mọi người để tránh những sự cố nguy cơ lây lan từ dịch bệnh và giá cả phải chăng là điều rất cần thiết cho tất cả mọi người trong thời điểm dịch bệnh COVID-19 đã phá hoại tất cả các khái niệm về cuộc sống như trước đây. Nó cũng đã làm cho các công nghệ không ngừng phát triển để thay thế dần con người vào những nơi nguy hiểm cũng như đặt tính mạng con người lên trên. Với các ca nhiễm ngày càng tăng, những biến chủng COVID-19 ngày càng tinh vi và cướp đi rất nhiều sinh mạng của con người do nhiều nguyên nhân khác nhau.

Nghiên cứu hướng tới việc xây dựng một công cụ điều khiển máy ảnh với phần cứng được xây dựng trên một thiết bị nhỏ gọn và được viết với ngôn ngữ mã nguồn mở. Qua đó, cung cấp cho mọi người một phần mềm hoàn toàn miễn phí với nhiều chức năng cao cấp, có thể nâng cấp hoặc thêm các chức năng mong muốn vào phần mềm hay có thể thương mại hóa sản phẩm. Bên cạnh đó, phần mềm có sử dụng các thuật toán, mô hình máy học để ứng dụng vào thực tế cuộc sống như:

Phát hiện phát hiện người có mang khẩu trang hay không mang khẩu trang và phát ra cảnh báo bằng nhóm các mô hình về YOLO ở thời gian thực thay vì sử dụng các mô hình R-CNN (như R-CNN, Fast R-CNN, Faster-RCNN hay Mask R-CNN).

Nhận dạng, phân loại các loại rau củ quả cho đầu bếp AI sử dụng . Đầu bếp AI sẽ quét các rau củ quả trong nhà và gợi ý các món ăn dinh dưỡng thuần chay.

Công cụ điểm danh cho học sinh, sinh viên hay hệ thống chấm công cho công nhân viên hoặc những hệ thống cửa thông minh tự động mở cho những nông trại, trang trại sử dụng thư viện face_recognition và thư viện Dlib thay vì mô hình MTCNN.

Giải mã mã QR, tăng tốc độ khung hình cho camera hay chụp ảnh, quay video với một vài bộ lọc màu sắc bằng usb camera Logitech C922 trên phần cứng thiết bị NVIDIA Jetson Nano. Nghiên cứu tạo ra chương trình có thể chạy trên nhiều nền tảng khác nhau.

1.5 Đối tượng nghiên cứu và phạm vi nghiên cứu

Nghiên cứu sử dụng các tập dữ liệu:

- Phát hiện đối tượng được chia dữ liệu ngẫu nhiên ra hai phần là huấn luyện (training) và kiểm thử (testing) như sau:
    - Tập dữ liệu người mang khẩu trang hay không mang khẩu trang: sử dụng tập dữ liệu MAFA và Wider Face [22], chứa 7971 ảnh với hai nhãn là

người có mang và không mang khẩu trang. Tập dữ liệu được gán nhãn với định dạng Pascal VOC xml.

- Tập dữ liệu rau củ quả với khoảng 693 ảnh, chứa 7 loại rau củ quả khác nhau và sử dụng 50% tập dữ liệu trên internet [23] và 50% tập dữ liệu được tạo ra trong nghiên cứu. Tập dữ liệu được gán nhãn với định dạng YOLO txt

- Nhận dạng khuôn mặt: với tập dữ liệu khoảng 180 ảnh chứa 6 mặt người khác nhau và 100% tập dữ liệu được tạo ra trong nghiên cứu. Dữ liệu chứa hình ảnh mỗi người nên có ít nhất từ 10 đến 20 hình ảnh chính diện. Dữ liệu này không cần gán nhãn mà chỉ cần tạo một thư mục đặt tên giống với tên tương ứng với nhãn, sau đó cho tất cả hình ảnh của người đó vào một thư mục.

1.6 Phương pháp nghiên cứu

Tìm hiểu tài liệu, kiến thức về các mô hình học sâu về phát hiện đối tượng, nhận dạng khuôn mặt, thư viện quản lý camera, nhận dạng mã vạch, phát hiện màu sắc, tăng tốc độ khung hình, v.v... Ngôn ngữ lập trình sử dụng là Python, giao diện người dùng PyQt.

Huấn luyện các mô hình, xây dựng, cài đặt giải thuật.

Nghiên cứu này áp dụng các mô hình sau:

- Mô hình thuật toán YOLO [24] và đặc biệt với kiến trúc YOLOV4-Tiny kết hợp với NVIDIA DeepStream vào bài toán phát hiện đối tượng.
- Mô hình thư viện dlib và thư viện face_recognition vào bài toán nhận dạng khuôn mặt.

Thực nghiệm trên các tập dữ liệu.

Kiểm tra và đánh giá kết quả thực nghiệm của các mô hình đề xuất.

1.7 Những đóng góp chính của đề tài

Đề tài có một số điểm mới sau đây:

- Đề xuất ngôn ngữ Python trong việc viết ứng dụng.
- Viết lại phần mềm quản lý cấu hình của máy ảnh theo ý riêng của tác giả từ giao diện đến các tính năng từ cơ bản đến năng cao.
- Nghiên cứu cũng cung cấp các chức năng quản lý camera thông thường như lưu trữ hình ảnh hay các đoạn phim với các FPS khác nhau.
- Đề xuất nghiên cứu cấu hình thêm một số tính năng tích hợp các giải thuật về việc xử lý các chứng năng liên quan đến AI, ML, DL như YOLOV4 [25], Dlib [26], face_recognition [27], OpenCV để phát hiện đối tượng hay nhận dạng khuôn mặt, phát hiện màu sắc, quét mã QR code hàng loạt.
- Công cụ điều khiển máy ảnh hiệu quả hơn, có thể dễ dàng thay đổi máy ảnh khác nhau mà không cần phải thay đổi cấu hình phần mềm quá nhiều.

- Khởi đầu của quy trình làm một máy ảnh được tối ưu dựa trên phầm mềm mã nguồn mới với giấy phép mở.
- Cung cấp một cái nhìn khác về việc xử lý tín hiệu tại chỗ với thời gian thực mà không cần phải tốn bộ nhớ để lưu trữ hay quan tâm đến độ trễ khi xử lý tín hiệu tại chỗ.
- Giải quyết được bài toán giảm tải lưu trữ và giảm thời gian hồi đáp, giải quyết được giá thành cũng như tốc độ xử lý ở thời gian thực, có thể cắm vào (plug-in) hay rút ra (plug-out) các camera để có thể mang đến hiệu suất cao nhất và hiệu quả nhất.
- Những đóng góp của mã nguồn phần mềm hay tài liệu đã tham khảo qua đều được tải lên ở trang: https://github.com/cobaohieu/thesis/.

1.8 Bố cục quyển luận văn

Luận văn có 5 chương:

Chương 1: Giới thiệu.

Chương 2: Cơ sở lý thuyết

Chương 3: Phương pháp nghiên cứu.

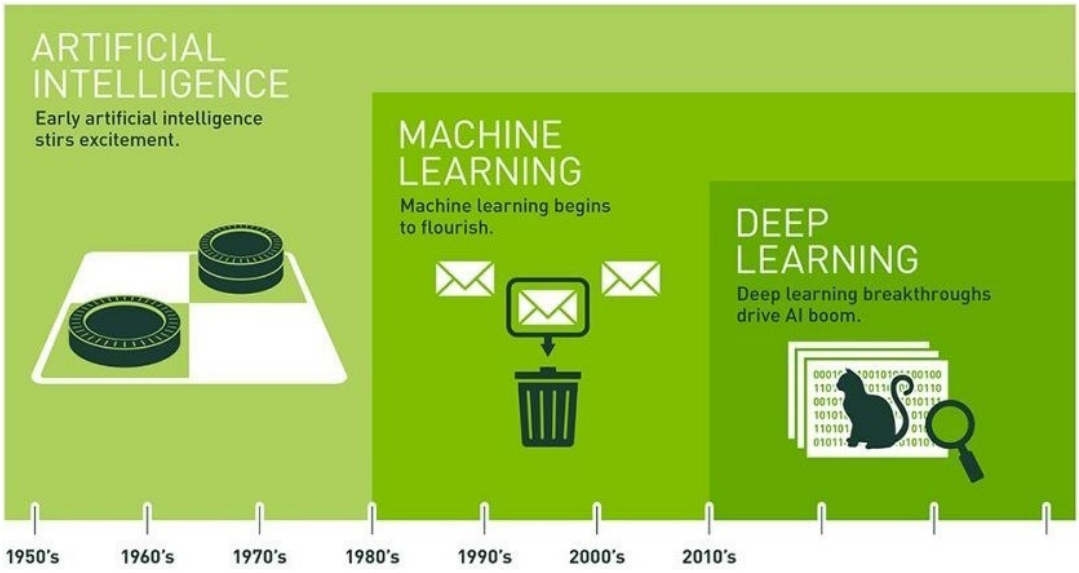
Chương 4: Kết quả thực nghiệm và đánh giá.

Chương 5: Kết luận và hướng phát triển.

CHƯƠNG 2. CƠ SỞ LÝ THUYẾT

2.1 Một số khái niệm

2.1.1 Trí tuệ nhân tạo (Artificial Intelligent - AI)



Hình 2.1: Mối quan hệ của trí tuệ nhân tạo với máy học và học sâu [28].

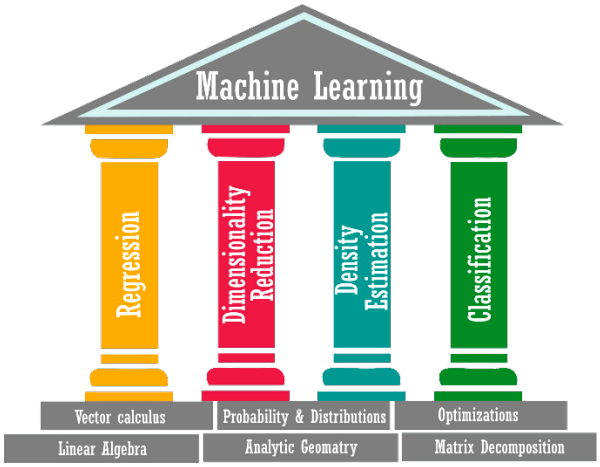
Trí tuệ nhân tạo (Artificial Intelligent) là một ngành thuộc lĩnh vực khoa học máy tính.

Trí tuệ nhân tạo là trí tuệ do con người lập trình tạo ra với mục đích giúp máy tính có thể tự động hóa các hành vi thông minh như con người. Trí tuệ nhân tạo khác với việc lập trình logic trong các ngôn ngữ lập trình là việc ứng dụng các hệ thống máy học để mô phỏng trí tuệ của con người trong các xử lý mà con người làm tốt hơn máy tính. Cụ thể, trí tuệ nhân tạo giúp máy tính có được những trí tuệ của con người như: biết suy nghĩ và lập luận để giải quyết vấn đề, biết giao tiếp do hiểu được các ngôn ngữ, tiếng nói, biết học và tự thích nghi.

Sau đây là một vài ứng dụng kỹ thuật hiện hành đang phát triển nhanh chóng của trí tuệ nhân tạo:

- Nhận dạng, phân loại, gắn thẻ hình ảnh tĩnh.
- Cải thiện hiệu suất chiến lược thương mại theo thuật toán.
- Quy trình xử lý dữ liệu bệnh nhân hiệu quả và có khả năng nhân rộng.
- Dự tính thời điểm bảo trì.
- Phát hiện và phân loại vật thể.
- Phân bố nội dung trên các phương tiện truyền thông xã hội.
- Bảo vệ khỏi những mối đe dọa an ninh mạng.

2.1.2. Học máy hoặc máy học (Machine Learning)



Hình 2.2: Ngôi nhà với máy nhà là máy học [29].

2.1.2.1 Định nghĩa

Học máy hoặc máy học (Machine Learning) nổi lên như một bằng chứng của cuộc cách mạng công nghiệp lần thứ 4 (với các cuộc cách mạng lần lượt như thứ 1 - động cơ hơi nước, thứ 2 - năng lượng điện, thứ 3 - công nghệ thông tin). Trí Tuệ Nhân Tạo đang len lỏi vào mọi lĩnh vực trong đời sống hằng ngày. Xe tự hành của Google và Tesla, hệ thống tự tag khuôn mặt trong ảnh của Facebook, trợ lý ảo Siri của Apple, hệ thống gợi ý sản phẩm của Amazon, hệ thống gợi ý phim của Netflix, máy chơi cờ vây AlphaGo của Google DeepMind, v.v…, chỉ là một vài trong vô vàn những ứng dụng của trí tuệ nhân tạo/học máy.

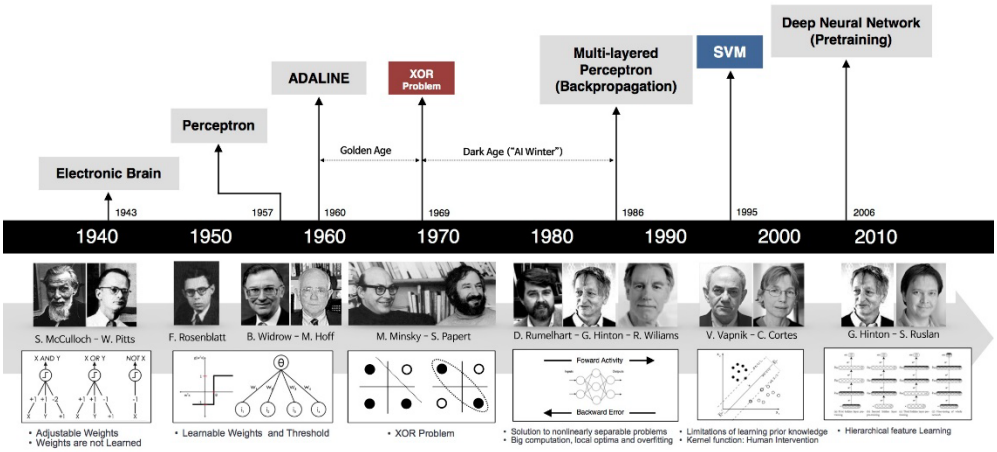
2.1.2.2 Các phương pháp học máy

Học máy là một tập con của trí tuệ nhân tạo và có nhiều loại: học có giám sát (Supervised Learning), học không giám sát (Unsupervised Learning), học bán giám sát (Semi-Supervised Learning), học củng cố (Reinforcement Learning).

Học có giám sát (Supervised Learning) là thuật toán dự đoán đầu ra (outcome) của một dữ liệu mới (new input) dựa trên các cặp (input, outcome) đã biết từ trước. Cặp dữ liệu này còn được gọi là dữ liệu (data) và nhãn (label).

Học có giám sát là nhóm phổ biến nhất trong các thuật toán học máy.

2.1.3 Học sâu (Deep Learning)



Hình 2.3: Lịch sử phát triển của mạng học sâu [30].

Những năm gần đây, khi mà khả năng tính toán của các máy tính được nâng lên một tầm cao mới và lượng dữ liệu khổng lồ được thu thập bởi các hãng công nghệ lớn, học máy đã tiến thêm một bước dài và một lĩnh vực mới được ra đời gọi là học sâu (Deep Learning).

Học sâu đã giúp máy tính thực thi những việc tưởng chừng như không thể vào 10 năm trước: phân loại cả ngàn vật thể khác nhau trong các bức ảnh, tự tạo chú thích cho ảnh, bắt chước giọng nói và chữ viết của con người, giao tiếp với con người, hay thậm chí cả sáng tác văn hay âm nhạc và nhiều điều không thể tưởng khác.

2.1.4 Thị giác máy tính (Computer Vision)

Thị giác máy tính là một lĩnh vực gồm nhiều phương pháp như thu nhận ảnh, xử lý ảnh kỹ thuật số, phân tích và nhận dạng các hình ảnh. Nói chung thị giác máy tính là nhận vào dữ liệu đa chiều từ thế giới thực sau đó xử lý để cho ra các thông tin số hoặc các biểu tượng. Việc phát triển lĩnh vực này bắt nguồn từ việc sao chép các khả năng thị giác con người về sự nhận dạng và hiểu biết một hình ảnh mang tính điện tử. Việc nhận dạng hình ảnh có thể xem như việc giải quyết vấn đề của các biểu tượng hay thông tin từ dữ liệu hình ảnh qua cách dùng các mô hình được xây dựng với sự giúp đỡ của các ngành lý thuyết học, thống kê học, vật lý học và hình học. Thị giác máy tính còn được mô tả là sự tổng thể của một dải các quá trình tự động và tích hợp các phương thức thể hiện cho các nhận thức của thị giác.

Thị giác máy tính là một môn học khoa học liên quan đến lý thuyết với các hậu duệ là các hệ thống nhân tạo có trích xuất các thông tin từ những hình ảnh. Dữ liệu hình ảnh có thể nhiều dạng như: ảnh chụp (ảnh 2 chiều), chuỗi video, các cảnh từ nhiều camera, hay dữ liệu đa chiều từ máy quét y học (ảnh 3 chiều), v.v…

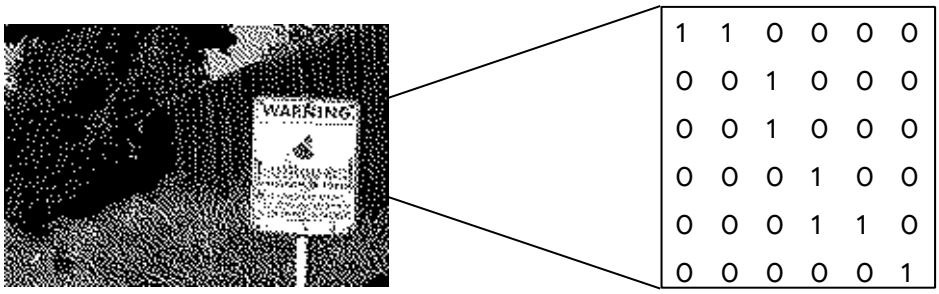
Các lĩnh vực con của thị giác máy tính gồm: tái cấu trúc cảnh, dò tìm sự kiện, theo dõi video, nhận dạng bố cục đối tượng, học, chỉ mục, đánh giá chuyển động, phục hồi ảnh, v.v…

2.1.5 Ảnh kĩ thuật số (Digital image)

Ảnh kĩ thuật số (digital image) là ảnh tĩnh. Ảnh kĩ thuật số là một dạng biểu diễn của ảnh ở dạng ma trận số hai chiều f(x, y). Tùy vào độ phân giải của ảnh có cố định hay không, ảnh kĩ thuật số được chia ra làm hai loại là ảnh véc tơ (độ phân giải không cố định) và ảnh raster (hay còn gọi là bitmapped, độ phân giải cố định). Thuật ngữ ảnh kĩ thuật số thường được dùng để nói đến ảnh raster.

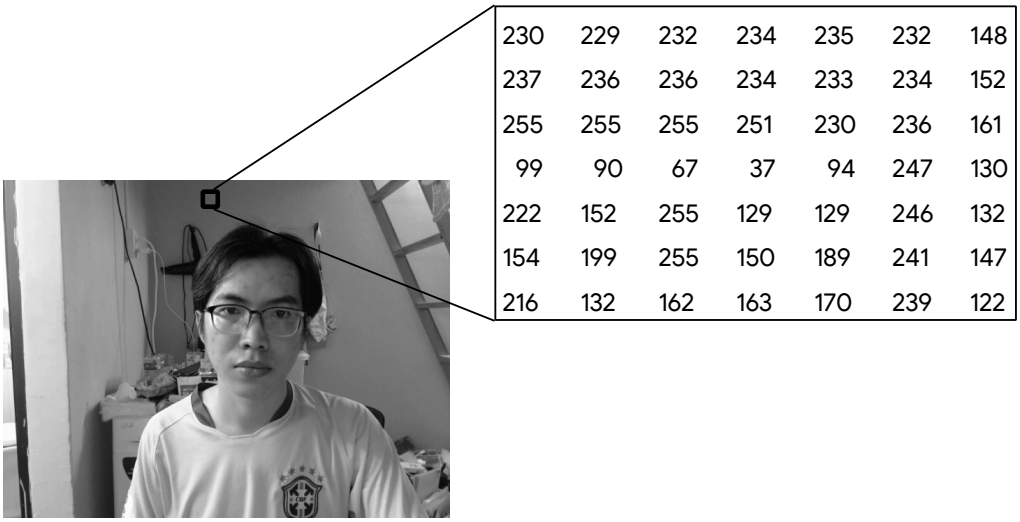
Các định dạng phổ biến của ảnh kĩ thuật số bao gồm:

- Ảnh nhị phân: 1-bit/pixel.



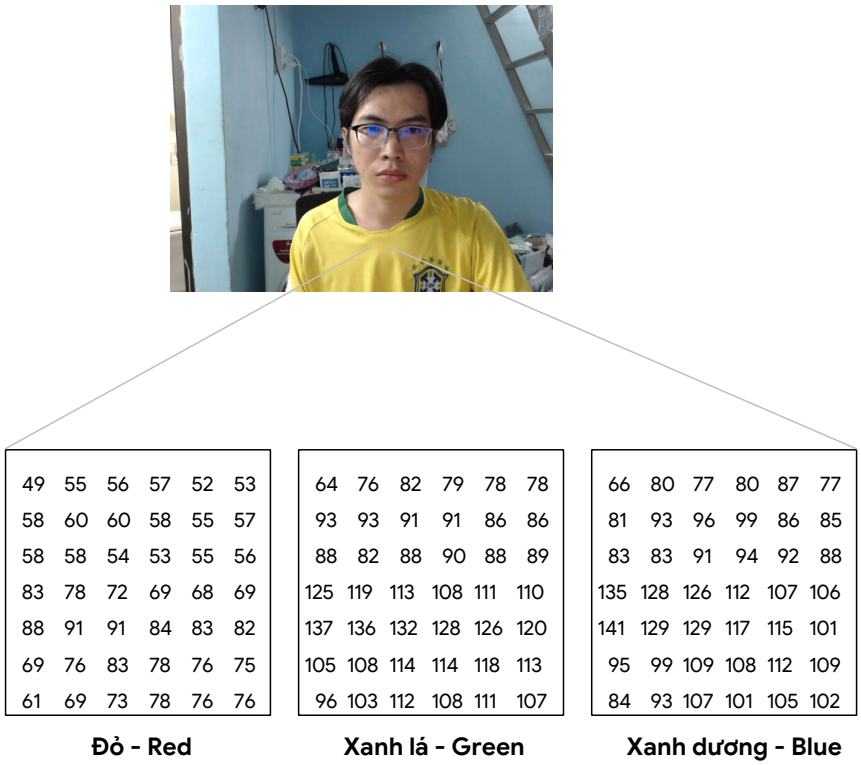
Hình 2.4: Một hình ảnh nhị phân [31].

- Ảnh xám: 8-bit/pixel.



Hình 2.5: Một hình ảnh xám.

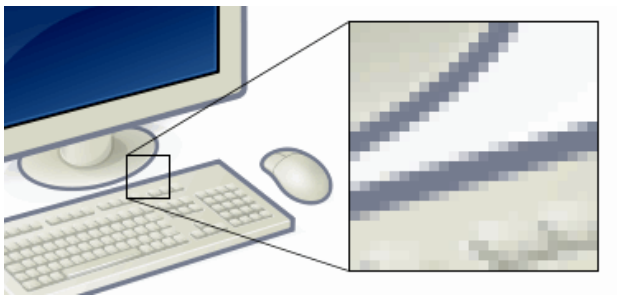
- Ảnh màu: 16-bit/pixel hay 24-bit/pixel.



Hình 2.6: Một hình ảnh màu thực sự.

2.1.5.1 Điểm ảnh (Pixel)

Trong ảnh kĩ thuật số, một điểm ảnh (pixel) là phần tử nhỏ nhất của ảnh. Mỗi một điểm ảnh là một mẫu (sample) của ảnh. Càng nhiều điểm ảnh, ảnh kĩ thuật số càng biểu diễn chính xác hơn về nội dung của ảnh gốc. Đặc trưng của một điểm ảnh gồm hai thành phần: tọa độ (x,y) và cường độ sáng (intensity).



Hình 2.7: Minh họa về các điểm ảnh riêng lẻ được hiển thị dưới dạng hình vuông nhỏ khi phóng to một hình ảnh raster trên máy tính [32].

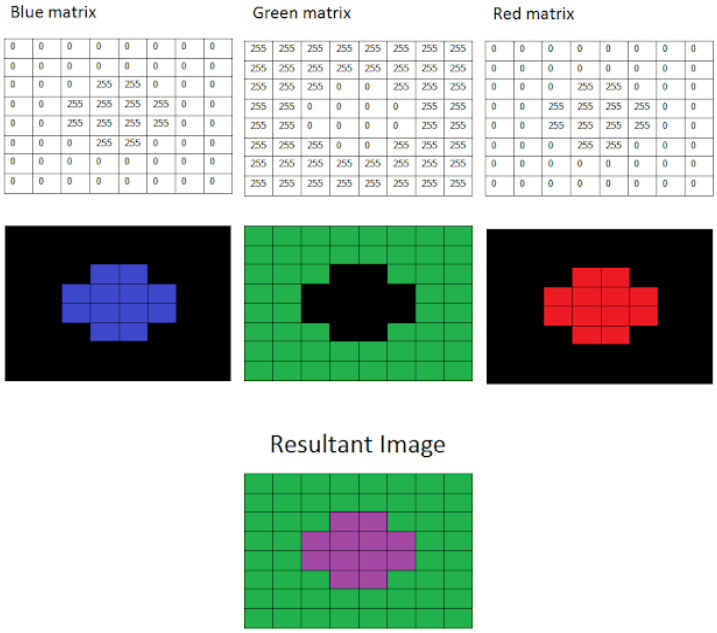
2.1.5.2 Ảnh màu

Để biểu diễn hình ảnh màu trong không gian màu RGB, ba ma trận mức xám 256, tương ứng lần lượt với ba màu sắc được sử dụng như màu đỏ (Red - R), xanh lá (Green - G), xanh dương (Blue - B). Màu sắc của một điểm ảnh được quyết định bởi giá trị cường độ (intensity) tại ba ma trận màu cùng tọa độ.

Để biểu diễn cho một điểm ảnh màu cần 24-bit. 24-bit này được chia thành ba khoảng 8-bit. Mỗi màu cũng phân thành L cấp màu khác nhau (thường L=28=256). Mỗi khoảng này biểu diễn cho cường độ sáng của một trong các màu chính.

Vì thế, để lưu trữ hình ảnh màu người ta có thể lưu trữ từng màu riêng biệt, mỗi màu lưu trữ như một hình ảnh đa cấp xám. Do đó, không gian nhớ dành cho một ảnh màu lớn gấp 3 lần một ảnh đa cấp xám cùng kích cỡ.

Trong thư viện OpenCV ảnh màu được biểu diễn ở không gian màu BGR xanh dương (Blue - B), xanh lá (Green - G) và đỏ (Red - R).



Hình 2.8: Một hình ảnh màu được biểu diễn với ba ma trận màu BGR trong OpenCV [33].

2.1.5.3 Ảnh xám

Ảnh xám hay ảnh đen trắng là hình ảnh chỉ bao gồm 2 màu: màu đen và màu trắng. Người ta phân mức đen trắng đó thành L mức. Nếu sử dụng số bit B=8 bít để mã hóa mức đen trắng (hay mức xám) thì L được xác định:

$$L=2^B \quad (2.1)$$

Nếu L=2, B=1, nghĩa là chỉ có hai mức: mức 0 và mức 1. Ảnh xám còn được gọi là ảnh nhị phân. Mức 1 ứng với màu sáng, còn mức 0 ứng với màu tối. Nếu L>2 ta có ảnh đa cấp xám.

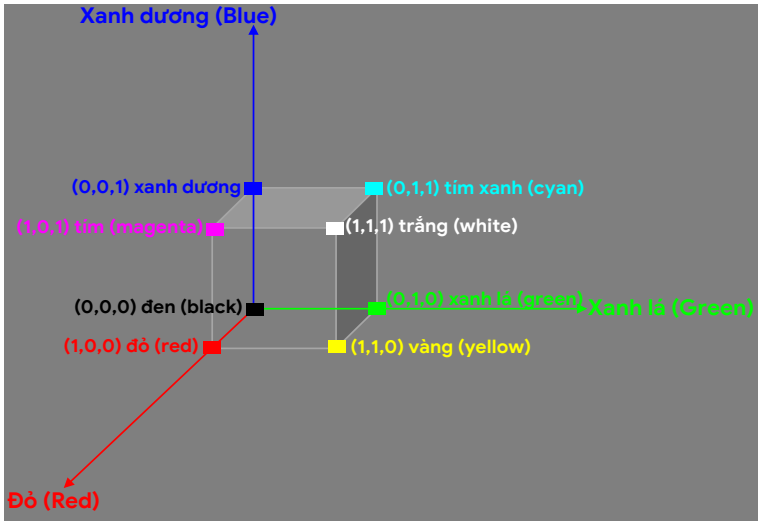
Với ảnh nhị phân, mỗi điểm ảnh được mã hóa trên 1-bit, còn với ảnh 256 mức, mỗi điểm ảnh được mã hóa trên 8-bit. Như vậy, với ảnh đen trắng: nếu dùng 8-bit (1 byte) để biểu diễn mức xám thì số các mức xám có thể biểu diễn được là 256-bit. Mỗi mức xám được biểu diễn dưới dạng là một số nguyên nằm trong khoảng từ 0 đến 255, với mức 0 biểu diễn cho mức cường độ đen nhất và 255 biểu diễn cho mức cường độ sáng nhất.

Ảnh nhị phân khá đơn giản, các phần tử ảnh có thể coi như các phần tử logic. Ứng dụng chính của nó được dùng theo tính logic để phân biệt đối tượng ảnh với nền hay để phân biệt điểm biên với điểm khác.

2.1.5.4 Thang đo mức xám hay mức xám của ảnh (Grayscale)

Mức xám của ảnh (greyscale) là một trong những giá trị số của điểm ảnh biểu diễn mức độ ánh sáng (light intensity) tại điểm ảnh đấy. Thông thường, trong xử lý ảnh hiện tại, mức xám hay sử dụng nhất là mức 256 (mức xám có giá trị từ 0 đến 255).

2.1.5.5 Hệ tọa độ màu, mô hình màu hay không gian màu RGB (Red Green Blue)



Hình 2.9: Hệ tọa độ màu RGB tương ứng với hệ tọa độ x-y-z.

Tổ chức quốc tế về chuẩn hóa màu CIE (Commission Internationale d'Eclairage) đưa ra một số chuẩn để biểu diễn màu. Các hệ này có các chuẩn riêng. Hệ chuẩn màu CIE-RGB dùng 3 màu cơ bản đỏ (Red - R), xanh lá (Green – G), xanh dương (Blue – B) và được ký hiệu là RGBCIE để phân biệt với các chuẩn khác.

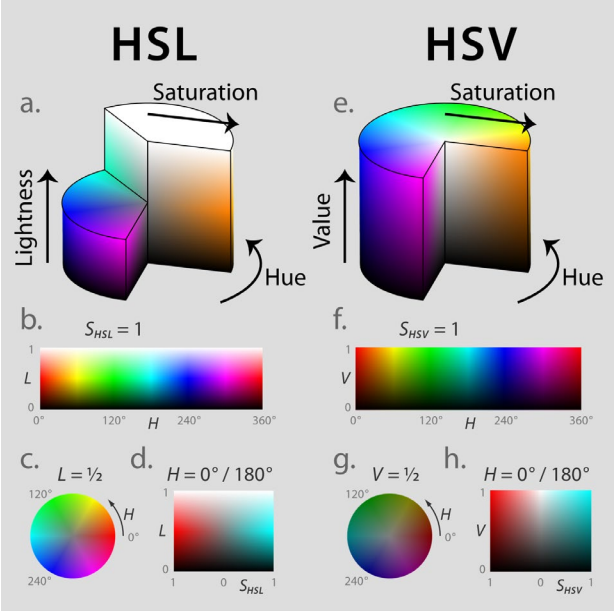
Một màu sắc trong hệ tọa độ màu là tổ hợp của các màu cơ bản theo một tỷ lệ nào đó được tính theo công thức sau:

$$P_x = [red, green, blue]^T \quad (2.2)$$

- Trong đó:
  - Px: mỗi điểm ảnh màu.
  - T: chuyển vị.

Red, Green, Blue tại ba trục nhận giá trị [0 255]. Mô hình RGB có thể biểu diễn hơn 16 triệu màu. Trong đó có giá trị là R (255, 0, 0), G có giá trị là (0, 255, 0) và B có giá trị là (0, 0, 255).

2.1.5.6 Hệ tọa độ màu, mô hình màu hay không gian màu HSV (Hue Saturation Value)



Hình 2.10: Hệ tọa độ màu HSL, HSV các màu sắc được biểu diễn ngược chiều kim đồng hồ [34].

Mô hình màu HSV như một tiêu chuẩn biểu thị màu trên Internet có nguồn gốc từ các tiêu chuẩn cho ti vi màu năm 1952 của RCA và việc sử dụng tiêu chuẩn RGB bởi Edwin Land trong các camera Land/Polaroid.

Hệ tọa độ màu HSV cũng tương tự như thệ tọa độ HSL. Hệ tọa độ màu HSV có 3 yếu tố là màu sắc (hue), độ bão hoà hay độ đậm đặc (saturation) và giá trị cường độ sáng (value). Độ bão hoà chỉ ra sắc độ màu (cường độ màu) chỉ độ đậm nhạt của màu, còn độ sáng sẽ chỉ ra độ sáng độ sáng tối.

Hệ tọa độ này thường được biểu diễn dưới dạng hình trụ hay hình nón. Các màu này sẽ được biểu diễn theo các vòng tròn ngược chiều kim đồng hồ. Vòng tròn từ 0-360 độ là trường biểu diễn cho màu sắc (hue). Trường này bắt đầu với màu đỏ (red primary), sau đó là đến màu xanh lá đầu tiên (green primary) nằm trong vùng màu từ 0-120 độ, từ 120-240 độ là màu xanh lá đến xanh dương (green primary – blue primary). Dải từ 240 – 360 là từ màu đen đến màu đỏ.

2.1.5.7 Chuyển đổi hệ tọa đồ màu BGR thành Gray

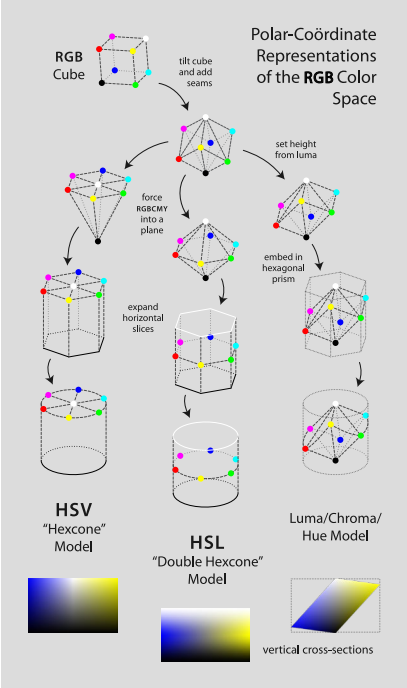
Các chuyển đổi trong không gian màu BGR thành mức xám (grayscale) hoặc ngược lại bằng cách thêm hay xóa kênh alpha, đảo ngược thứ tự kênh, chuyển đổi sang màu BGR 16-bit (B5: G6: R5 hoặc B5: G5: R5) bằng cách sử dụng:

BGR [A] đến Gray: $Y \leftarrow 0.114 \cdot B + 0.587 \cdot G + 0.299 \cdot R$ (2.3)

hoặc

Gray đến BGR [A]:B←Y,G←Y,R←Y,A←max(ChannelRange) (2.4)

2.1.5.8 Chuyển đổi hệ tọa độ màu BGR thành HSV



Hình 2.11: Chuyển đổi hệ tọa độ màu từ RGB thành HSV [34].

Trong trường hợp hình ảnh là 8-bit và 16-bit thì B, G và R được chuyển đổi sang định dạng dấu phẩy động và được chia tỷ lệ để phù hợp với phạm vi 0 đến 1.

$$V \leftarrow \max(B, G, R) \tag{2.5}$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(B, G, R)}{V} \\ 0 \end{cases} \tag{2.6}$$

- Với điều kiện V≠0, nếu không thì:

$$H \leftarrow \begin{cases} 60(G - R)/(V - \min(B, G, R)) \\ 120 + 60(R - B)/(V - \min(B, G, R)) \\ 240 + 60(B - G)/(V - \min(B, G, R)) \end{cases} \tag{2.7}$$

- Nếu H <0 thì H←H + 360. Thì đầu ra sẽ là $0 \le V \le 1$, $0 \le S \le 1$, $0 \le H \le 360$.

Sau đó, các giá trị được chuyển đổi thành kiểu dữ liệu tương ứng:

- Hình ảnh 8-bit: V← 255V, S← 255S, H← H/2 (để phù hợp từ 0 đến 255).
- Hình ảnh 16-bit: V←65535V, S←65535S, H←H.
- Hình ảnh 32-bit: H, S và V được giữ nguyên.

2.1.6 Tốc độ khung hình (frame per second – FPS)

Tốc độ khung hình (frames per second hoặc fps) là tần số (tốc độ) mà hình ảnh (khung hình) hoặc tỷ suất khung hình (frame rate) liên tiếp được chụp hoặc hiển thị trên màn hình mỗi giây. Tốc độ khung hình cũng có thể được gọi là tần số khung hình (frame frequency) và được biểu thị bằng Hertz (Hz). Tốc độ khung hình trong thông số kỹ thuật của máy ảnh điện tử có thể đề cập đến tỷ lệ tối đa có thể, trong đó, trên thực tế, các cài đặt khác (chẳng hạn như thời gian phơi sáng) có thể giảm tần số xuống một con số thấp hơn.

Một số chỉ số fps phổ biến như: 24fps, 30fps, 60fps, 120fps, 240fps, v.v…

Hình 2.12: So sánh tốc độ khung hình khác nhau [35].

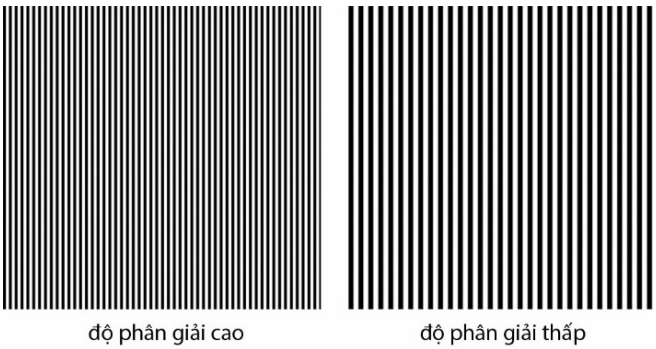
2.1.7 Độ phân giải (resolution)

Độ phân giải (resolution) của ảnh là thước đo của chi tiết rõ ràng nhỏ nhất trong ảnh, mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị (dpi). Độ phân giải càng cao thì hình ảnh càng nhiều chi tiết.

Theo định nghĩa, khoảng cách giữa các điểm ảnh phải được chọn sao cho mắt người vẫn thấy được sự liên tục của ảnh. Việc lựa chọn khoảng cách thích hợp tạo nên một mật độ phân bổ, đó chính là độ phân giải và được phân bố theo trục x và y trong không gian hai chiều, v.v...

Độ phân giải là yếu tố phụ thuộc vào khả năng của máy ảnh, phụ thuộc cảm biến ảnh (digital sensor).

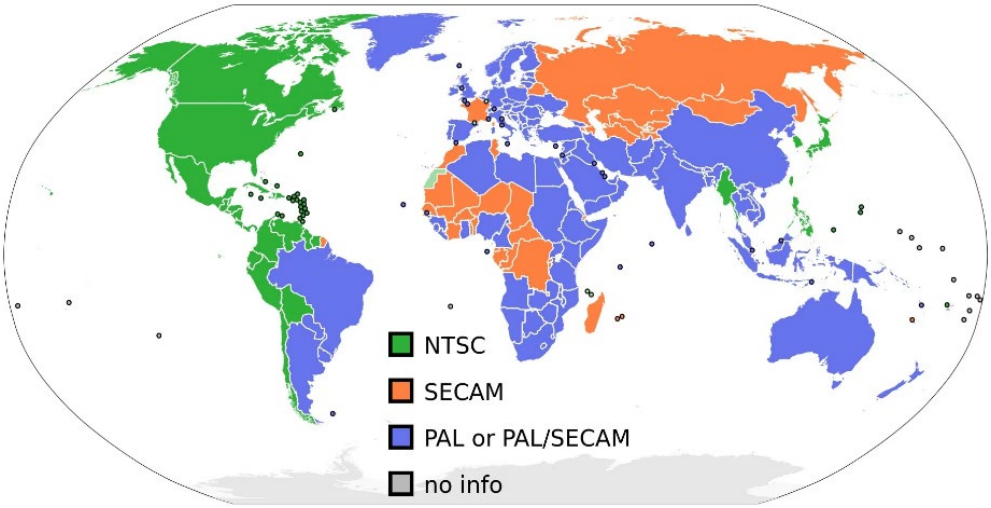
Độ phân giải thể hiện khả năng cảm biến của máy ảnh, tách bạch các phần tử gần nhau về không gian của các chi tiết.



Hình 2.13: So sánh giữa độ phân giải cao và độ phân giải thấp [36].

2.1.8 Video

Video hoặc video clip là một chuỗi các khung hình (frame) ảnh có quan hệ thời gian giữa các khung hình biểu diễn ảnh động. Video có thể được ghi lại, sao chép, phát lại hay phát sóng trực tiếp trên các phương tiện truyền thông, v.v…



Hình 2.14: Các tốc độ khung hình được tiêu chuẩn hóa bởi Hiệp hội các nhà biên tập phim điện ảnh và truyền hình (SMPTE) phổ biến trên thế giới [37].

Một vài tần số và độ phân giải phổ biến:

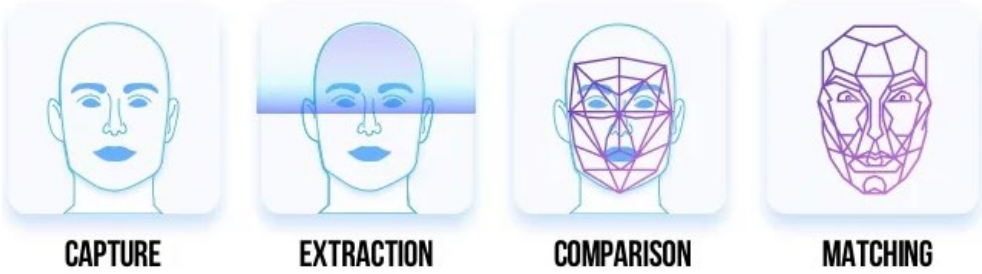
- NTSC (phổ biến ở nước Mỹ và Nhật Bản): 525 dòng (lines), 60 khung hình/giây (frame per second).
- PAL (phổ biến ở nước các Châu Âu): 625 dòng, 50 khung hình/giây.
- SECAM (phổ biến ở nước Pháp): 625 dòng, 50 khung hình/giây.
- Phim: 24 khung hình/giây.
- HDTV: với tỉ lệ khung hình 16:9, 720 dòng, 60 khung hình/giây.
- SVGA: với độ phân giải 1024x720, 72 khung hình/giây.

2.1.9 Mã phản hồi nhanh (Quick Response Code – QR Code)

Mã QR là một mã vạch ma trận (hay mã vạch hai chiều) được phát triển bởi công ty Denso Wave (Nhật Bản) vào năm 1994. Mục đích chính là theo dõi xe cộ trong quá trình sản xuất. Nó được thiết kế để cho phép quét các bộ phận với tốc độ cao.

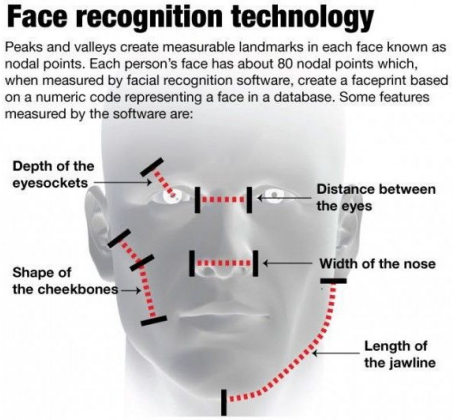
Tiêu chuẩn Nhật Bản cho các mã QR, JIS X 0510, được công bố vào tháng 1 năm 1999, và Tiêu chuẩn Quốc tế ISO tương ứng, ISO/IEC18004, được chấp thuận vào tháng 6 năm 2000.

2.1.10 Giới thiệu về nhận dạng khuôn mặt



Hình 2.15: Mô tả các định danh khuôn mặt [38].

Nhận dạng khuôn mặt (face recognition) là trích xuất các thành phần sinh học trên khuôn mặt. Các thành phần sinh học này là các đặc điểm trên khuôn mặt khác nhau ở mỗi người. Có nhiều phương pháp khác nhau trích xuất sự kết hợp khác nhau của các tính năng, thường được gọi là điểm nút. Không có hai người nào có thể có tất cả các điểm nút giống nhau ngoại trừ các cặp song sinh giống hệt nhau.

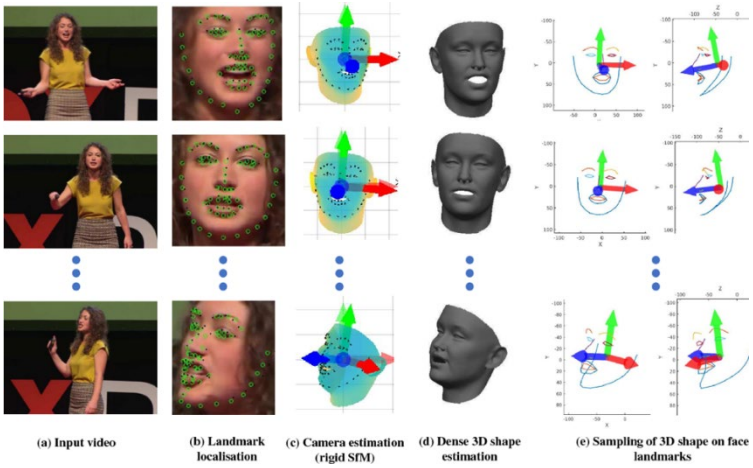


Hình 2.16: Mô tả về nhận dạng trên khuôn mặt [38].

Một số phương pháp nhận dạng khuôn mặt:

- Phương pháp nhận dạng 2 chiều (2D) hay còn gọi là phương pháp nhận dạng truyền thống. Phương pháp này xác thực khuôn mặt bằng cách trích xuất ra một điểm mốc hay cột mốc (landmark) cho khuôn mặt. Các cột mốc như là một bản đồ xác định các vị trí cố định trên khuôn mặt của một người như mắt, mũi, miệng, lông mày, râu, v.v… Phương pháp xác định các cột mốc trên khuôn mặt đã loại bỏ những phần thông tin không cần thiết và giữ lại những thông tin chính. Khi đó mỗi khuôn mặt sẽ được nén thành một véc tơ n chiều. Thông thường là 68 chiều. Phương pháp này áp dụng các thuật toán như SVM, KNN, Naive Bayes, Random Forest, Multi-layer Perceptron (MLP), … để phân loại khuôn mặt cho một người.

Hình 2.17: Các điểm ảnh trên khuôn mặt với phương pháp nhận dạng 2 chiều [39].



Hình 2.18: Mô tả về phương pháp nhận dạng 3 chiều trên khuôn mặt [40].

- Phương pháp nhận dạng 3 chiều (3D) sẽ sử dụng không gian 3 chiều để biểu diễn khuôn mặt để xác định các đặc trưng khác nhau trên bề mặt khuôn mặt như các đường viền (contour) của mắt, mũi, cằm. Lợi thế của phương pháp này là không bị ảnh hưởng bởi những thay đổi về ánh sáng như các phương pháp 2D vì thế dữ liệu độ chính xác của nhận dạng khuôn mặt này được cải thiện đáng kể. Để tạo ra một ảnh 3D, một cụm ba camera được áp dụng. Mỗi camera sẽ hướng vào một góc khác nhau. Tất cả các camera này phối hợp cùng nhau trong việc theo dõi khuôn mặt của một người trong thời gian thực và có thể nhận dạng chúng. Nhận dạng khuôn mặt trên điện thoại iPhone là nhận dạng khuôn mặt 3D. Và cần phải sẽ phải quay tròn khuôn mặt của người dùng khi xác thực nó để thuật toán học chụp lại hình ảnh các góc độ khác nhau.
- Các phương pháp nhận dạng khác: nhận dạng cảm biến da và phương pháp kết hợp. Phương pháp kết hợp có thể sử dụng nhiều thông tin đồng thời từ phương pháp cột mốc trên khuôn mặt, nhận

dạng 3D, nhận dạng cảm biến da trong các trường hợp khuôn mặt có các biểu cảm khác nhau.

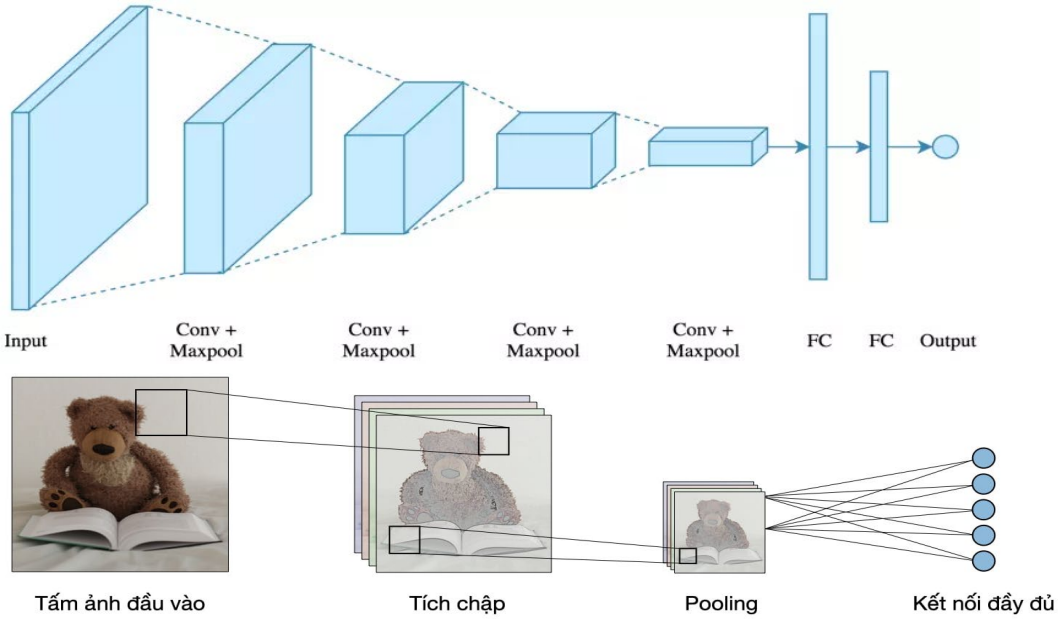
2.2 Mạng nơ ron tích chập (Convolutional Neural Network – CNN hay CNNs)

2.2.1 Giới thiệu về mạng nơ ron tích chập

Mạng tích chập neuron hay còn được biết đến với tên CNN hay CNNs, là một kiến trúc mạng nơ ron học sâu.

CNN với lịch sử ra đời khá lâu. Kiến trúc đầu tiên hay kiến trúc gốc của mô hình CNN được giới thiệu bởi một nhà khoa học máy tính người Nhật vào năm 1980 [41]. Không lâu sau đó vào năm 1998, Yan LeCun lần đầu huấn luyện mô hình CNN với thuật toán lan truyền ngược (back-propagation) cho bài toán nhận dạng chữ viết tay [42]. Tuy nhiên, mãi đến năm 2012, khi một nhà khoa học máy tính người Ukraine Alex Krizhevsky (đệ của Geoffrey Hinton) xây dựng mô hình CNN (AlexNet) và sử dụng GPU để tăng tốc quá trình huấn luyện deep nets để đạt được top 1 trong cuộc thi Computer Vision thường niên ImageNet với độ lỗi phân lớp top 5, giảm hơn 10% so với những mô hình truyền thống trước đó, đã tạo nên làn sóng mạnh mẽ sử dụng deep CNN với sự hỗ trợ của GPU. Cũng nhờ có GPU mà các bài toán đã được giải quyết ngày càng nhiều vấn đề trong thị giác máy tính.

CNN là một dạng mạng neural được cấu thành bởi các tầng sau:



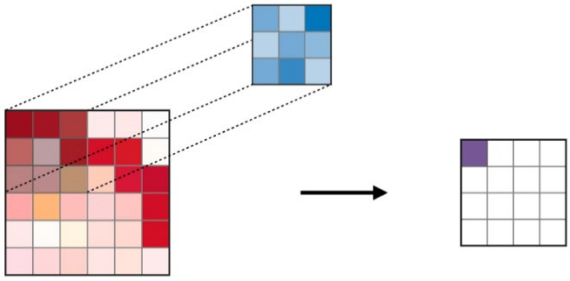
Hình 2.19: Kiến trúc của mạng CNN [43].

2.2.2 Lớp tích chập (Convolution hay CONV)

Lớp tích chập còn có 2 khái niệm khác với tên gọi là Convolution Filter và Convolutional Layer.

Trong mạng nơ ron thông thường, từ đầu vào qua các lớn ẩn (hidden layer) rồi ra được tầng đầu ra. Nâng cấp hơn với lớp tích chập, CONV sử dụng các bộ lọc

để thực hiện phép tích chập khi đưa chúng đi qua đầu vào II theo các chiều của nó. Các siêu tham số của các bộ lọc này bao gồm kích thước bộ lọc FF và độ trượt (stride) SS. Kết quả tầng đầu ra OO được gọi là bản đồ kích hoạt (activation map) hay bản đồ đặc trưng (feature map). Mỗi bản đồ đặc trưng hay bản đồ kích hoạt này là một bản quét (scan) của tầng đầu vào ban đầu, nhưng được trích xuất ra các đặc trưng (feature) cụ thể. Quét như thế nào thì lại dựa vào Convolution Filter hay kernel.



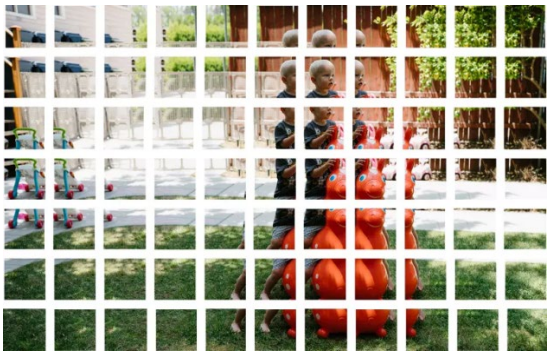
Hình 2.20: Các lớp tích chập khi trích xuất đặc trưng [43].

Lưu ý: Bước tích chập cũng có thể được khái quát hóa cả với trường hợp một chiều (1D) và ba chiều (3D).

2.2.2.1. Stride and Padding

Stride là khoảng cách giữa 2 kernel khi quét. Với stride = 1, kernel sẽ quét 2 ô ngay cạnh nhau, nhưng với stride = 2, kernel sẽ quét ô số 1 và ô số 3. Bỏ qua ô ở giữa. Điều này nhằm tránh việc lặp lại giá trị ở các ô bị quét.

Stride và kích thước (size) của kernel càng lớn thì kích thước (size) của bản đồ đặc trưng càng nhỏ, lý do đó là bởi kernel phải nằm hoàn toàn trong đầu vào. Có một cách để giữ nguyên kích cỡ của bản đồ đặc trưng so với ban đầu, đó được gọi là padding. Khi điều chỉnh padding = 1, tức là thêm 1 ô bọc xung quanh các cạnh của đầu vào, muốn phần bọc này càng dày thì cần phải tăng padding lên.



Hình 2.21: Các cửa sổ trượt khi quét qua kernel [44].

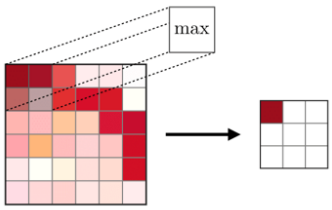
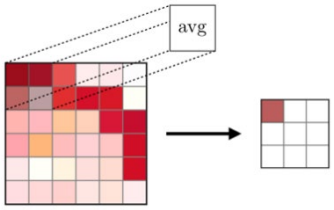
2.2.2.2. Lớp tổng hợp (Pooling layer - POOL)

Lớp tổng hợp là một phép giảm tần số mẫu (downsampling) hay làm giảm số siêu tham số (hyperparameter) cần phải tính toán, từ đó giảm được

thời gian tính toán, tránh quá khớp (overfitting), thường được sử dụng sau tầng tích chập qua đó giúp tăng tính bất biến không gian.

Cụ thể, thì max pooling và average pooling là những dạng tônge hợp đặc biệt, mà tương ứng là trong đó giá trị lớn nhất và giá trị trung bình được lấy ra.

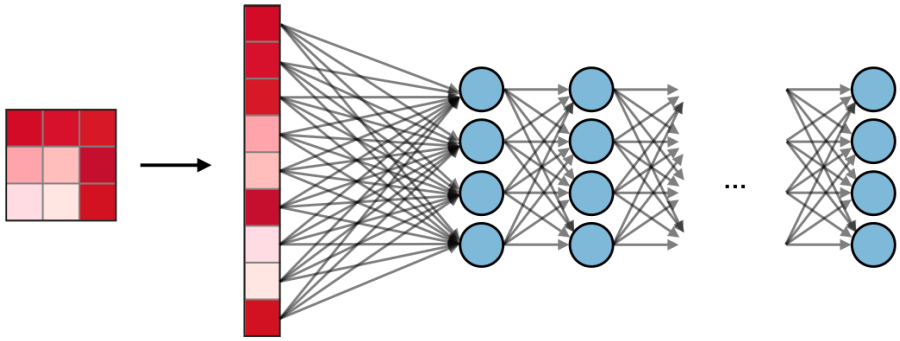
Bảng 2.1: So sánh giữa max pooling và average pooling.

Kiểu	Max pooling	Average pooling
Chức năng	Từng phép tổng hợp chọn giá trị lớn nhất trong khu vực mà nó đang được áp dụng.	Từng phép tổng hợp tính trung bình các giá trị trong khu vực mà nó đang được áp dụng.
Minh họa		
Nhận xét	• Bảo toàn các đặc trưng đã phát hiện. • Được sử dụng thường xuyên.	• Giảm kích thước bản đồ đặc trưng. • Được sử dụng trong mạng LeNet.

2.2.2.3. Lớp kết nối đầy đủ (Fully Connected - FC)

Lớp kết nối đầy đủ (FC) nhận tầng đầu vào là các dữ liệu đã được làm phẳng, mà mỗi tầng đầu vào đó được kết nối đến tất cả nơ ron. Trong mô hình mạng CNNs, các tầng kết nối đầy đủ thường được tìm thấy ở cuối mạng và được dùng để tối ưu hóa mục tiêu của mạng. Ví dụ, như độ chính xác của lớp.

Thường thì sau các lớp Conv và Pooling sẽ là hai lớp FC, một lớp (layer) để tập hợp các lớp đặc trưng (feature layer) đã tìm ra, chuyển đổi dữ liệu từ 3D, hoặc 2D thành 1D, tức chỉ còn là một véc tơ. Còn một lớp nữa là tầng đầu ra, số nơ ron của lớp này phụ thuộc vào số tầng đầu ra muốn tìm ra.



Hình 2.22: Mô tả tầng đầu vào đi qua lớp FC [43].

2.3 Mạng YOLO (You Only Look Once)

2.3.1 Giới thiệu về mạng YOLO

YOLO (You Only Look Once) là một giải pháp cho bài toán nhận dạng đối tượng khá phổ biến cho đến thời điểm hiện tại, thuật toán dựa trên mạng tích chập neuron và được đề xuất bởi Joseph Redmon cùng các cộng sự lần đầu tiên vào năm 2016 với tiêu đề You Only Look Once: Unified, Real-Time Object Detection [46]. Nó có khả năng xử lý video ở thời gian thực với đội trễ tối thiểu mà vẫn giữ được độ chính xác đáng nể, cùng với chức năng detect object với một ưu điểm nổi trội là nhanh hơn nhiều so với những mô hình cũ. Chỉ cần một lần truyền về phía trước để phát hiện tất cả các đối tượng trong một hình ảnh.

YOLO được thiết kế trong Darknet, một khung mạng neurel mã nguồn mở viết bằng ngôn ngữ C cùng với CUDA – được phát triển bởi cùng tác giả tạo ra YOLO, Joseph Redmon.

Về tốc độ xử lý của YOLO so với các thuật toán khác thì tốc độ xử lý YOLO mang lại sẽ nhanh trong cùng điều kiện tính toán, ảnh tầng đầu vào trực tiếp đi qua mạng nơ ron mà không cần phải đi qua một ống dẫn (input pipeline) phức tạp [46].

Về đánh giá sự xuất hiện của đối tượng thì YOLO đánh giá sự xuất hiện của đối tượng trên toàn bộ bức ảnh mà không đánh giá trên một vùng giới hạn như phương pháp cửa sổ trượt (sliding window) hay mạng RPN (Region Proposal Network).

2.3.2 Nguyên lý hoạt động

YOLO chia ảnh tầng đầu vào thành một lưới có kích thước là S × S. Nếu điểm trung tâm của đối tượng rơi vào một ô nào đó trong lưới thì ô đó có nhiệm vụ nhận dạng đối tượng. Mỗi ô trong lưới sẽ nhận dạng B khung chứa đối tượng (bounding box). Mỗi véc tơ nhận dạng sẽ có năm giá trị bx, by, bw, bh và p lần lượt là hoành độ và tung độ điểm trung tâm của đối tượng trong ô đó, chiều ngang (width) và chiều cao (height) của khung chứa đối tượng và cuối cùng là độ tin cậy (confidence). Độ tin cậy cho biết khả năng xuất hiện của đối tượng trong ô đó là

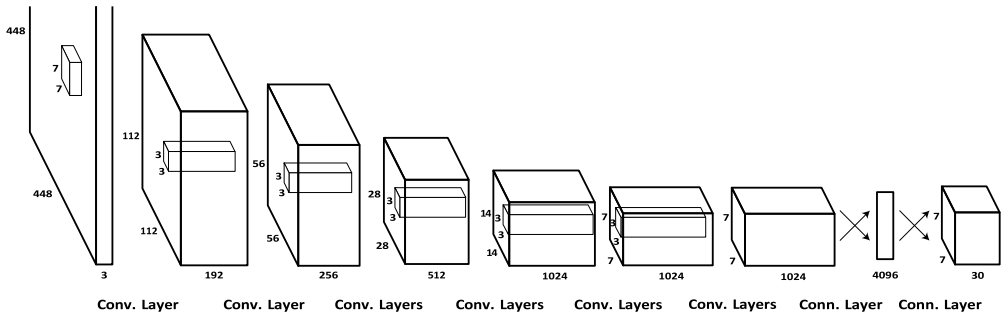
bao nhiêu, nếu giá trị của p thấp hơn một ngưỡng cho trước thì các giá trị còn lại không còn quan trọng.

Ngoài năm giá trị được trình bày ở trên, véc tơ nhận dạng còn bao gồm C giá trị (C bằng số lớp) biểu thị xác suất của từng lớp, giá trị Ci lớn nhất nghĩa là đối tượng được nhận dạng có chỉ số bằng i. Véc tơ đặc trưng của YOLO được mô tả trong hình.

2.3.3 Các phiên bản của mạng YOLO

2.3.3.1. Phiên bản 1 (YOLOv1)

Phiên bản đầu tiên của YOLO sử dụng 24 lớp tích chập cho việc trích xuất đặc trưng và hai lớp kết nối đặc cho việc nhận dạng đối tượng. Kiến trúc của YOLOv1 được biểu diễn như hình bên dưới.

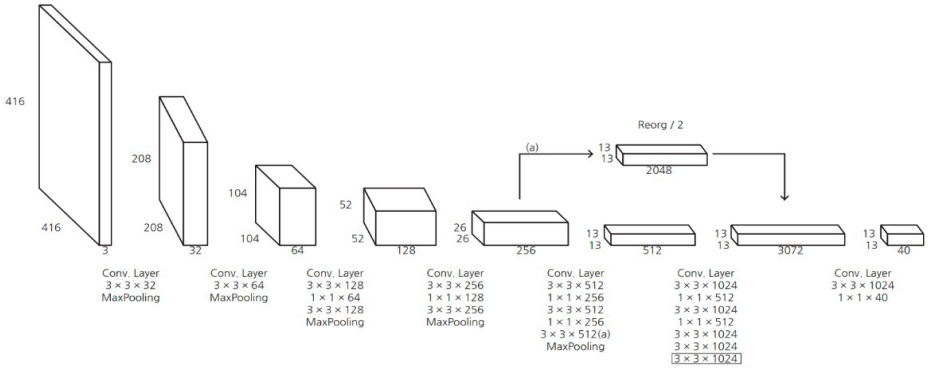


Hình 2.23: Kiến trúc của mạng YOLOv1 [46].

2.3.3.2 Phiên bản 2 (YOLOv2)

YOLOv2 là một phiên bản cải tiến của YOLOv1 về độ chính xác và thời gian nhận dạng.

YOLOv2 sử dụng phương pháp chuẩn hóa hàng loạt (batch normalization) trên các lớp tích chập thay vì sử dụng các lớp dropout sau các lớp kết nối đặc. Chính điều này đã làm cho mAP tăng lên gần 2%.

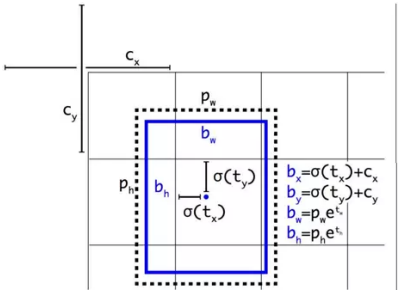


Hình 2.24: Kiến trúc mạng YOLOv2 [45].

Với YOLOv1, kích thước của các hộp neo (anchor box) được lựa chọn một cách ngẫu nhiên, với cách làm này, thuật toán sẽ hoạt động hiệu quả với một số loại đối tượng nhưng lại ảnh hưởng đến các đối tượng khác. Từ phiên bản 2 của YOLO, tác giả đề xuất sử dụng thuật toán K-means, cụ thể là duyệt

qua toàn bộ tập dữ liệu để phân cụm kích thước các khung chứa đối tượng, từ đó đề xuất giá trị của các hộp neo sao cho phù hợp hơn.

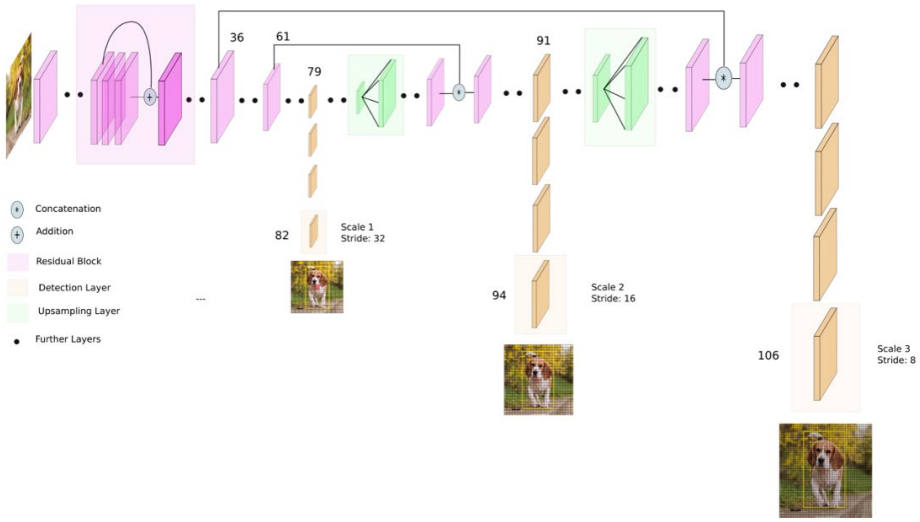
Thay vì sử dụng 2 lớp kết nối đặc (fully - connected) ở cuối mạng nơ ron, YOLOv2 bỏ qua các lớp này và tiến hành nhận dạng trên véc tơ đặc trưng từ các lớp tích chập phía trước theo một cách khác.



Hình 2.25: Hàm kích hoạt của YOLOv2 [45].

2.3.3.3 Phiên bản 3 (YOLOv3)

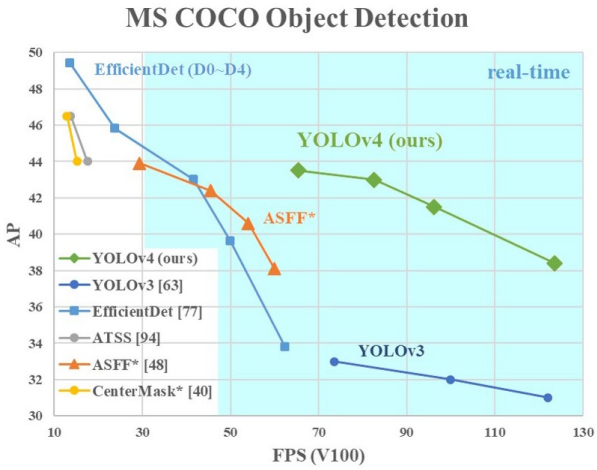
YOLOv3 có nhiều cải tiến so với các phiên bản trước, đặc biệt với với việc sử dụng nhiều kích thước khác nhau của lưới nhận dạng (multi-scales). Thay vì chỉ sử dụng một lưới có kích thước cố định là S × S như các phiên bản trước thì YOLOv3 sử dụng 3 lưới có kích thước lần lượt là 13 × 13, 26 × 26, 52 × 52, sau đó kết hợp các kết quả nhận dạng lại trước khi áp dụng ngưỡng và non-max suppression. Chính nhờ vào cách làm trên đã khiến cho YOLOv3 có khả năng nhận dạng các vật thể nhỏ tốt hơn và độ chính xác tăng đáng kể.



Hình 2.26: Kiến trúc mạng YOLOv3 [46].

2.3.3.4 Phiên bản 4 (YOLOv4)

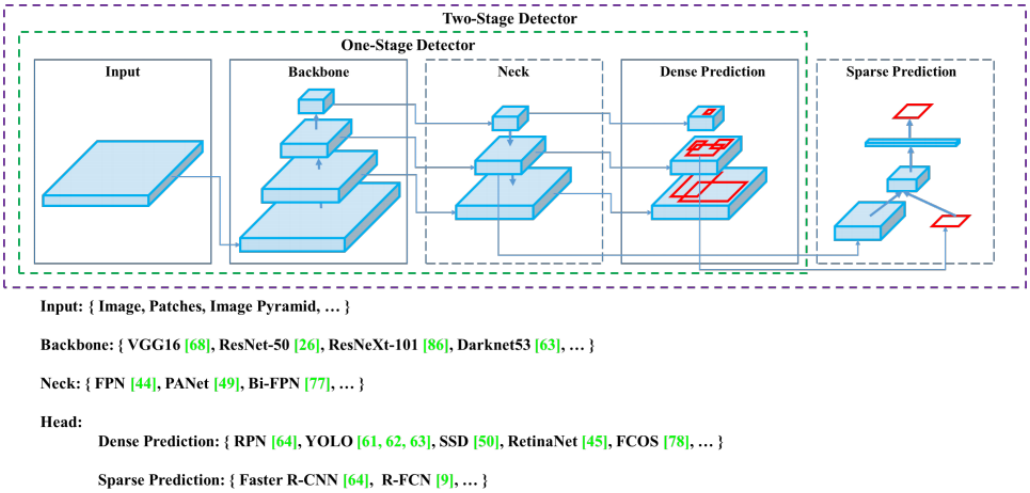
YOLOv4 có nhiều sự cải tiến đặc biệt giúp tăng độ chính xác và tốc độ hơn đối với người anh em YOLOv3 cũng như các mô hình khác trên cùng tập dữ liệu COCO và được xử lý trên phần cứng GPU V100.



Hình 2.27: So sánh hiệu suất và tốc độ xử lý của các mô hình YOLOv4 với các mô hình khác trên tập dữ liệu COCO [25].

Cấu trúc của YOLOv4 được tác giả chia làm bốn phần:

- Đầu vào (input),
- Xương sống (backbone),
- Cổ (neck)
- Đầu (head) gồm có:
  - Dự đoán dày đặc (dense prediction) - sử dụng các one-stage-detection như mạng YOLO, mạng SSD
  - Dự đoán thưa thớt (sparse prediction) – sử dụng các two-stage-detection như mạng RCNN.



Hình 2.28: Kiến trúc của mạng YOLOv4 [25].

➢ **Lựa chọn xương sống (Backbone):**

Mạng xương sống cho nhận dạng vật thể thường được huấn luyện trước (pre-train) thông qua bài toán phân loại ImageNet. Huấn luyện trước có nghĩa là trọng số của mạng đã được điều chỉnh để xác định các đặc trưng liên quan trong một hình ảnh, mặc dù chúng sẽ được tinh chỉnh trong nhiệm vụ

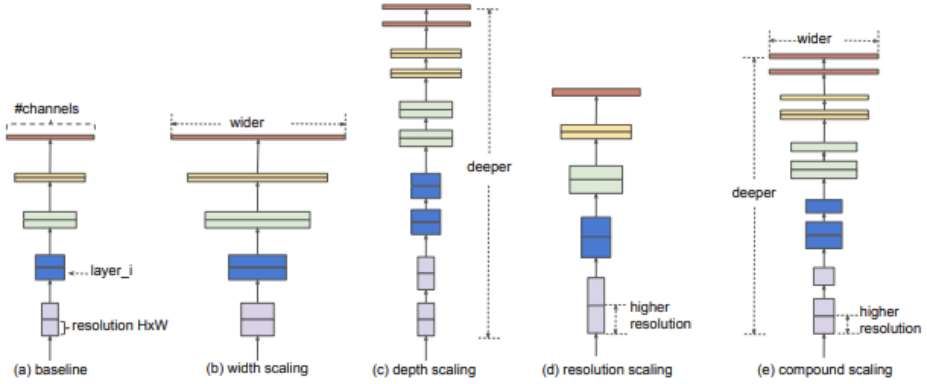
mới là phát hiện đối tượng. Tác giả đã sử dụng các xương sống như: CSPResNext50, CSPDarknet53, EfficientNet-B3.

Bảng 2.2: Các tham số của mạng nơ ron để phân loại hình ảnh.

Backbone model	Input network resolution	Receptive field size	Parameters	Average size of layer output (WxHxC)	BFLOPs (512x512 network resolution)	FPS (GPU RTX 2070)
CSPResNext50	512x512	425x425	20.6 M	**1058 K**	(15.5 FMA)	62
CSPDarknet53	512x512	725x725	27.6 M	950 K	(26.0 FMA)	**66**
EfficientNet-B3 (ours)	512x512	**1311x1311**	12.0 M	668 K	(5.5 FMA)	26

Mạng tích chập kết nối dày đặc (dense connected convolutional network - DenseNet) được thiết kế để kết nối các lớp trong mạng nơ-ron phức tạp nhằm mục đích làm giảm bớt vấn đề độ dốc (gradient) biến mất (khó có thể sao chép tín hiệu đã bị thất thoát trong một mạng rất sâu) để tăng cường lan truyền tính năng, khuyến khích mạng sử dụng lại các tính năng và giảm số lượng thông số mạng.

Mạng EfficientNet được thiết kế bởi Google Brain chủ yếu nghiên cứu về vấn đề mở rộng quy mô của mạng nơ-ron tích chập. Tác giả đã có rất nhiều ý kiến khi đưa ra quyết định mở rộng ConvNet bao gồm kích thước đầu vào, tỷ lệ chiều rộng, tỷ lệ chiều sâu và mở rộng tất cả những điều trên. Tác giả đã chọn mạng EfficientNet vì mạng có một vài một điểm hoàn hảo, có thể tối ưu cho tất cả các thông số đó và thông qua tìm kiếm để tìm thấy điểm đó.

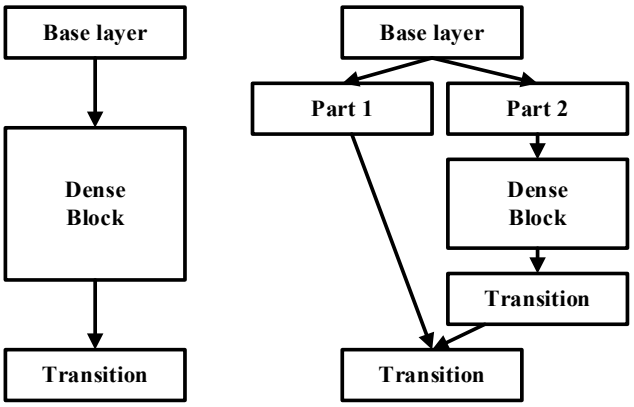


Hình 2.29: Kiến trúc của mạng EfficientNet [47].

EfficientNet vượt trội hơn các mạng khác có cùng kích thước về phân loại hình ảnh. Tuy nhiên, tác giả của YOLOv4 cho rằng các mạng khác có khả năng hoạt động tốt hơn trong cài đặt để phát hiện đối tượng nên quyết định thử nghiệm với tất cả 3 mạng CNN ở trên. Và cuối cùng thì tác giả chọn mạng CSPDarknet53 với xương sống cho mô hình.

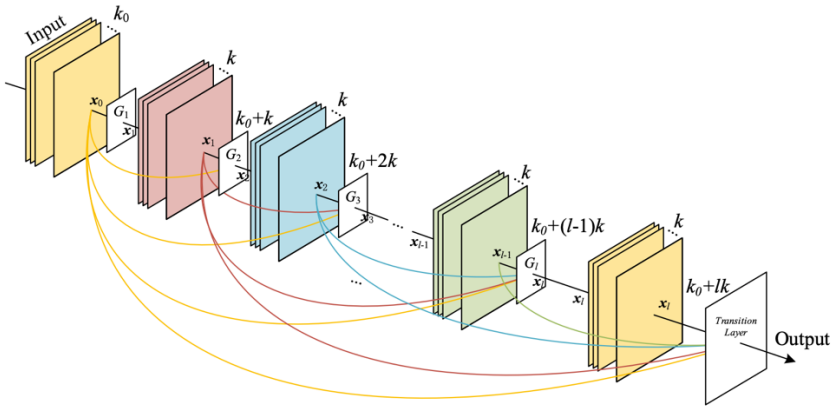
Cấu trúc của CPSDarknet53 được cấu tạo từ CSP và Darknet53.

CSP (Cross-Stage-Partial connections): có nguồn gốc từ kiến trúc Dense-Net sử dụng đầu vào trước đó và nối nó với đầu vào hiện tại trước khi chuyển vào lớp Dense. Nó có nhiệm vụ chia đầu vào của khối thành hai phần, một phần sẽ qua các khối chập, phần còn lại thì không (đi thẳng tới cuối khối). Sau đó, hai phần sẽ được cộng lại và đưa vào khối tiếp theo. Ý tưởng ở đây là loại bỏ các nút thắt tính toán trong DenseNet và cải thiện việc học bằng cách chuyển phiên bản chưa chỉnh sửa của bản đồ đặc trưng.



Hình 2.30: Mô tả cấu trúc CSP [49].

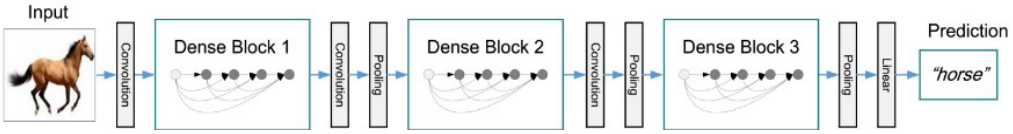
Khối Dense bao gồm nhiều lớp conv x(i) và H(i). Mỗi lớp H(i) bao gồm chuẩn hóa hàng loạt (batch normalization), ReLU và theo sau bởi một lớp tích chập. Các lớp H(i) này thay vì lấy đầu vào là đầu cuối của lớp ngay trước nó, thì các lớp này sẽ lấy tất cả các đầu cuối của các lớp trong khối Dense đó làm đầu vào.



Hình 2.31: Cấu trúc của khối Dense [48].

DenseNet (dense connected convolutional network) là một trong những netwok mới nhất cho visual object recognition. Nó cũng gần giống Resnet nhưng có một vài điểm khác biệt. DenseNet có cấu trúc gồm các dense block và các lớp chuyển tiếp (transition layers) được khối xếp dense - lớp chuyển tiếp -khối dense - lớp chuyển tiếp như hình 1.30. Với CNN truyền thống có L lớp thì sẽ có L kết nối (connection), còn trong mạng DenseNet sẽ

có L(L+1)/2 kết nối (connection) (tức là các lớp phía trước sẽ được liên kết với tất cả các lớp phía sau nó).



Hình 2.32: Cấu trúc của DenseNet [49].

YOLOv4 sử dụng CSPDarknet53 để làm xương sống vì CSPDarknet53 có độ chính xác trong nhiệm vụ phát hiện đối tượng (task object detection) cao hơn so với ResNet. Mặc dù ResNet có độ chính xác trong nhiệm vụ phân lớp (task classification) cao hơn, hạn chế này có thể được cải thiện nhờ kích hoạt Mish (activation Mish) và một vài kỹ thuật khác.

- **Bag of Freebies (BoF) cho xương sống:**

Một số phương pháp giúp cải thiện kết quả suy luận (inference) mà không làm ảnh hưởng tới tốc độ suy luận như tăng dữ liệu (data augmentation), mất cân bằng lớp (class imbalance), hàm tính chi phí (cost function), nhãn mềm (soft labeling), v.v…

- **Bag of Specials (BoS) cho xương sống:**

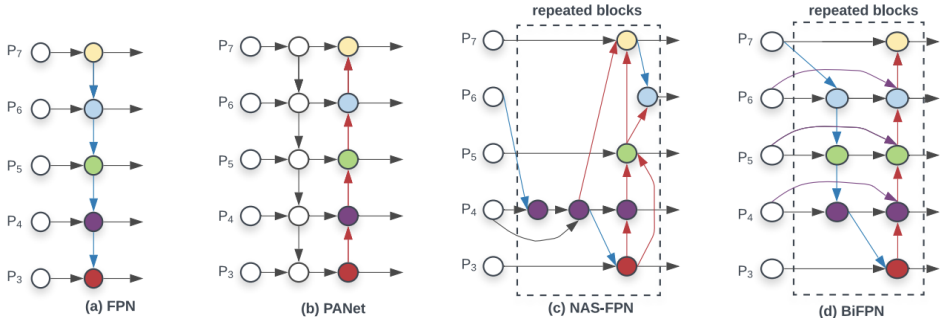
YOLOv4 sử dụng các phương pháp BoS sau cho xương sống: kích hoạt Mish (Mish activation), cross-stage partial connections (CSP), Multi-input weighted residual connections (MiWRC).

- **Phần cổ (neck) – tổng hợp đặc trưng:**

Cổ có nhiệm vụ trộn và kết hợp các bản đồ đặc trưng đã học được thông qua quá trình trích xuất đặc trưng từ xương sống và quá trình nhận dạng (YOLOv4 gọi là dự đoán Dense).

Với mỗi lần thực hiện phát hiện (detect) với các kích thước ảnh được thay đổi tỉ lệ (rescale) khác nhau, tác giả đã thêm các luồng đi từ dưới lên và các luồng đi từ trên xuống vào cùng nhau theo từng phần hoặc được nối với nhau trước khi đưa vào phần đầu (head). Từ đó lớp nhận dạng sẽ chứa thông tin phong phú hơn.

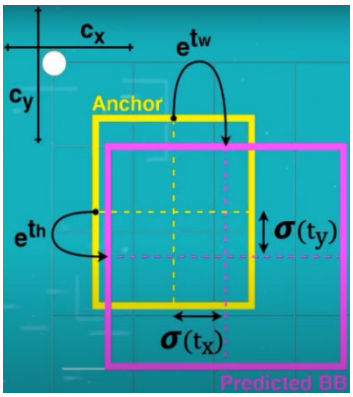
YOLOv4 đã cho phép tùy biến sử dụng các cấu trúc mạng cho phần cổ như là : FPN, PAN, NAS-FPN, BiFPN, ASFF, SFAM, SSP.



Hình 2.33: Một số cấu trúc mạng sử dụng cho phần cổ YOLOv4 [49].

- **Phần đầu (head)– bước nhận dạng:**

YOLOv4 sử dụng phần đầu giống như YOLOv3 với các hộp neo (anchor box) và nhận dạng với ảnh có kích thước khác nhau.



Hình 2.34: Quá trình huấn luyện để tinh chỉnh kích thước của hộp neo sao cho giống với vật thể nhất.

- **Bag of Freebies (BoF) cho máy dò:**

YOLOv4 sử dụng các BoF sau cho các máy dò như: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, huấn luyện đối phương (self-adversarial training), loại bỏ các lưới nhạy (eliminate grid sensitivity), sử dụng nhiều hộp neo cho một ground truth (using multiple anchors for a single ground truth), cosine annealing scheduler, tối ưu siêu tham số (optimal hyperparameters), huấn luyện ngẫu nhiên các hình dạng (random training shapes).

- **Bag of Specials (BoS) cho máy dò :**

Các đặc điểm của BoS cho máy dò trong YOLOv4: kích hoạt Mish (Mish activation), thay đổi khối SPP-block, thay đổi khối SAM-block, thay đổi khối PAN (path-aggregation), DIoU-NMS.

2.3.4 Hạn chế của mạng YOLO

Mặc dù mạng YOLO không phải là phương pháp có độ chính xác cao nhất thế nhưng mạng YOLO vẫn là mô hình mạng được ưa chuộng trong nhiều dự án thực tế được triển khai thi công, khi mà độ chính xác không phải là ưu tiên hàng đầu.

Mạng YOLO tạo ra các ràng buộc không gian mạnh mẽ đối với các dự đoán hộp giới hạn (bounding box), bởi mỗi ô lưới (grid cell) chỉ dự đoán rất ít hộp giới hạn và chỉ có duy nhất một lớp (class). Ngoài ra, YOLO còn có ràng buộc về không gian và giới hạn về số lượng các đối tượng lân cận mà mô hình có thể dự đoán được. YOLO phải chiến đấu với việc xử lý các vật thể nhỏ xuất hiện theo nhóm. Cũng vì những lý do trên và bên cạnh đó là mô hình còn học các dự đoán đến từ các hộp giới hạn của dữ liệu đã được gán nhãn trước làm cho mô hình này gặp

nhiều khó khăn trong việc tổng quát hóa các đối tượng ở các tỉ lệ cấu hình mới hoặc tỉ lệ bất thường.

Cuối cùng, trong khi huấn luyện về một hàm mất mát (loss function) để ước tính hiệu suất phát hiện thì hàm mất mát này đã xử lý các lỗi giống nhau trong các hộp giới hạn có kích thước nhỏ hơn so với các hộp giới hạn có kích thước lớn. Một lỗi nhỏ trong hộp có kích thước lớn nhìn chung thì không có tác hại to lớn gì nhưng một lỗi rất nhỏ trong hộp cũng có thể ảnh hưởng đáng kể đến giá trị IoU. Nguyên nhân xảy ra lỗi như trên chính là do mức cục bộ hóa chưa phù hợp (incorrect localizations) của mô hình.

2.3.5 Lợi ích của việc xử dụng mạng YOLO

Mạng YOLO với tốc độ xử lý nhanh và rất phù hợp cho việc xử lý hình ảnh ở thời gian thực.

Dự đoán (vị trí đối tượng và các lớp) được thực hiện từ một mạng duy nhất. Có thể được huấn luyện từ đầu đến cuối dễ dàng và cải thiện độ chính xác với số mẫu càng lớn.

Phương pháp sử dụng mạng YOLO vượt trội hơn so với các phương pháp khác khi hợp nhất từ hình ảnh tự nhiên sang các lĩnh vực khác nhau như các tác phẩm nghệ thuật.

Phương pháp đề xuất khu vực giới hạn phân loại cho khu vực cụ thể. YOLO có thể dễ dàng dự đoán ranh giới hình ảnh vì nó có thể truy cập vào toàn bộ hình ảnh. Với những hình ảnh có nhiều hình ảnh phụ thì YOLO thể hiện xử lý một cách hiệu quả và ít sai hơn trong các khu vực chỉ có nền.

Không những thế, YOLO còn phát hiện một đối tượng trên mỗi ô lưới và thực thi sự đa dạng ở các không gian trong việc dự đoán đối tượng.

2.4 Biểu đồ của hướng dốc (Histogram of Oriented Gradients - HOG)

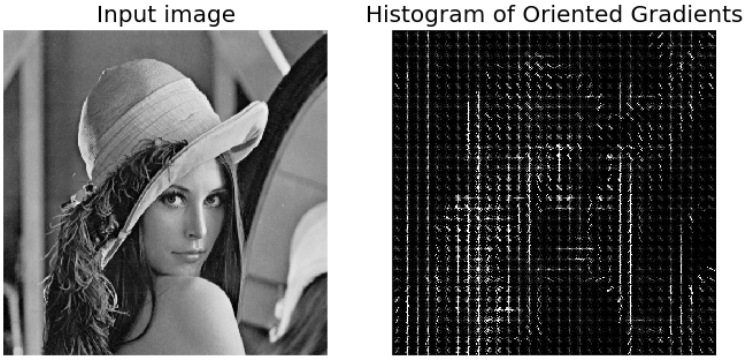
Trích xuất đặc trưng HOG sẽ tạo ra các bộ mô tả đặc trưng (feature descriptor) nhằm mục đích phát hiện vật thể (object detection).

Ý tưởng của trích xuất đặc trưng HOG là từ một bức ảnh, HOG sẽ lấy ra 2 ma trận quan trọng giúp lưu thông tin ảnh đó là cường độ gradient (gradient magnitude) và phương hướng của gradient (gradient orientation). Bằng cách kết hợp 2 thông tin này vào một biểu đồ phân phối histogram, trong đó độ lớn gradient được đếm theo các nhóm bins của phương gradient. Sau đó, HOG sẽ thu được véc tơ đặc trưng HOG đại diện cho histogram.

Trên thực tế, HOG còn hoạt động phức tạp hơn khi véc tơ HOG sẽ được tính trên từng vùng cụ bộ như mạng CNN và sau đó là phép chuẩn hóa cụ bộ để đồng nhất độ đo. Cuối cùng véc tơ HOG tổng hợp từ các véc tơ trên vùng cục bộ.

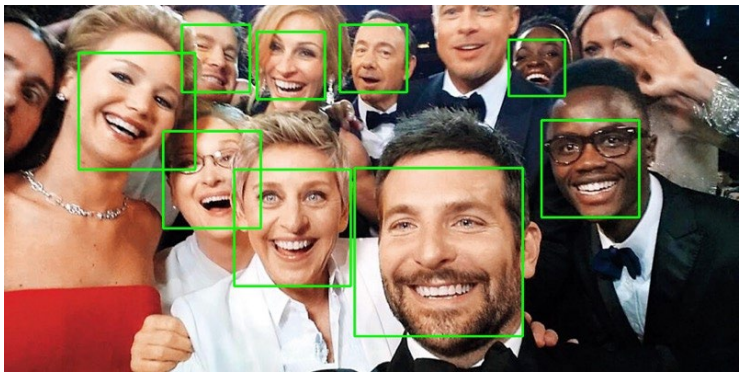
HOG là một phương pháp trích xuất đặc trưng từ ảnh được dùng trong bài toán phát hiện đối tượng. Với mỗi bức ảnh được hình thành từ rất nhiều điểm ảnh (pixel) thì

phương pháp HOG về cơ bản sẽ tính toán sự thay đổi của mỗi điểm ảnh so với các điểm ảnh xung quanh để tạo ra đặc trưng riêng cho ảnh.



Hình 2.35: Ví dụ về trích xuất đặc trưng HOG [50].

Phương pháp này trở nên phổ biến từ năm 2005 vì đã cho thấy hiệu quả rất cao trong khi mô hình nhận dạng người qua đường kết hợp HOG với Linear SVM. Trong số đó, không thể không kể đến mô hình trong thư viện Dlib. Dlib cũng là mô hình nhận dạng khuôn mặt tốt nhất nếu bỏ qua các mô hình học sâu khác.



Hình 2.36: Kết quả phát hiện các khuôn mặt có trong hình ảnh khi dùng mô hình Dlib dựa trên trích xuất đặc trưng [50].

2.5 Phép giản nỡ (dilation)

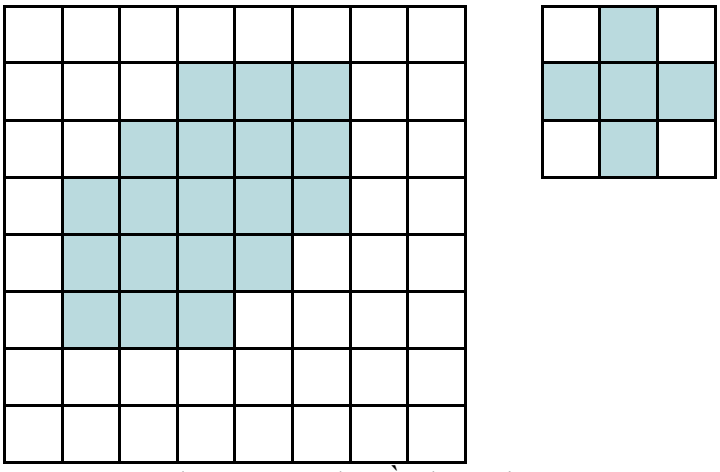
Toán tử hình thái là công cụ toán học để xử lý hình dạng trong ảnh. Toán tử hình thái sử dụng hướng tiếp cận lý thuyết tập hợp. Toán tử hình thái được ứng dụng trong tác biên ảnh, lấp đầy vùng ảnh, tạo kết nối giữa các vùng ảnh hoặc làm xương ảnh.

Phép giãn nỡ (dilation) là một trong bốn phép toán tử của hình thái. Phép toán này có tác dụng làm cho đối tượng ban đầu trong ảnh tăng lên về kích thước, các lỗ nhỏ trong ảnh được lấp đầy, nối liền đường biên ảnh đối với những đoạn rời nhỏ, lấp đầy khoảng trống, mở rộng vùng ảnh. Công thức tính của phép toán được hình thành khi đối tượng X và phần tử cấu trúc (mẫu) B trong không gian Euclide hai chiều:

$$X \oplus B = \bigcup_{x \in X} B_x \tag{2.8}$$

- Trong đó:
  - B_x là dịch chuyển của B đến vị trí x.
  - B là tập hợp của tất cả các B_x với x thuộc X.

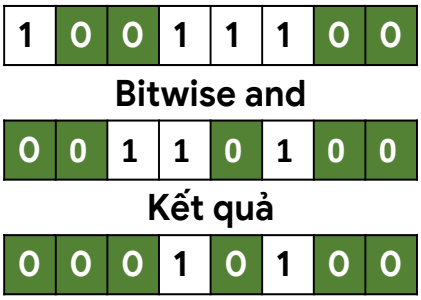
- $X \neq Ø$.



Hình 2.37: Ví dụ về phép giãn nở.

2.6 Phép bitwise and

Toán tử bitwise and thực hiện kết hợp logic trên các bit tương ứng với các toán hạng của nó. Đối với mỗi cặp bit chiếm cùng một vị trí trong hai số, nó chỉ trả về một khi cả hai bit được bật.



Hình 2.38: Mô tả toán tử bitwise and.

Mẫu bit kết quả là giao điểm của các đối số của toán tử. Nó có hai bit được bật ở các vị trí mà cả hai toán hạng là một. Ở tất cả các vị trí khác, ít nhất một trong các đầu vào có bit là 0.

Điều này tương đương với tích của hai giá trị bit. Công thức tính bitwise của số a và b bằng cách nhân các bit của chúng với mọi chỉ số i:

$$(a \,\&\, b)_i = a_i \times b_i \tag{2.9}$$

2.7 Ngôn ngữ lập trình và các công cụ

Trong Luận văn này, ngôn ngữ lập trình Python với kho thư viện phong phú được sử dụng giúp người dùng có thể dễ dàng tiếp cận những hàm, thư viện đã được tối ưu hóa như thư viện argparse, cython, face_recognition, dlib, future, image, imutils, joblib, launchpadlib, libusb, libusb1, lxml, mkl, mss, numpy, opencv, pickle, Pillow, protobuf, pybind11, pygobject, pynetworktables, pyqt5-sip, pyqt5-tools, PyQtWebEngine, pyshine, pyudev, pyusb, pyzbar, scipy, screeninfo, setuptools, sounddevice,

testresources, v.v..…giao diện đồ họa PyQt sẽ cung cấp giao diện hướng đối tượng mạnh mẽ đến các bộ công cụ.

Công cụ gán nhãn ảnh labelImg được dùng để gán nhãn cho hình ảnh và phục vụ cho mục đích huấn luyện các mô hình máy học. Công cụ labelImg có thể gán nhãn cho hình ảnh và xuất ra các tập tin như:

- Tập dữ liệu với phần mở rộng .xml với định dạng PASCAL VOC được sử dụng trong hầu hết các mô hình học sâu. Nghiên cứu sử dụng trong tập dữ liệu người mang khẩu trang hay không mang khẩu trang
- Tập dữ liệu với phần mở rộng .txt được sử dụng trong mô hìnhYOLO. Nghiên cứu sử dụng trong tập dữ liệu dữ liệu rau củ quả.

2.8 Phương pháp đánh giá mô hình phát hiện đội tượng, nhận diện khuôn mặt

Cách tính IOU và hàm mất mát:

- Để cập nhật trọng số cho mô hình, các phương pháp object detection thường dùng hàm loss là IoU.

$$IoU = \frac{\left|B \bigcap B^{gt}\right|}{\left|B \bigcup B^{gt}\right|} \tag{2.10}$$

$$L_{IoU} = 1 - \frac{\left|B \bigcap B^{gt}\right|}{\left|B \bigcup B^{gt}\right|} \tag{2.11}$$

Trong đó:

B: boundary box.

Gt: ground truth.

- Tuy nhiên, khi sử dụng IoU, xét hai trường hợp hộp dự đoán giới hạn (prediction bounding box) và hộp giới hạn thực địa (groundtruth bounding box) không giao nhau (overlapping), làm cho dự đoán khó có thể biết được trường hợp nào tốt hơn trường hợp nào. Do đó, gây khó khăn cho việc cập nhật trọng số theo hướng khiến hộp dự đoán giới hạn tiến gần tới hộp giới hạn thực địa.
- GioU giải quyết điều này bằng cách thêm vào IoU một thành phần C - hộp giới hạn nhỏ nhất chứa cả hộp dự đoán giới hạn và hộp giới hạn thực địa (xem như khoảng cách giữa hai hộp giới hạn này).

$$L_{GIoU} = 1 - IoU + \frac{\left|C - B \bigcup B^{gt}\right|}{\left|C\right|} \tag{2.12}$$

Trong đó: C là hộp giới hạn nhỏ nhất bao phủ giữa B và Bgt.

- Tuy nhiên, khi sử dụng GIoU lại có một vấn đề là mô hình có khuynh hướng mở rộng hộp dự đoán giới hạn trước cho tới khi nó giao nhau (overlapping) với hộp giới hạn thực địa, sau đó mới co lại để giảm IoU.
- DIoU giải quyết được vấn đề này bằng cách không chỉ đưa hộp giới hạn C vào ràng buộc mà còn đưa khoảng cách giữa tâm của hộp dự đoán giới hạn và tâm của hộp giới hạn thực địa. Bây giờ, thành phần c ở mẫu chỉ đóng vai trò chuẩn hóa khoảng cách giữa hai tâm của hộp dự đoán giới hạn và hộp giới hạn thực địa.

$$R_{DIoU} = \frac{\rho^2(b, b^{gt})}{c^2} \tag{2.13}$$

Trong đó: b và bgt biểu thị các điểm chính giữa của B và Bgt, ρ(.) là khoảng cách của Euclidean, và c là độ dài đường chéo của hộp bao quanh nhỏ nhất bao phủ hai hộp.

$$L_{DIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} \tag{2.14}$$

- Cuối cùng, CIoU đưa thêm vào tham số giúp duy trì tỷ lệ của các hộp giới hạn.

$$R_{DIoU} = \frac{\rho^2(b, b^{gt})}{c^2} \tag{2.15}$$

Trong đó, b và bgt biểu thị các điểm chính giữa của B và Bgt, ρ(.) là khoảng cách của Euclidean, và c là độ dài đường chéo của hộp bao quanh nhỏ nhất bao phủ hai hộp.

$$R_{CIoU} = \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{2.16}$$

Trong đó, α tham số cần bằng dương (positive trade-off parameter) và v đo lường tính nhất quán của tỉ lệ khung hình.

$$v = \frac{4}{\pi^2}(\arctan\frac{w^{gt}}{h^{gt}} - \arctan\frac{w}{h})^2 \tag{2.17}$$

- Sau đó, hàm mất mát sẽ được định nghĩa:

$$L_{CIoU} = 1 - IoU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha v \tag{2.18}$$

- Và tham số cần bằng (trade-off) α được định nghĩa:

$$\alpha = \frac{v}{(1 - IoU) + v'} \tag{2.19}$$

Cách tính Precision và Recall:

$$\text{Prediction} = \frac{TP}{TP+FP} \tag{2.20}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{2.21}$$

Trong đó mô hình được đánh giá dựa trên bảng tiêu chí sau.

Bảng 2.3: Tiêu chí đánh giá mô hình.

STT	Dự đoán	Nghĩa	Nội dung
1	TP	True Positive	Mô hình dự đoán đúng lớp của đối tượng/khuôn mặt và khung chứa đối tượng/khuôn mặt.
2	FP	False Positive	Mô hình dự đoán không có phát hiện/nhận dạng được đối tượng/khuôn mặt.
3	FN	False Negative	Mô hình dự đoán được khung chứa đối tượng/khuôn mặt nhưng sai lớp của đối tượng/khuôn mặt.

Cách tính F1-score (với F1-score là trung bình điều hòa (harmonic mean) của precision và recall):

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} \quad (2.22)$$

2.9 CUDA, cuDNN, DeepStream SDK và TensorRT

2.9.1 Giới thiệu về CUDA và nhân CUDA

CUDA (Compute Unified Device Architecture) là kiến trúc hợp nhất tính toán của các thiết bị điện tử được phát triển độc quyền bởi hãng công nghệ NVIDIA.

Nhân CUDA (CUDA core) là một nhân xử lý trong GPU của card đồ họa. Nhân CUDA có cấu trúc cách thức tính toán đơn giản. Số lượng nhân CUDA có thể được tích hợp trên GPU lên đến con số hàng nghìn.

Nhân CUDA đóng vai trò quan trọng trong quá trình tính toán, xử lý thông tin nhận được của GPU. GPU có chứa càng nhiều nhân CUDA thì khả năng tính toán đồng thời nhiều thông tin càng nhanh và chính xác. Nhân CUDA là một trong những thành phần quyết định đến chất lượng hình ảnh hiển thị khi người dùng chơi trò chơi điện tử, xử lý tệp đồ họa được kết xuất sau khi thiết kế có độ tỉ mỉ, chính xác ra sao hay thậm chí là quyết định thời gian nghiên cứu một chủ đề khoa học sẽ nhanh hay chậm.

2.9.2 Giới thiệu về cuDNN

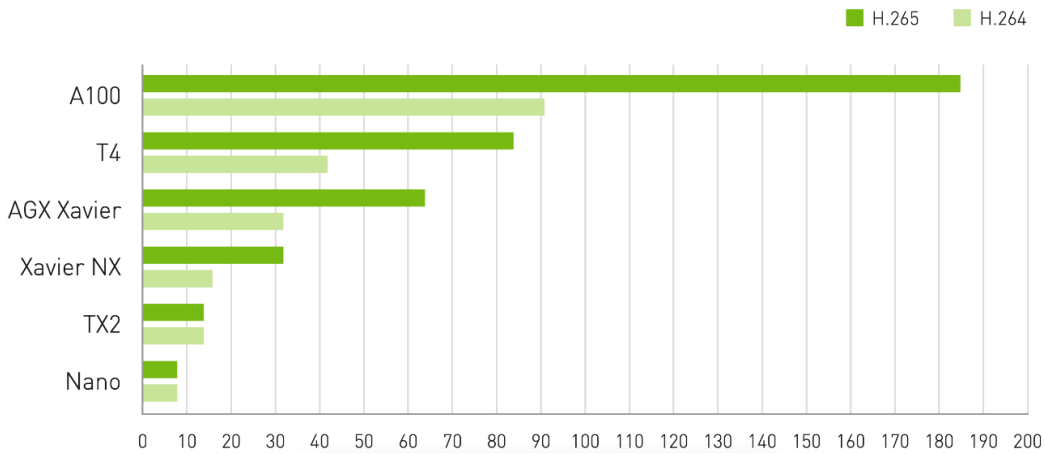
CuDNN – NVidia CUDA® Deep Neural Network là một thư viện nền tảng cho các mạng nơ ron học sâu được tăng tốc bởi GPU. CuDNN cung cấp các thiết lập được tinh chỉnh cho các thủ tục được chuẩn hóa như chuyển tiếp (forward) and quay lui (backward) tích chập (convolution), tổng hợp (pooling), chuẩn hóa (normalization) và các lớp kích hoạt. CuDNN là một phần của DL SDK do NVidia cung cấp một môi trường phát triển toàn diện cho các nhà phát triển C và C ++ để xây dựng các ứng dụng tăng tốc GPU với hiệu suất cao cùng các thư viện.

2.9.3 Giới thiệu về DeepStream SDK

DeepStream SDK được phát triển bởi NVIDIA. DeepStream cung cấp cho các nhà phát triển tùy chọn phát triển bằng ngôn ngữ C/C++, Python hoặc sử dụng lập trình đồ họa mã thấp với Graph Composer. DeepStream cung cấp nhiều phần bổ trợ (plug-in) và tiện ích mở rộng được tăng tốc phần cứng. Hỗ trợ chạy trên cả hai nền tảng NVIDIA Jetson Nano và dGPU.

DeepStream là một phần không thể thiếu của NVIDIA Metropolis - nền tảng để xây dựng các dịch vụ và giải pháp đầu cuối (end-to-end) giúp chuyển đổi dữ liệu điểm ảnh và cảm biến thành những thông tin chi tiết hữu ích.

Tốc độ triển khai và thực thi nhanh chóng với các ứng dụng và dịch vụ Vision AI. DeepStream cung cấp các tính năng bảo mật hỗ trợ đa nền tảng. DeepStream còn cung cấp bộ công cụ phân tích phát trực tuyến hoàn chỉnh để xử lý đa cảm biến với thuật toán AI trên video và hình ảnh. DeepStream SDK có thể áp dụng AI để truyền phát video và có thể đồng thời tối ưu hóa về giải mã/mã hóa video, chia tỷ lệ và chuyển đổi hình ảnh cũng như kết nối edge-to-cloud để tối ưu hóa hiệu suất hoàn chỉnh từ đầu đến cuối.



Hình 2.39: Tóm tắt mật độ luồng (stream) đạt được ở chuẩn H.264 và H.265 với khung hình 1080p/30fps trên các thiết bị của NVIDIA [51].

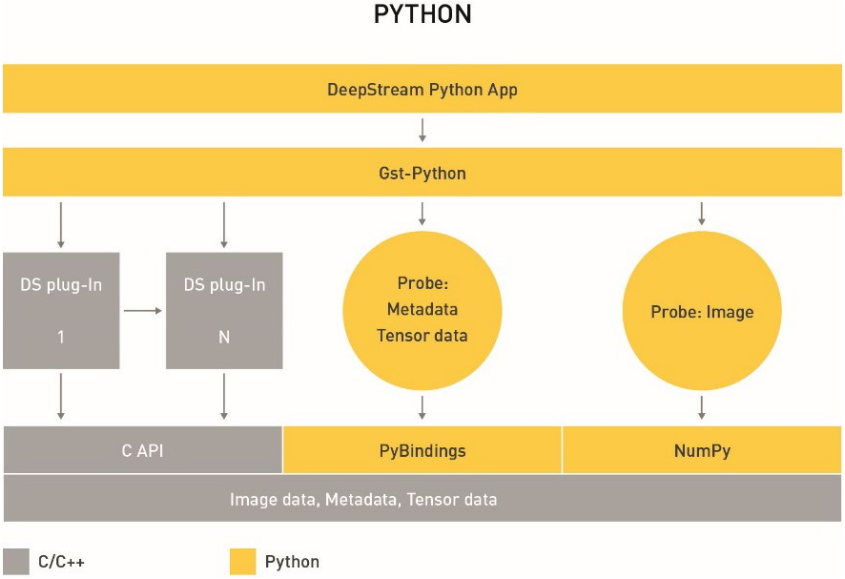
DeepStream cung cấp sự linh hoạt từ tạo mẫu nhanh đến các giải pháp cấp sản xuất đầy đủ. Sử dụng NVIDIA TensorRT cho khả năng suy luận thông lượng cao với các tùy chọn hỗ trợ đa GPU, đa luồng để có thể đạt được hiệu suất tốt nhất.

DeepStream được xây dựng không chỉ phục vụ cho nhà phát triển mà còn cho cả doanh nghiệp. DeepStream cung cấp hỗ trợ cho nhiều mô hình AI dựa trên phân loại hình ảnh và phân đoạn và phát hiện đối tượng phổ biến như SSD hay hiện đại hơn như YOLO, FasterRCNN và MaskRCNN. Ngoài ra, có thể tùy chỉnh hệ thống DeepStream.

					Jetson Nano*	Jetson TX2*	Jetson Xavier NX			Jetson AGX Xavier			T4	A100
Application	Models	Inference Resolution	Precision	Model Accuracy	GPU (FPS*)	GPU (FPS)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)	DLA1 (FPS)	DLA2 (FPS)	GPU (FPS)	GPU (FPS)
People Detect	PeopleNet-ResNet34	960x544	INT8	84%	12	31	172	48	48	305	53	53	926	3345
Vehicle Detect	TrafficCamNet-ResNet18	960x544	INT8	84%	19	51	274	89	89	486	111	111	1353	3855
Vehicle Detect	DashCamNet-ResNet18	960x544	INT8	80%	18	46	261	91	91	460	116	116	1341	3870
Face Detect	FaceDetect-IR-ResNet18	384x240	INT8	96%	101	276	1126	455	455	2007	624	624	2516	5578

Hình 2.40: Hiệu suất luồng khi dùng DeepStream trên khung hình 1080p/30fps trên các thiết bị của NVIDIA [51].

Mô hình kiến trúc hoạt động và xử lý của DeepStream.



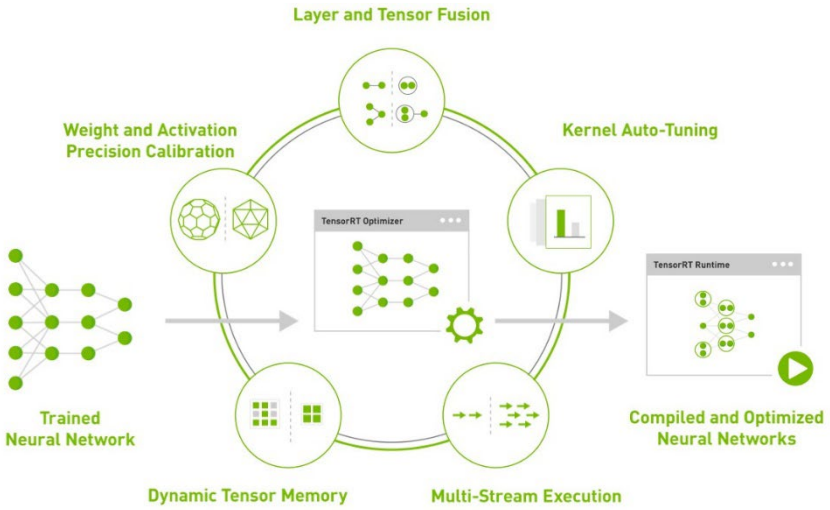
Hình 2.41: Kiến trúc xử lý của DeepStream [52].

2.9.4 Giới thiệu về TensorRT

TensorRT là bộ công cụ phát triển phần mềm (SDK) học do Nvidia phát triển. TensorRT ra đời nhằm cải thiện tốc độ inference, giảm độ trì truệ trên các thiết bị đồ họa NVIDIA(GPU). TensorRT có thể cải thiện tốc độ suy luận lên đến 2-4 lần so với các dịch vụ real-time và nhanh hơn gấp 30-40 lần so với hiệu suất khi dùng CPU trong quá trình suy luận. TensorRT hỗ trợ ngôn ngữ C++ và Python. TensorRT hỗ trợ những nền tảng: Embedded (Jetson), Automotive (Drive PX), Data center (Tesla GPUs).

2.9.4.1 Phương pháp tối ưu của TensorRT

TensorRT sẽ thực hiện năm loại tối ưu để tăng hiệu suất inference:



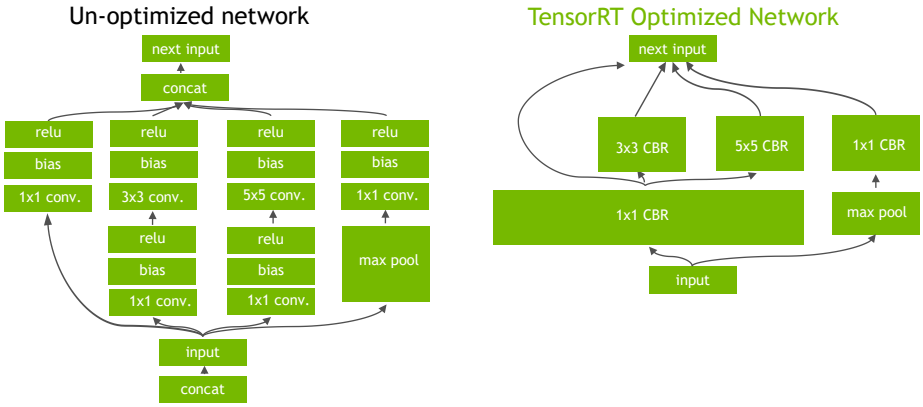
Hình 2.42: Năm loại tối ưu hóa để tăng hiệu suất suy luận trên TensorRT [53].

- Precision Calibration: trong suốt quá trình huấn luyện, các tham số và hàm kích hoạt trong độ chính xác Tensor Cores FP32 (Float Point 32) sẽ được chuyển đổi về độ chính xác FP16 hoặc INT8. Việc chuyển đổi như vậy sẽ giúp tối ưu mô hình như giảm độ trì truệ, tăng tốc độ suy luận.

Bảng 2.4: Vùng giá trị của các Tensor Cores FP32, FP16, INT8.

Tensor Cores	Vùng giá trị	Minimum Positive Value
FP32	-3.4x1038~+3.4x1038	1.4×10^{-45}
FP16	-65504~+65504	5.96×10^{-8}
INT8	-128~+127	1

- Layer và Tensor Fusion: TensorRT sẽ gộp các layer và tensor để tối ưu hóa bộ nhớ GPU và băng thông. Các node sẽ được gộp theo chiều dọc, chiều ngang hoặc cả hai chiều. Việc gộp như vậy sẽ giúp giảm một số dung lượng như đọc và ghi dữ liệu Tensor cho mỗi lớp.



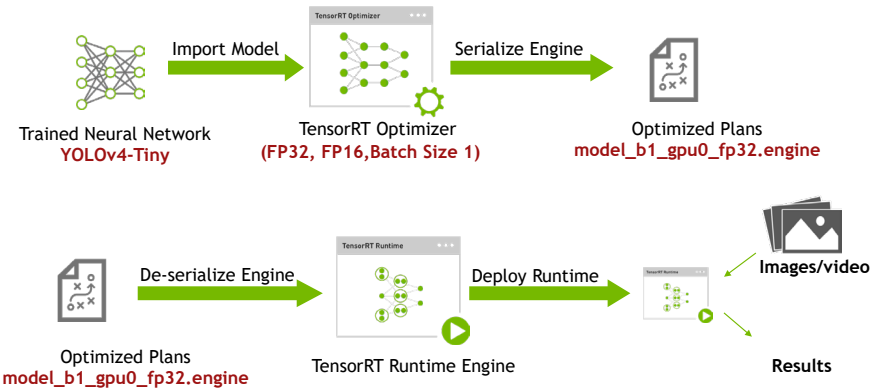
Hình 2.43: Sự kết hợp lớp dọc và lớp ngang của TensorRT trên mô hình GoogLeNet Inception [54].

- Cải thiện việc sử dụng GPU: khởi chạy ít kernel hơn, sử dụng bộ nhớ và băng thông tốt hơn.
- Hợp nhất hàng dọc: kết hợp các lệnh gọi kernels theo tuần tự.
- Hợp nhất hàng ngang: kết hợp các kernels có đầu vào giống nhau nhưng khác trọng lượng.

- Kernel auto-tuning: trong quy trình tối ưu, một vài kernel được cung cấp dành riêng cho việc tối ưu sẽ thực thi trong suốt tiến trình.
  - Có nhiều thuật toán triển khai cấp thấp cho các việc phép tính thông thường.
  - TensorRT chọn các tối ưu kernel dựa trên các tham số như: batch_size, filter-size, input data size.
  - TensorRT chọn tối ưu kernel dựa trên mục tiêu của nền tảng.
- Dynamic Tensor Memory:
  - Chỉ phân bổ bộ nhớ cần thiết cho mỗi Tensor và hạn chế thời gian sử dụng của Tensor.
  - Giảm dung lượng bộ nhớ và cải thiện khả năng tái sử dụng bộ nhớ.
- Multiple Stream Execution: TensorRT cho phép xử lý song song nhiều luồng đầu vào.

2.5.4.2 Kiến trúc mô hình TensorRT

Sơ đồ kiến trúc tối ưu mô hình và thực thi mô hình của TensorRT.



Hình 2.44: Sơ đồ kiến trúc tối ưu mô hình và thực thi mô hình của TensorRT [54].

CHƯƠNG 3. PHƯƠNG PHÁP NGHIÊN CỨU

3.1 Môi trường thực nghiệm

3.1.1 Tổng quan về thiết bị Nvidia Jetson Nano

Hình 3.1: Nvidia Jetson Nano Kit [55].

Một số thông số về phần cứng và phần mềm của thiết bị NVIDIA Jetson Nano mà đề tài luận văn sử dụng để thực nghiệm.

Bảng 3.1: Thông số phần cứng của NVIDIA Jetson Nano.

STT		Thông số kỹ thuật
1	Mô-đun	Jetson Nano Developer Kit Carrier Board A02
2	Chip xử lý (CPU)	Lõi tứ ARM Cortex-A57 MPCore 1.43 GHz
3	Chip đồ họa (GPU)	NVIDIA Maxwell với 128 lõi NVIDIA CUDA®
4	RAM	4 GB 64-bit LPDDR4, 1600MHz 25.6 GB/s
5	Bộ nhớ lưu trữ	Thẻ nhớ microSD 128GB
6	Độ phân giải mã hóa quay phim	250MP/giây 1×4K@30fps | 2×1080p@60fps | 4×1080p@30fps | 4×720p@60fps | 9×720p@30fps (H.264/H.265)
7	Độ phân giải giải mã quay phim	500MP/giây 1×4K@60fps | 2×4K@30fps | 4×1080p@60 fps | 8×1080p@30fps | 9×720p@60fps (H.264/H.265)
8	Máy ảnh	2xMIPI CSI-2 (15 vị trí đầu nối Camera Flex)
9	Kết nối không dây	Intel® Dual Band Wireless-AC 8265 M.2 Key E (PCIe x 1)
10	Kết nối có dây	Gigabit Ethernet (RJ45)
11	Cổng xuất hình	1×HDMI 2.0, 1×DisplayPort
12	Cổng kết nối	4×USB 3.0 Type-A, USB 2.0 Micro-B
13	Đầu nối SO-DIMM	260 chân
14	Đầu cắm mở rộng	40 chân với tín hiệu – 3×I2C, 2×SPI, UART, I2S, GPIOs

15	Nhiệt độ hoạt động	Từ 25ºC – 80ºC
16	Nguồn vào	Micro-USB (5V⎓2.5A) hoặc DC (5V⎓4A)
17	Kích thước lõi	69.6mm×45mm
18	Kích thước toàn bộ Kit	100mm×80mm×29mm

Bảng 3.2: Thông số phần mềm của NVIDIA Jetson Nano.

STT	Thông số phần mềm	
1	Hệ điều hành	Ubuntu 18.04.5
2	Jetpack SDK	4.5.1 bao gồm L4T 32.5.1
3	TensorRT	7.1.3
4	CUDA	10.2
5	cuDNN	8.0
6	DeepStream	5.1
7	Dlib	19.9
8	VPI (Vision Programing Interface)	1.0
9	OpenCV	4.4.0
10	Visionworks	1.6
11	NVIDIA Nsight Systems	2020.5
12	NVIDIA Nsight Graphics	2020.5
13	face_recognition	1.30

3.1.2 Tổng quan về thiết bị Camera Logitech C922 Pro Stream

Camera Logitech C922 là webcam truyền trực tuyến ở thời gian thực với chất lượng hình ảnh chuẩn HD 720p siêu nhanh đạt tốc độ khung hình 60fps. Camera cung cấp phần mềm hỗ trợ cho hai hệ điều hành: Windows và Mac OS. Một số thông số về phần cứng và thông số camera hỗ trợ trên hệ điều hành Linux.

Hình 3.2: Camera Logitech C922 Pro Stream [56].

Bảng 3.3: Thông số phần cứng của Camera Logitech C922 Pro Stream.

STT	Thông số kỹ thuật	
1	Độ phân giải máy ảnh	3 MP
2	Độ phân giải quay phim	Full HD 1080p@30fps
3		HD 720p@60fps
4	Khả năng thu phóng	Lên tới 1x
5	Tầm nhìn chéo (dFoV)	78°
6	Loại tiêu cự	Lấy nét tự động
7	Lấy nét tự động	Có
8	Loại thấu kính	Kính

9	Tự động điều chỉnh ánh sáng	Có
10	Micrô tích hợp	Âm thanh nổi
11	Các mic khử tiếng ồn	2 mic thu mọi hướng
12	Phạm vi của mic	Lên tới 1 m
13	Kết nối	USB-A
18	Chiều cao	44 mm
19	Chiều rộng	95 mm
20	Chiều dài	71 mm
22	Trọng lượng	162 g
14	Độ dài dây	1,5 m
15	Chân máy	Có
16	Kẹp phổ dụng sẵn sàng cho chân máy phù hợp với máy tính xách tay, màn hình LCD hoặc CRT	Có

Bảng 3.4: Thông số hỗ trợ thiết lập của Camera Logitech C922 Pro Stream.

STT	Thông số	Vùng giá trị
1	Độ sáng	0-255
2	Độ tương phản	0-255
3	Độ bão hòa	0-255
4	Độ nhạy sáng	0-255
5	Độ phơi sáng	3-2047
6	Độ sắc nét	0-255
7	Lấy nét	0-250
8	Thu phóng	100-500
9	Độ mờ	0-255

3.2 Các tập dữ liệu

3.2.1 Tập dữ liệu rau củ quả

Một số hình ảnh, thông tin mô tả về 7 lớp của tập dữ liệu rau củ quả đã thu thập được.

Hình 3.3: Một số hình ảnh về tập dữ liệu rau củ quả.

Bảng 3.5: Thông tin về tập dữ liệu rau củ quả.

Lớp	Nhãn	Mô tả	Train
1	carrot	Củ cà rốt	71
2	cucumber	Quả dưa chuột	94

3	daikon	Củ cải trắng	87
4	jicama	Củ đậu (củ sắn nước)	102
5	lime	Quả chanh	94
6	potato	Củ khoai tây	72
7	tomato	Quả cà chua	95
	Khác	Nhiều loại trong ảnh	78
		Tổng	**693**

3.2.2 Tập dữ liệu có mang và không mang khẩu trang

Một số hình ảnh, thông tin mô tả về 2 lớp của tập dữ liệu đối tượng có mang và không có mang khẩu trang quả đã thu thập được.

Hình 3.4: Một số hình ảnh về tập dữ liệu đối tượng có mang và không mang khẩu trang

Bảng 3.6: Thông tin về tập dữ liệu đối tượng có mang và không mang khẩu trang.

Lớp	Nhãn	Mô tả	Train
1	face_with_mask	Mang khẩu trang	4800
2	face_without_mask	Không mang khẩu trang	2846
	Khác	Nhiều loại trong ảnh	325
		Tổng	**7971**

3.2.3 Tập dữ liệu các khuôn mặt

Một số hình ảnh và thông tin về 6 lớp trong tập dữ liệu nhận dạng khuôn mặt đã thu thập được.

Hình 3.5: Một số hình ảnh về tập dữ liệu nhận dạng khuôn mặt.

Bảng 3.7: Thông tin về tập dữ liệu nhận dạng khuôn mặt.

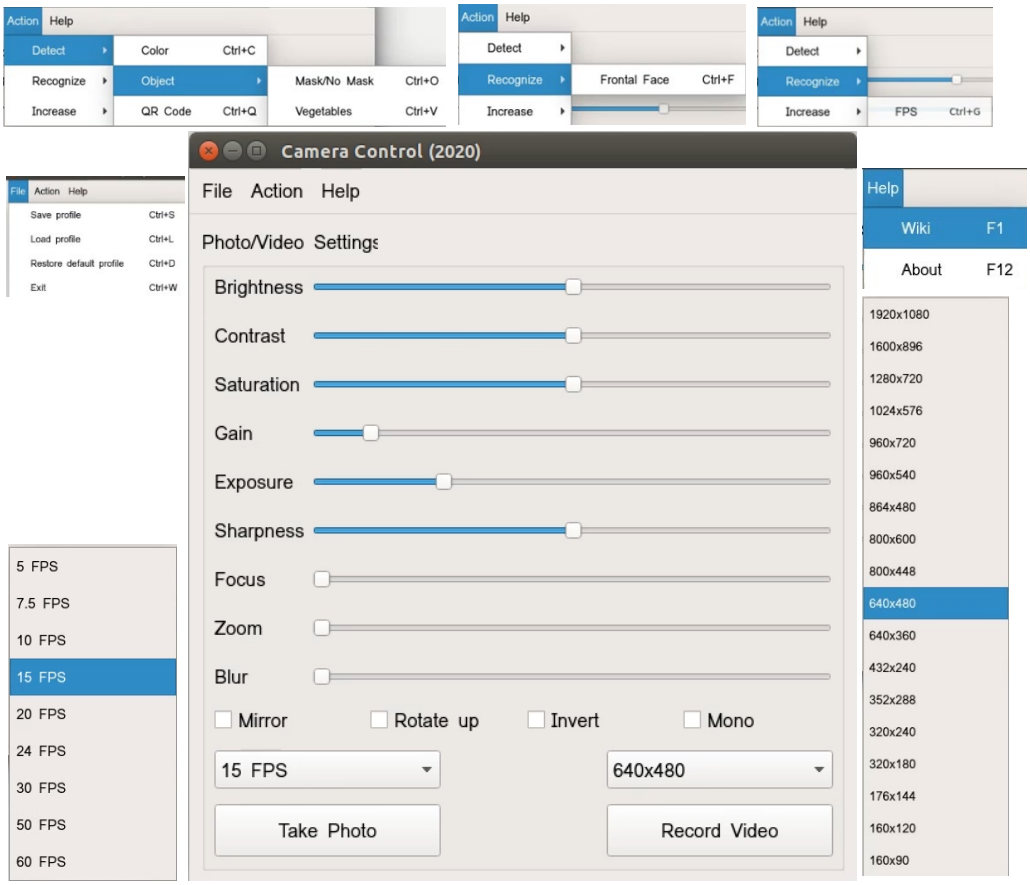
Lớp	Nhãn	Mô tả	Train
1	co_hieu	Cổ Bảo Hiếu	109
2	gia_phuc	Lê Gia Phúc	18
3	lai_dong	Cổ Lai	11
4	thanh_van	Nguyễn Thị Thanh Vân	17
5	thuy_an	Nguyễn Huỳnh Thuý An	14
6	tuyet_nhung	Cổ Thị Tuyết Nhung	11
		Tổng	**180**

3.3 Tổng quan về chương trình nghiên cứu

3.3.1 Giao diện chương trình

Giao diện được thiết kế với lớp áo từ thư viện PyQt phiên bản thứ 5. Giao diện mang phong cách hiện đại với lối thiết kế giao diện phẳng (flat UI) nhằm đem đến cảm giác gần gũi, cũng như mang lại trải nghiệm tối ưu nhất cho người sử dụng.

Giao diện chỉ có duy nhất một biểu mẫu (form) để thuận tiện cho việc thao tác và tinh chỉnh. Người dùng có thể thao tác bằng chuột hay dùng các phím tắt để kích hoạt các chức năng khác nhau.



Hình 3.6: Giao diện của chương trình nghiên cứu.

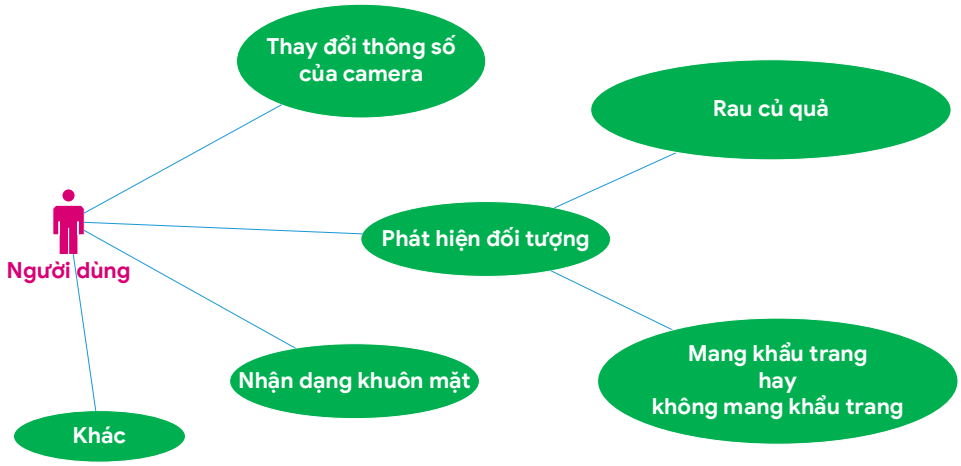
Giao diện với một thanh menu chứa các hành động như: File, Action, Help. Trong các hành động này được tổ chức thành các tính năng ẩn bên trong:

- Menu File gồm các chức năng sau: Save profile, Load profile, Restore default profile, Exit.
- Menu Action gồm các chức năng sau:
    - Detect.
        - Color.
        - Object.
            - Mask/No Mask.
            - Vegetables.
        - QR Code.
    - Recognize.
        - Frontal Face.
    - Increase.
        - FPS.
- Menu Help gồm các chức năng sau: Wiki, About.

Ngoài cùng là một thẻ menu Photo/Video Settings dùng cho việc tinh chỉnh và bộ lọc hình ảnh đầu vào và đầu ra chẳng hạn như: Brightness, Contrast, Saturation, Gain, Exposure, Sharpness, Focus, Zoom, Blur, Mirror, Rotate up, Invert, Mono, hộp kết hợp (combo box) FPS, hộp kết hợp (combo box) Resolution, nút (button) Take Photo, nút (button) Record.

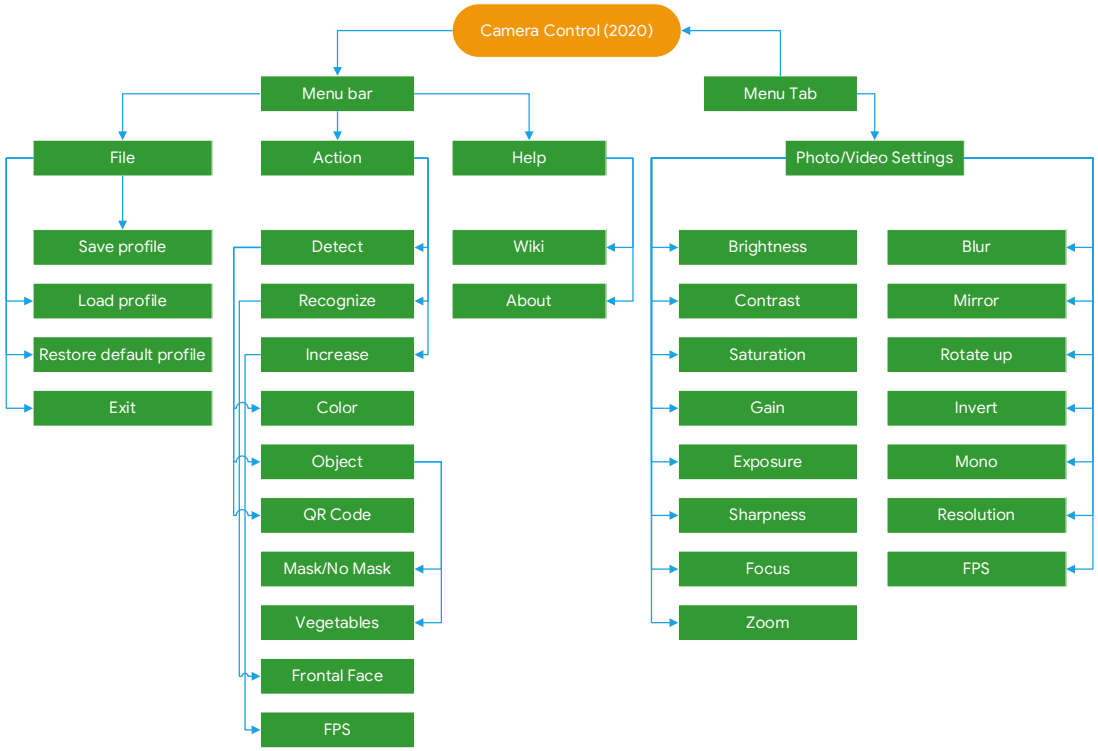
3.3.2 Sơ đồ xử lý của chương trình

Sơ đồ use case của chương trình



Hình 3.7: Sơ đồ use case của chương trình.

Sơ đồ kiến trúc của chương trình

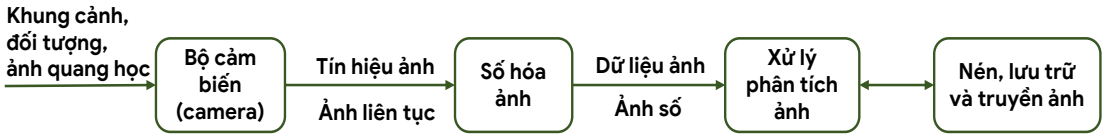


Hình 3.8: Sơ đồ xử lý của chương trình.

3.4 Các phương pháp nghiên cứu

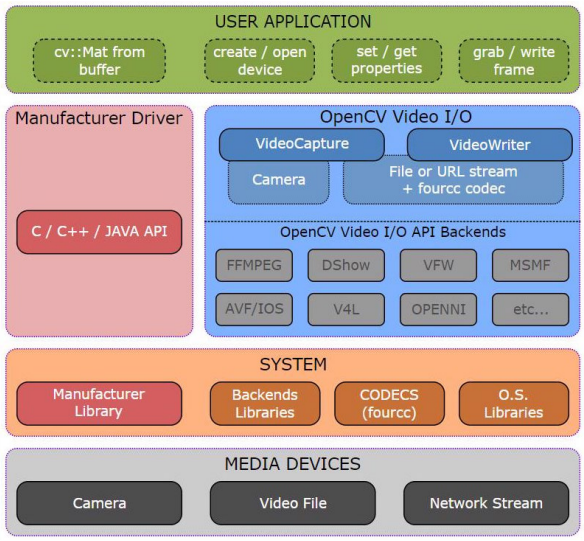
3.4.1 Phương pháp nghiên cứu camera

Xây dựng nghiên cứu xử lý hình ảnh



Hình 3.9: Kiến trúc của nghiên cứu xử lý hình ảnh cơ bản.

Nghiên cứu sử dụng chương trình xử lý hình ảnh chủ yếu từ thư viện OpenCV để camera lấy hình ảnh ở đầu vào cũng như hiển thị hình ảnh ở đầu cuối hay xử lý từng khung hình ở đầu giữa.



Hình 3.10: Kiến trúc nghiên cứu xử lý hình ảnh thông qua OpenCV [57].

3.4.1.1 Xử lý hình ảnh

Thay đổi độ sáng (brightness), độ tương phản (contrast), độ phơi sáng (exposure value – EV) và độ nhạy sáng (gain/iso)

Độ sáng là thuộc tính chủ quan, đặc trưng cho khả năng cảm nhận độ rọi (luminanace) và phụ thuộc vào độ rọi của môi trường xung quanh.

Độ tương phản biểu diễn sự thay đổi độ sáng của đối tượng so với nền. Nói một cách khác, độ tương phản là độ nổi của điểm ảnh hay vùng ảnh so với nền. Dải tần nhạy sáng là giới hạn dải mức xám xuất hiện trong ảnh và được đo bằng độ tương phản.

Độ bão hòa đặc trưng cho độ thuần khiết tương đối. Độ bão hòa phụ thuộc vào độ rộng của phổ ánh sáng và thể hiện lượng màu trắng được trộn với sắc độ.

Công thức dùng để chỉnh độ sáng dùng phép toán nhân và cộng bằng cách thay đổi giá trị α và β:

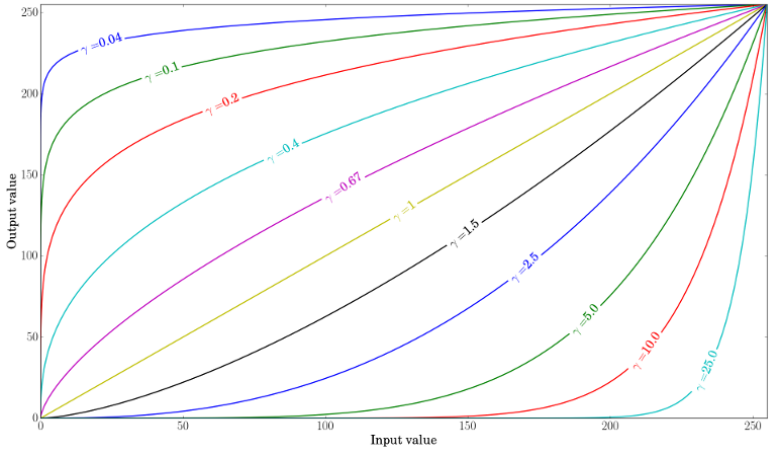
$$g(i,j) = \alpha \cdot f(i,j) + \beta \qquad (3.1)$$

- Trong đó:
  - α > 0 là tham số độ lợi (gain) hay tham số để điều chỉnh độ tương phản (contrast).
  - β là tham số độ lệch (bias) hay tham số để điều chỉnh độ sáng (brightness).
  - f(i,j) là hình ảnh đầu vào.
  - g(i,j) là hình ảnh đầu ra.
  - i và j biểu diễn cho hàng và cột nằm trong điểm ảnh.

Nếu như tăng giảm giá trị α và β lên cao có thể xảy ra một số trường hợp như nghịch đảo β sẽ cải thiện độ sáng nhưng đồng thời hình ảnh sẽ xuất hiện một lớp màn che nhẹ do độ tương phản sẽ giảm. Bên cạnh đó độ lợi α có thể làm giảm hiệu ứng này nhưng bởi vì ảnh hưởng đến đến độ bão hòa nên sẽ làm mất một số chi tiết trong vùng sáng ban đầu.

Để thay đổi độ phơi sáng và độ nhạy sáng nghiên cứu dùng phương pháp biến đổi gamma hay biến đổi luật số mũ. Biến đổi gamma có thể được sử dụng để điều chỉnh độ sáng của hình ảnh bằng cách sử dụng phép biến đổi không tuyến tính giữa các giá trị đầu vào và giá trị đầu ra được ánh xạ:

$$O = \left(\frac{I}{255}\right)^{\gamma} \times 255 \qquad (3.2)$$



Hình 3.11 Đồ thị cho các giá trị khác nhau của gamma [58].

Khi giá trị $\gamma < 1$ các vùng tối ban đầu sẽ sáng hơn và biểu đồ sẽ bị dịch chuyển sang phải và ngược lại với $\gamma > 1$.

Hình 3.12: Kết quả hình ảnh thiếu sáng và hình ảnh sau khi hiệu chỉnh gamma ($\gamma = 0.4$) [58].

Thay độ bão hòa (saturation):

Chuẩn hóa các giá trị hình ảnh BGR thành [0, 1].

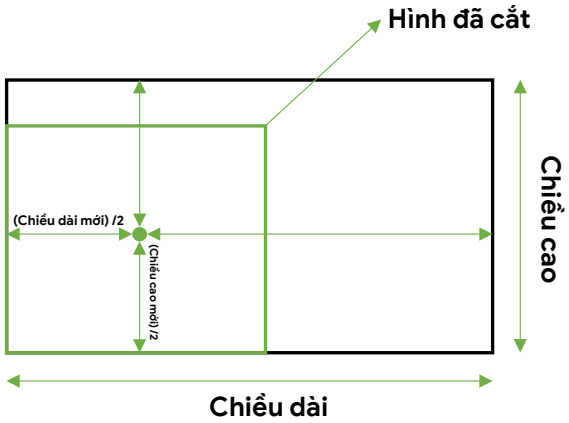
Sau đó, chuyển đổi không gian màu BGR thành không gian màu HSV.

Các thành phần độ bão hòa có thể được điều chỉnh theo chuyển đổi gamma nâng cao độ tương phản.

Cuối cùng, chuyển đổi không gián màu HSV sang không gian màu BGR.

Thay đổi thu phóng (zoom) hình ảnh:

Các chức năng phóng to và thu nhỏ được cấu trúc để hoạt động bằng cách điều chỉnh giá trị tỷ lệ.

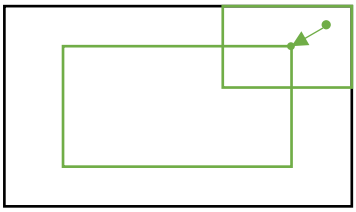


Hình 3.13: Các hiển thị một khung hình mới khi một điểm được chọn để phóng to.

Hình tròn có nghĩa là vị trí được chạm vào, chương trình nghiên cứu sẽ kiểm tra kích thước của phần trên và phần dưới và các bên dựa vào vị trí đó và coi phần nhỏ là một nửa của khung hình mới để tìm và cắt khung hình có kích thước mới. Một biến là bội số của tỷ lệ được dùng để thay đổi kích thước màn hình phù hợp.

Lấy kích thước của hình ảnh, tìm giá trị trung tâm, tính toán kích thước theo tỷ lệ, cắt và tăng kích thước về kích thước ban đầu để trả về.

Việc thu phóng khi chạm ở các cạnh có xu hướng làm biến dạng màn hình hoặc phóng to quá đột ngột, vì vậy để khắc phục điều này cần để nó tương xứng với thu phóng khi chạm.

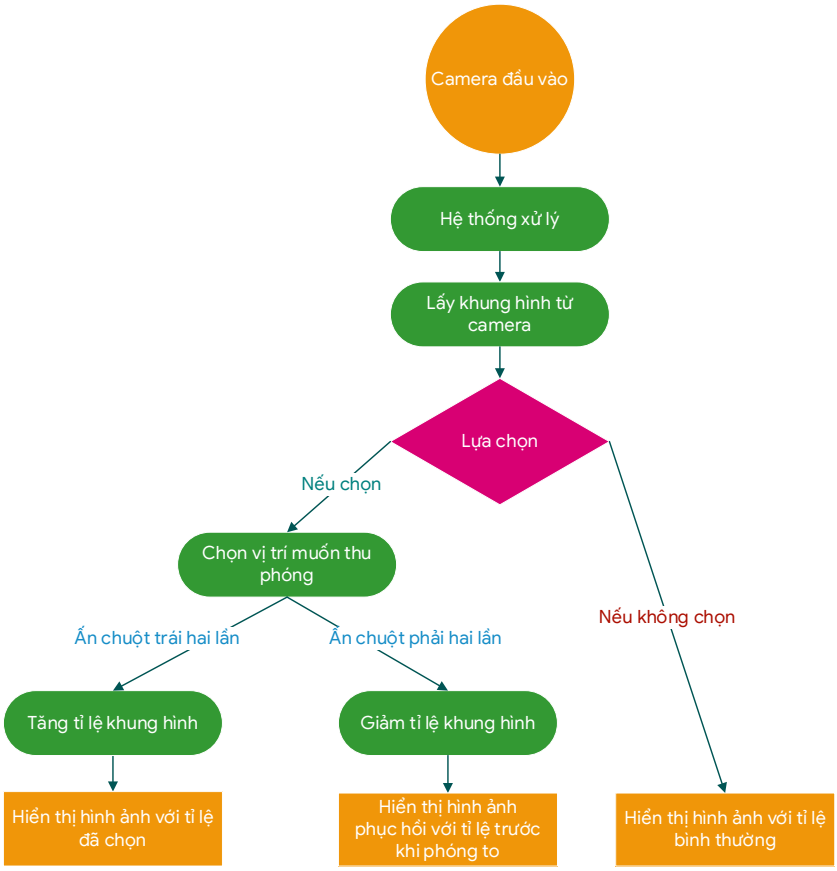


Hình 3.14: Mô tả khi ấn hai lần vào một cạnh để thu phóng.

Trong trường hợp chạm vào phần cạnh, vị trí cảm ứng được điều chỉnh theo một tỷ lệ nhất định để tránh bị méo bằng cách bật tính năng phóng đại trong khi duy trì tỷ lệ hợp lý.

Hình 3.15: So sánh trước và sau khi hình ảnh được phóng to.

Sơ đồ nghiên cứu xử lý thu phóng ảnh.



Hình 3.16: Sơ đồ nghiên cứu thu phóng ảnh.

3.4.1.2 Bộ lọc hình ảnh và các xử lý khác

Làm mờ (blur) hình ảnh:

Làm mờ hay làm mịn (Gaussian blur) là kỹ thuật làm mịn được sử dụng phổ biến nhất để loại bỏ nhiễu trong hình ảnh và video. Là bộ lọc hữu ích nhất mặc dù không phải là nhanh nhất. Bộ lọc hoạt động bằng cách xoay từng điểm trong mảng đầu vào với một hạt nhân Gaussian và sau đó tổng hợp tất cả chúng để tạo ra mảng đầu ra. Tức một hình ảnh đầu vào được đối chiếu với một kernel của Gaussian để tạo ra hình ảnh được làm mịn hay mờ dần.

Cần xác định kích thước của kernel trước khi lọc hình ảnh. Độ lệch chuẩn của phân bố Gaussian theo hướng X và Y nên được lựa chọn cẩn thận khi xem xét kích thước của kernel sao cho các cạnh của kernel gần bằng 0.

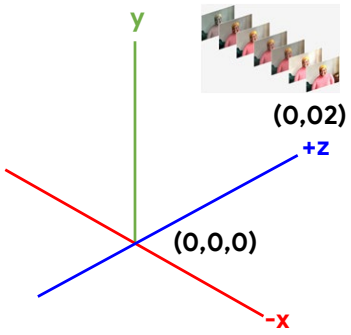
$$G_0(x,y) = Ae^{\frac{-(x-\mu_x)^2}{2\sigma_x^2}+\frac{-(y-\mu_y)}{2\sigma_y^2}} \quad (3.3)$$

- Trong đó:
  - μ là giá trị trung bình (đỉnh).
  - σ2 đại diện cho phương sai (trên mỗi biến x và y).

Hình 3.17: Hình ảnh trước khi làm mờ và sau khi làm mờ.

Lật ảnh theo gương (mirror) và xoay ảnh (rotate up):

Nghiên cứu sử dụng phương pháp lật ảnh được cung cấp từ OpenCV để lật một hình ảnh qua trục x hoặc y của nó.



Hình 3.18: Khung hình đang ở trục z và vuông góc với hai trục x, y.

Lật mảng theo một trong ba cách khác nhau (chỉ số hàng và cột dựa trên 0):

- Nếu chỉ số hàng và cột bằng 0 thì sẽ lật hình ảnh quanh trục x. Lật hình ảnh theo chiều dọc để chuyển đổi giữa nguồn gốc hình ảnh trên cùng bên trái và dưới cùng bên trái.

Hình 3.19: Lật hình ảnh theo gương (mirror).

- Nếu chỉ số hàng và cột lớn hơn 0 thì sẽ lật quanh trục y. Lật ngang của hình ảnh với sự dịch chuyển ngang tiếp theo và tính toán chênh lệch tuyệt đối để kiểm tra sự đối xứng trục dọc.
- Nếu chỉ số hàng và cột bé hơn 0 thì sẽ đảo quanh cả hai trục x - y. Lật đồng thời theo chiều ngang và chiều dọc của hình

ảnh với sự dịch chuyển tiếp theo và tính toán chênh lệch tuyệt đối để kiểm tra tính đối xứng trung tâm.

Hình 3.20: Xoay hình ảnh quay ngược chiều rotate up).

Đảo ngược màu sắc (invert) hình ảnh:

Về cơ bản, hàm Bitwise not sẽ đảo ngược các giá trị điểm ảnh. Tất cả các điểm ảnh lớn hơn 0 được đặt thành 0 và tất cả các điểm ảnh bằng 0 được đặt thành 255.

Hình 3.21: Hình ảnh trước khi đảo ngược màu.

Biến hình ảnh thành ảnh đơn sắc (mono): dùng Color space conversion:

Bộ lọc dùng chuyển đổi hệ tọa độ màu cho khung hình từ luồng camera đi vào. Sau đó không gian màu trên khung hình sẽ chuyển từ hình với hệ không gian màu BGR thành mức xám hay hình trắng đen (B&W hay Grayscale).

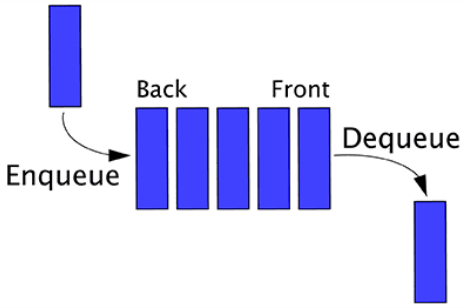
Hình 3.22: Hình ảnh BGR và hình ảnh sau khi được chuyển đổi qua mức xám (grayscale).

3.4.1.3 Xử lý tăng tốc độ khung hình

Đầu tiên để mở camera. Sau đó, bắt đầu một vòng lặp để lấy các khung hình tiếp theo từ camera. Các khung hình này sẽ được xử lý qua chương trình nghiên cứu tăng fps, thế nhưng OpenCV không cho phép vừa đọc vừa giải mã

khung hình trong luồng xử lý chính. Phương thức đọc khung hình từ OpenCV là một hoạt động chặn (blocking operation), luồng chính của hệ thống (main thread) hoàn toàn bị chặn (tức là bị đình trệ) cho đến khi khung hình tiếp theo được đọc từ camera, giải mã và lặp lại.

Bằng cách di chuyển các hoạt động xuất nhập (input/output - I/O) bị chặn này sang một luồng riêng biệt và duy trì hàng đợi các khung hình được giải mã để có thể cải thiện tốc độ xử lý FPS đáng kể. Sự gia tăng tốc độ xử lý khung hình xuất phát từ việc giảm đáng kể độ trễ, không phải chờ phương thức đọc và giải mã khung hình, thay vào đó hệ thống luôn có một khung được giả mã sẵn sàng để xử lý. Để thực hiện việc giảm độ trễ này, hệ thống chuyển việc đọc và giải mã các khung hình từ camera thành một luồng hoàn toàn riêng biệt của chương trình, giải phóng luồng chính để xử lý ảnh thực tế.



Hình 3.23: Cấu trúc dữ liệu hàng đợi và cách dữ liệu mới được xếp vào phía cuối danh sách, còn dữ liệu cũ hơn được xếp ở phía trước [59].

Cụ thể hơn là nghiên cứu sẽ sử dụng luồng (threading) và cấu trúc dữ liệu duy trì hàng đợi (queue). Và sau đó gộp tất cả lại để hiển thị ra kết quả các khung hình đã xử lý từ trước đó một cách nhanh chóng và liên tục được làm mới với thời gian giảm dần.

Sơ đồ nghiên cứu xử lý tăng tốc độ khung hình.

Hình 3.24: Sơ đồ nghiên cứu xử lý tăng tốc độ khung hình.

3.4.1.4 Phát hiện màu sắc (detect color)

Nghiên cứu xây dựng phát hiện màu sắc gồm các màu như: đen, trắng, đỏ, xanh lá, xanh dương, vàng, tím, cam, xám và sử dụng không gian màu màu HSV để tìm ra một màu nào đó là vì hệ màu này rất phù hợp với mục đích phát hiện, theo dõi đối tượng theo màu sắc.

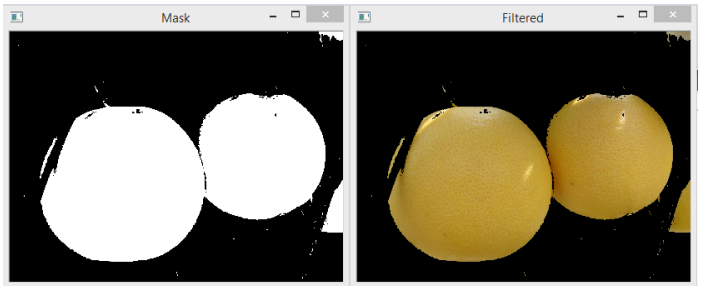
Để nghiên cứu xử lý được thì chương trình sẽ lấy khung hình từ camera đầu vào. Tiếp đó, các loại màu sắc khác nhau thì sẽ được cấu hình với các dải màu tương ứng như bảng bên dưới.

Bảng 3.8: Cấu hình các màu sắc trong vùng chứa mã màu.

STT	Màu sắc	Dải màu (color gamut)	
		Vùng ngưỡng thấp	Vùng ngưỡng cao
1	Đen	0, 0, 0	180, 255, 30
2	Trắng	159, 50, 70	180, 255, 255
3	Đỏ	0, 0, 231 0, 50, 70	180, 18, 255 9, 255, 255

4	Xanh lá	36, 50, 70	89, 255, 255
5	Xanh dương	90, 50, 70	128, 255, 255
6	Vàng	25, 50, 70	35, 255, 255
7	Tím	129, 50, 70	158, 255, 255
8	Cam	10, 50, 70	24, 255, 255
9	Xám	0, 0, 40	180, 18, 230

Sau khi chương trình quét qua tất cả các điểm ảnh (pixel) trong ô vuông. Đến đây, nghiên cứu chuyển đổi khung hình từ không gian màu BGR (OpenCV sắp kênh màu theo thứ tự BGR thay vì RGB) thành không gian màu HSV và tạo ra các mặt nạ của các điểm ảnh màu sắc. Đặt tâm của mặt nạ lên trên điểm hình ảnh cần xử lý. Thông qua bộ lọc, trích rút ra các điểm lân cận với điểm ảnh cần xử lý. Áp dụng hàm của mặt nạ lên giá trị của các điểm ảnh trong vùng lân cận. Các mặt nạ lúc này mang theo cả dải ngưỡng của màu sắc đã được cấu hình cho từng màu sắc khác nhau như bảng trên.



Hình 3.25: Mặt nạ màu được tạo ra cho màu cần phát hiện [60].

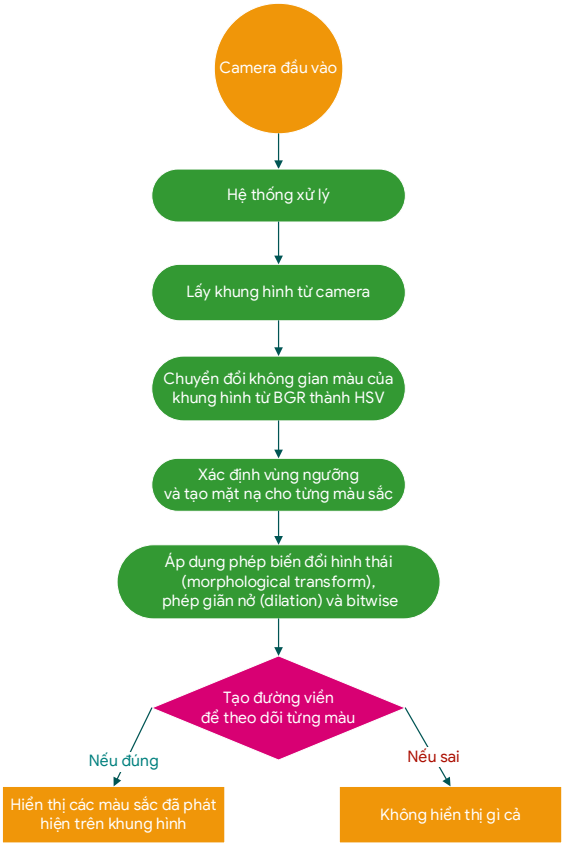
Khi xử lý trực tiếp hình ảnh từ không gian màu (RGB, HSV..) hoặc ảnh xám (gray) nghiên cứu phải tiếp nhận một số lượng lớn thông tin từ ảnh. Điều này có thể gây khó khăn cho việc xây dựng thuật toán và làm giảm tốc độ xử lý. Vì thế có một lớp toán tử biến đổi hình thái (morphological transform) kết hợp với phép giãn nở (dilation), toán tử bitwise để chuyển ảnh đầu về không gian ảnh nhị phân (chỉ gồm hai màu) nhằm làm đơn giản hóa quá trình xử lý. Điều này cũng làm nổi bật trọng tâm đối tượng, lọc bớt nhiễu trong khung hình.

Tiếp theo, là tạo đường viền (contour) để theo dõi từng màu sắc. Khi nghiên cứu so sánh ngưỡng của màu sắc chứa đường viền có các điểm ảnh trong ô vuông gần giống nhau nhất thì sẽ vẽ một hình chữ nhật có đường viền thẳng gần đúng (hình chữ nhật này không chuyển động) xung quanh khung hình đã nhị phân hóa. Cuối cùng là hiển thị kết quả là tên các màu sắc sau khi được phát hiện.

Một số các kỹ thuật nhị phân hóa ảnh ở xử lý ảnh cơ bản như đặt ngưỡng hay phát hiện viền candy (candy edge detection). Nghiên cứu sử dụng kỹ thuật đặt ngưỡng. Mục đích chủ yếu là làm nổi bật vùng trọng tâm, sau khi

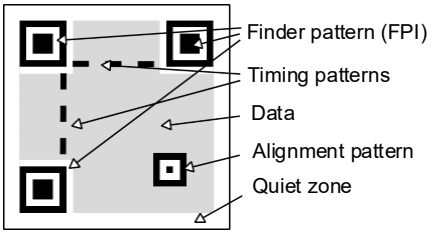
lấy được các đường bao từ khung hình để so sánh các điểm ảnh với ngưỡng dải màu.

Sơ đồ nghiên cứu xử lý khi phát hiện màu sắc.



Hình 3.26 Sơ đồ nghiên cứu xử lý phát hiện màu sắc.

3.4.1.5 Phát hiện mã QR (QR code)



Hình 3.27: Kiến trúc mã QR [61].

Mã QR đã không còn quá xa lạ với tất cả mọi người trên toàn thế giới. Mã QR đính kèm rất nhiều dữ liệu thông dụng và được lưu lành khắp nơi từ những vật phẩm được lưu trong kho, những chiếc căn cước công dân hay những linh kiện, vi xử lý nhỏ hơn. Mã QR có thể được mã hóa dưới dạng nhiều loại bảo mật khác nhau để có thể hỗ trợ việc truy xuất dữ liệu. Mã QR cũng giúp cho các bài toán về bảo mật trong phân loại và lưu trữ được giải quyết một cách đáng kể.

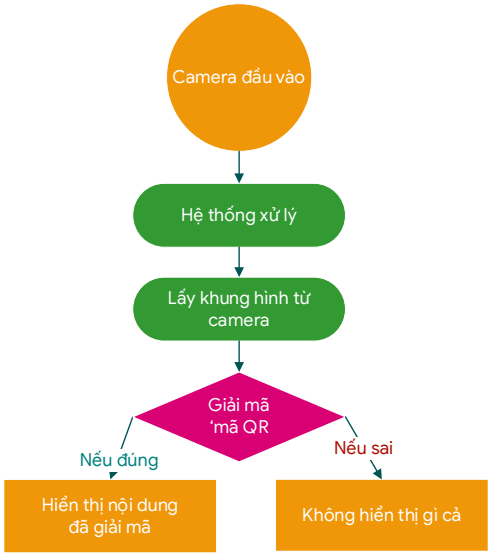
Khi một hình chứa mã QR hay hàng loạt mã QR trong hình lọt vào camera thì sẽ được nghiên cứu xử lý. Nghiên cứu sử dụng hệ thống giải mã

Zbar để giải mã các mã QR. Vì tốc độ giải mã của Zbar có thể nói là nhanh và có thể đáp ứng được bài toán xử lý thời gian thực (real-time) [61].

Với Zbar, nghiên cứu sẽ trích xuất các khung hình và kiểm tra các điều kiện như biểu tượng, hộp giới hạn, hình đa giác, hướng, chất lượng ảnh đầu vào của mã QR.

Sau đó, nghiên cứu vẽ một hộp hình chữ nhật bao quanh mã vạch được phát hiện và tiến hành giải mã thành chuỗi "utf-8" và trích xuất loại mã vạch. Cuối đó, là định dạng và hiển thị nội dung đã giải mã.

Sơ đồ nghiên cứu xử lý khi phát hiện mã QR.



Hình 3.28: Sơ đồ nghiên cứu xử lý phát hiện mã QR.

3.4.2 Phương pháp phát hiện đối tượng

3.4.2.1 Xây dựng và huấn luyện mô hình YOLOv4-Tiny

Để huấn luyện mô tình, nghiên cứu đã tinh chỉnh một loạt các tham số trong file cấu hình trong tệp YOLOv4-Tiny như bảng 3.9 và tệp Makefile như bảng 3.10.

Bảng 3.9: Thông số cần thay đổi trên tệp cấu hình yolov4-tiny.cfg để huấn luyện YOLOv4-Tiny.

STT	Nội dung	Giá trị	
		Rau củ quả	Mang và không mang khẩu trang
1	subdivisions	64	64
2	batches	14000	6000
3	steps	11200, 12600	4800, 5400
4	classes	7	2
5	filters	36	21

Bảng 3.10: Thông số cần thay đổi trên tệp Makefile.

STT	Nội dung	Giá trị

1	GPU	1
2	CUDNN	1
3	OPENCV	1

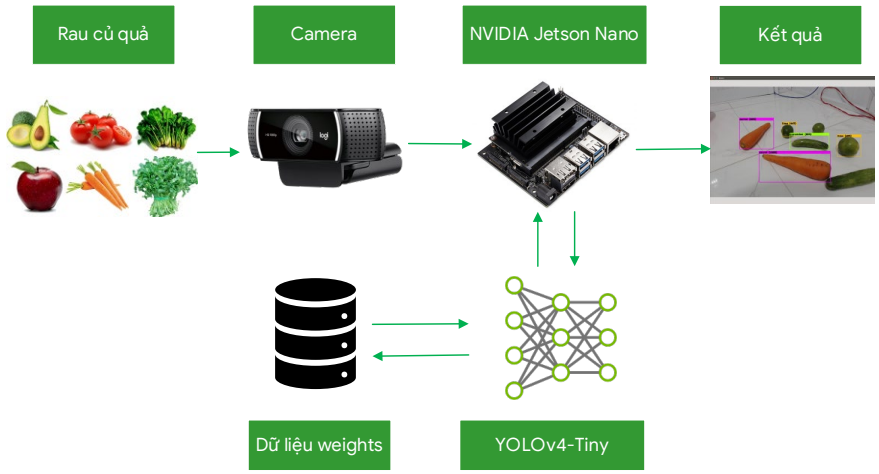
Ngoài ra, còn có hai tệp tin chứa đường dẫn ảnh: một phần dùng để huấn luyện train.txt chứa 80% tập dữ liệu và phần còn lại để kiểm tra valid.txt chứa 20% tập dữ liệu, cùng với tệp chứa tên lớp yolo.names.

Bảng 3.11: Số lượng dữ liệu trong hai tệp train.txt và valid.txt.

STT	Nội dung	Giá trị	
		Rau củ quả	Mang và không mang khẩu trang
1	train	555	6377
2	valid	138	1594
Tổng		**693**	**7971**

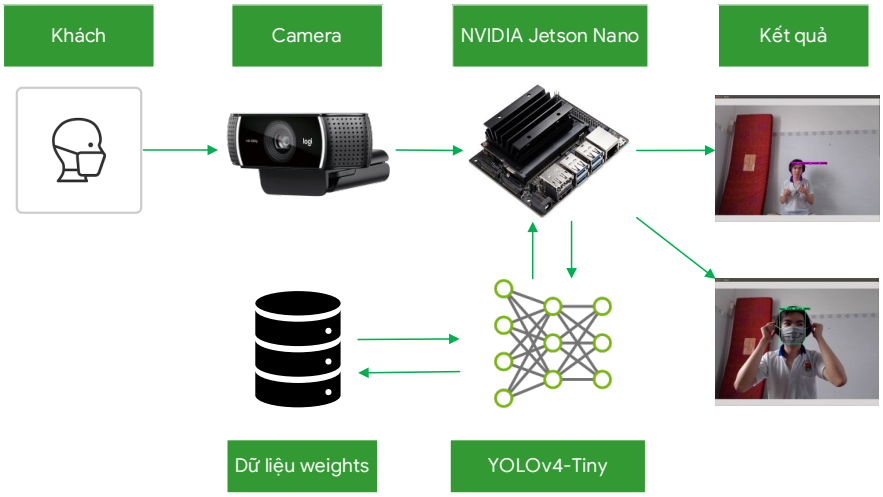
Nghiên cứu sử dụng một máy chủ miễn phí được cung cấp từ nhà Google là Google Colab để huấn luyện YOLOv4-Tiny trong 12 giờ và kết nối lưu trữ vào thư mục của Google Drive. Trong số các của mô hình Darknet được lưu lại trong thư mục sau lưu (backup) được sinh ra và tự động lưu lại sau mỗi 1000 lần lặp. Darknet sẽ tự động lưu mô hình tốt với phần mở rộng best.weights bằng cách dựa vào hiệu suất trên bộ xác thực sau mỗi chu kỳ. Darknet cũng sẽ lưu tệp huấn luyện cuối cùng từ quá trình huấn luyện với phần tên tệp có định dạng last.weights. Sau cùng, chương trình huấn luyện Darknet sẽ chọn tệp weights YOLOv4-Tiny tốt nhất sau khi huấn luyện.

Kiến trúc nghiên cứu YOLOv4-Tiny cho tập dữ liệu rau củ quả:



Hình 3.29: Kiến trúc nghiên cứu YOLOv4-Tiny cho tập dữ liệu rau củ quả.

Kiến trúc nghiên cứu YOLOv4-Tiny cho tập dữ liệu đối tượng có mang và không mang khẩu trang.



Hình 3.30: Kiến trúc nghiên cứu YOLOv4-Tiny cho tập dữ liệu đối tượng có mang và không mang khẩu trang.

3.4.2.2 Mô hình YOLOv4-Tiny kết hợp DeepStream

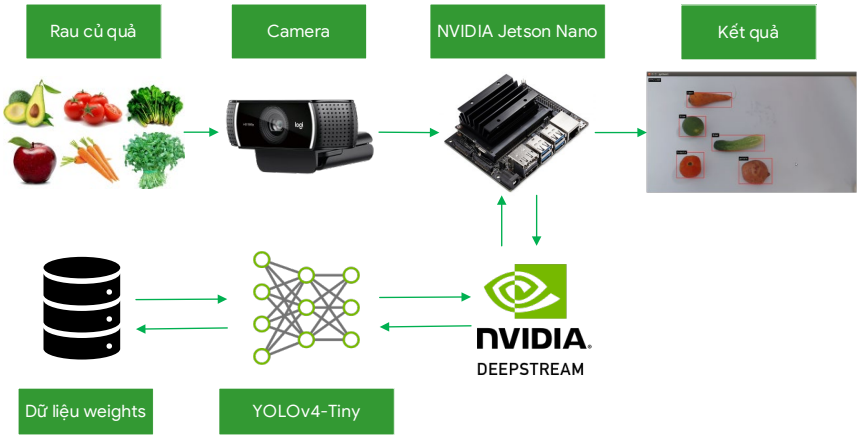
Trước khi nghiên cứu sử dụng kiến trúc NVIDIA DeepStream [62] với Tensor Cores FP32, nghiên cứu này cũng đã chỉnh sửa lại một số thông tin cần thiết trong tệp cấu hình config_infer_primary.txt để phù hợp với mô hình YOLOv4-Tiny. Hơn thế nữa là không cần phải chuyển đổi tệp weight mà vẫn có thể thanh chiếu đến NVIDIA TensorRT [62].

Bảng 3.12: Thông số cần thay đổi trên tệp cấu hình DeepStream config_infer_primary.txt.

STT	Nội dung	Giá trị
1	model-color-format	0
2	custom-network-config	yolov4-tiny.cfg
3	model-file	yolov4-tiny.weights
4	labelfile-path	yolo.names
5	model-engine-file	model_b1_gpu0_fp32.engine
6	batch-size	1
7	network-mode	32
8	num-detected-classes	2
9	Number of consecutive batches to be skipped	0
10	IOU threshold	0.7
11	Score threshold	0.25

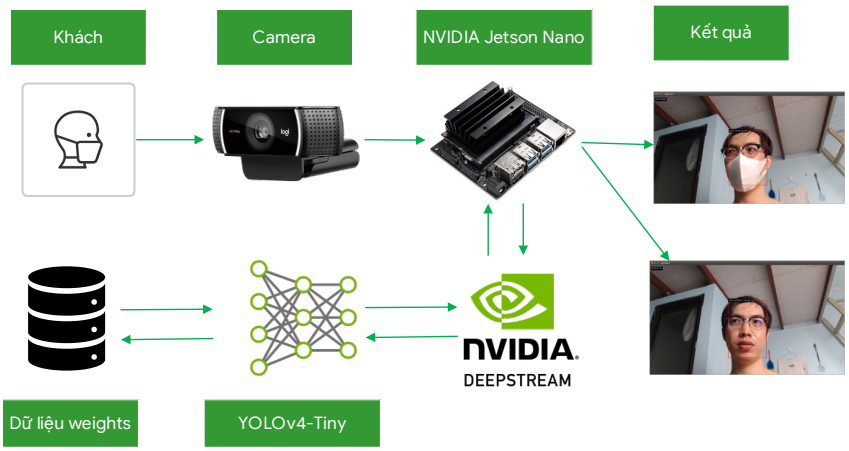
Kiến trúc mô hình nghiên cứu đã được tăng tốc độ xử lý khi kết hợp DeepStream.

- Kiến trúc nghiên cứu YOLOv4-Tiny kết hợp DeepStream cho tập dữ liệu rau củ quả.



Hình 3.31: Kiến trúc nghiên cứu YOLOv4-Tiny kết hợp DeepStream cho tập dữ liệu rau củ quả.

- Kiến trúc nghiên cứu YOLOv4-Tiny kết hợp DeepStream cho tập dữ liệu đối tượng có mang và không mang khẩu trang.



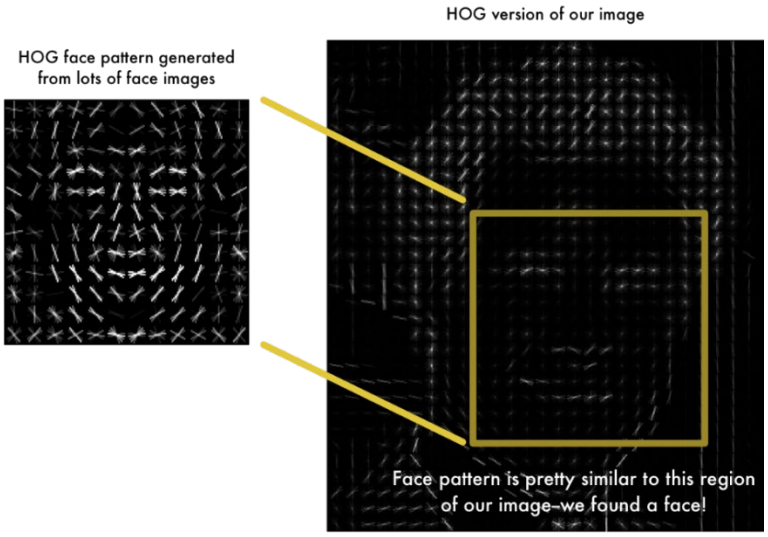
Hình 3.32: Kiến trúc nghiên cứu YOLOv4-Tiny kết hợp DeepStream cho tập dữ liệu đối tượng có mang và không mang khẩu trang.

3.4.3 Phương pháp nghiên cứu nhận dạng khuôn mặt

3.4.3.1 Xây dựng mô hình với thư viện face_cognition và Dlib

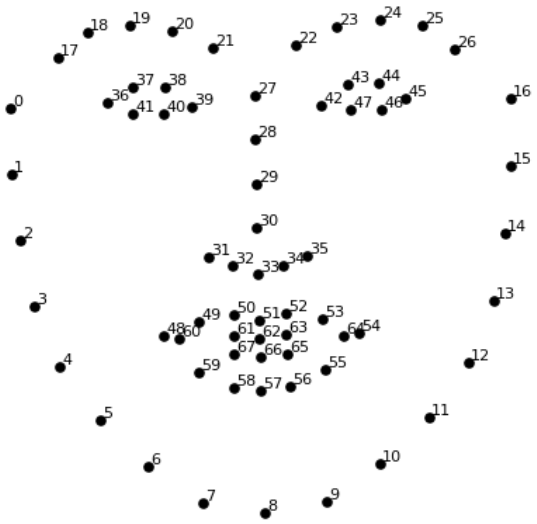
Tạo các mã hóa dữ liệu (encodings) (hay embeddings) của các khuôn mặt trong dữ liệu đã thu thập ở trên.

Đầu tiên là trích xuất các face ROIs. Để phát hiện và trích xuất các khuôn mặt có trong ảnh với HOG thì hình ảnh gốc sẽ được chuyển thành hình ảnh trắng đen để ghi lại các đặc điểm chính của khuôn mặt trên hình ảnh dù bất kể độ sáng của hình ảnh như thế nào.



Hình 3.33: Hình ảnh khi xử lý qua HOG [63].

Tiếp đến, khi có các face ROIs. Các ROIS sẽ được đưa qua mạng nơ ron để lấy các mã hóa dữ liệu. Sử dụng thuật toán ước tính mốc khuôn mặt để đưa ra 68 điểm cụ thể (hay điểm mốc) tồn tại trên mọi khuôn mặt như đỉnh cằm, mép ngoài của mỗi mắt, mép trong của mỗi lông mày, v.v… 68 điểm mốc mà này sẽ xác định trên tất cả khuôn mặt.

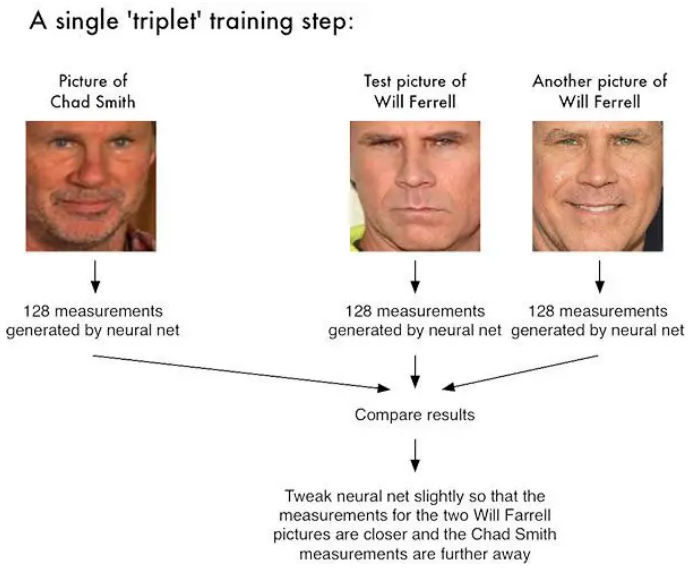


Hình 3.34: 68 mốc trên khuôn mặt từ Dlib được tạo ra bởi Brandon Amos của CMU, người làm việc trên OpenFace [63].

Đây là quá trình học cách ánh xạ từ hình ảnh khuôn mặt đến không gian Euclide nhỏ gọn nơi các khoảng cách tương ứng trực tiếp với số đo độ giống nhau của khuôn mặt. Khi không gian này đã được tạo ra, các tác vụ như nhận dạng khuôn mặt, xác minh và phân cụm có thể dễ dàng thực hiện bằng cách sử dụng các kỹ thuật tiêu chuẩn với nhúng dưới dạng véc tơ đặc trưng.

Hình 3.35: Thư viện face_recognition tạo ra véc tơ đặc trưng số có giá trị thực 128-d trên mỗi khuôn mặt [64].

Mạng nơ-ron học cách tạo ra 128 phép đo cho mỗi người một cách đáng tin cậy. Bất kỳ, 10 hình ảnh khác nhau của cùng một người sẽ cho ra số đo gần giống nhau.



Hình 3.36: Quá trình đào tạo để tạo ra 128 phép đo cho mỗi khuôn mặt [63].

Dữ liệu ở đây không cần huấn luyện lại từ đầu mạng để tạo mã hóa mà sử dụng lại mô hình để tái huấn luyện nhằm tạo ra các khuôn mặt nhúng (face embeddings).

Trong khi tái huấn luyện sẽ lưu lại các mã và nhãn được lưu vào một tệp dữ liệu pickle.

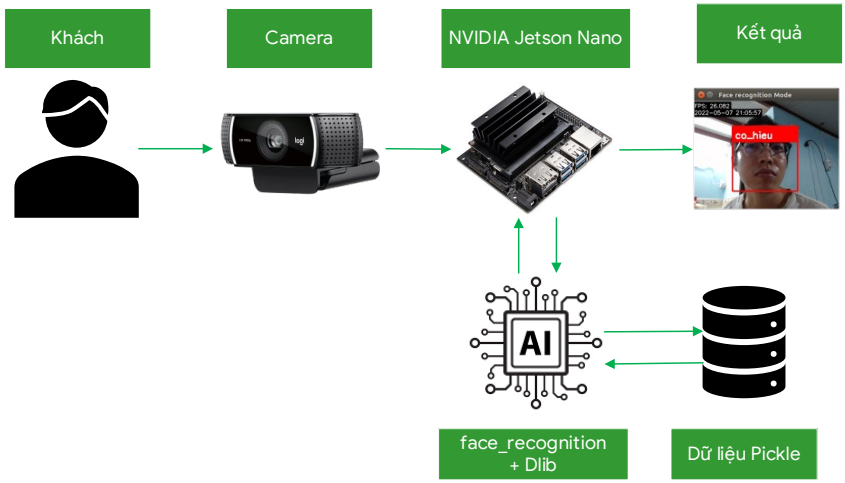
Khi sử dụng với ngưỡng khoảng cách 0,6, mô hình Dlib sẽ đạt được độ chính xác lên đến 99,38%. Tiêu chuẩn nhận dạng khuôn mặt LFW tiêu chuẩn, có thể so sánh với các phương pháp nhận dạng khuôn mặt hiện đại khác kể từ tháng 2 năm 2017 [65]. Độ chính xác có nghĩa là khi được hiển thị với một cặp hình ảnh khuôn mặt, công cụ sẽ xác định chính xác xem cặp đó thuộc về cùng một người hay là của những người khác nhau 99,38% thời gian.

Xác định vị trí khuôn mặt có hai phương pháp khác nhau có thể dùng là HOG và CNN (với mô hình Max-Margin Object Detection). Nghiên cứu vì

ưu tiên tốc độ xử lý nên đã sử dụng mô hình HOG thay vì sử dụng mô hình CNN để cho kết quả chính xác hơn.

Cấu trúc nghiên cứu bao gồm:

- Camera: đầu vào hình ảnh trực tiếp ở thời gian thực của một vị khách chưa qua xử lý.
- NVIDIA Jetson Nano: máy tính xử lý với các thuật toán AI sẽ phát hiện khuôn mặt của đối tượng và xử lý ảnh.
- Thư viện face_recognition với Dlib: nhận dạng đối tượng và lấy dữ liệu từ pickle để xử lý hình ảnh từ camera.
- Dữ liệu pickle: nơi lưu trữ cơ sở dữ liệu của nghiên cứu các khuôn mặt đã được tái huấn luyện.
- Kết quả: hiển thị thông tin về khuôn mặt của đối tượng đã được định danh.



Hình 3.37: Kiến trúc nghiên cứu xử lý nhận dạng khuôn mặt.

Nguyên lý hoạt động:

- Dữ liệu hình ảnh được chuyển tiếp từ camera và phát liên tục video trực tiếp cho đến khi kết thúc tiến trình xử lý của nghiên cứu bằng OpenCV. Sau đó, NVIDIA Jetson Nano sẽ gọi thư viện face_recognition với Dlib để tiến hành xử lý hình ảnh bằng cách tải cả dữ liệu đã huấn luyện trước đó và được mã hóa lưu trữ trong tệp pickle.
- Dữ liệu sẽ được giải mã ra và tải lên, nhận dạng khuôn mặt và trả về kết quả. Tiến trình mất hơn 5-20 giây.
    - Nếu khuôn mặt trùng với dữ liệu đã huấn luyện trước thì sẽ hiển thị thông tin của khuôn mặt đó.
    - Nếu khuôn mặt không trùng hoặc không có trong data training thì sẽ hiện ra thông tin là “Mặt không hợp lệ”.

3.4.3.2 Tái huấn luyện mô hình nhận dạng khuôn mặt

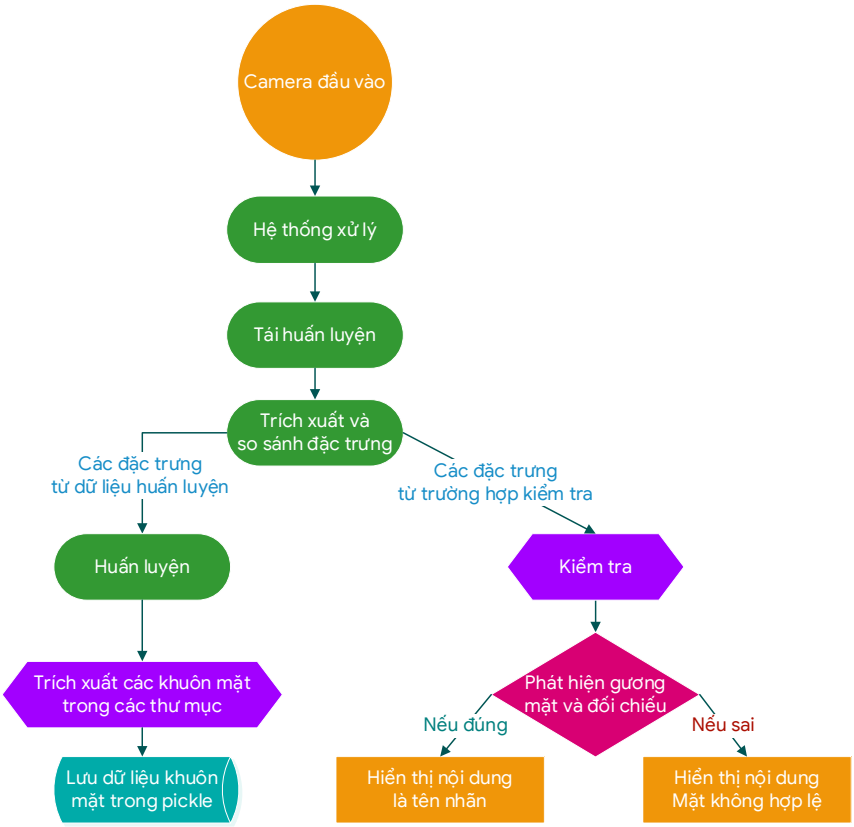
Thư viện face_recognition được xây dựng dựa trên thư viện Dlib sẽ được tái huấn luyện mô hình thay vì phải huấn luyện mô hình từ đầu như các mô hình mạng khác. Tái huấn luyện mô hình đảm nhận vai trò rất quan trọng trong nghiên cứu. Nó sẽ phân tích hình ảnh được gửi từ phía người dùng và lưu trữ các hình ảnh sau khi đã phân tích vào một cơ sở dữ liệu rồi mã hóa để phục vụ cho việc nhận dạng đối tượng sau này.

Hoạt động: nghiên cứu có 2 cách để thêm dữ liệu cho đối tượng nhận dạng là:

- Thêm mới trực tiếp đối tượng bằng cách nghiên cứu sẽ tiến hành tạo thư mục chứa tên đối tượng và them các hình ảnh có liên quan đến đối tượng vào thư mục vừa tạo. Kế đó, chạy lại chương trình huấn luyện dữ liệu và tạo ra một tệp dữ liệu pickle mới.
- Cập nhật lại dữ liệu cho đối tượng đã tồn tại trong cơ sở dữ liệu bằng cách mở thư mục chứa thông tin của đối tượng và thêm các hình ảnh mới nhất vào. Tiếp đến, chạy lại chương trình huấn luyện dữ liệu và tạo ra một tệp dữ liệu pickle mới.

3.4.3.3 Sơ đồ mô hình nhận dạng khuôn mặt

Sau khi tái huấn luyện mô hình thì đề tài đã thiết kế một sơ đồ mô hình nhận dạng khuôn mặt để phù hợp với bài toán đặt ra, cũng như thiết bị hỗ trợ.



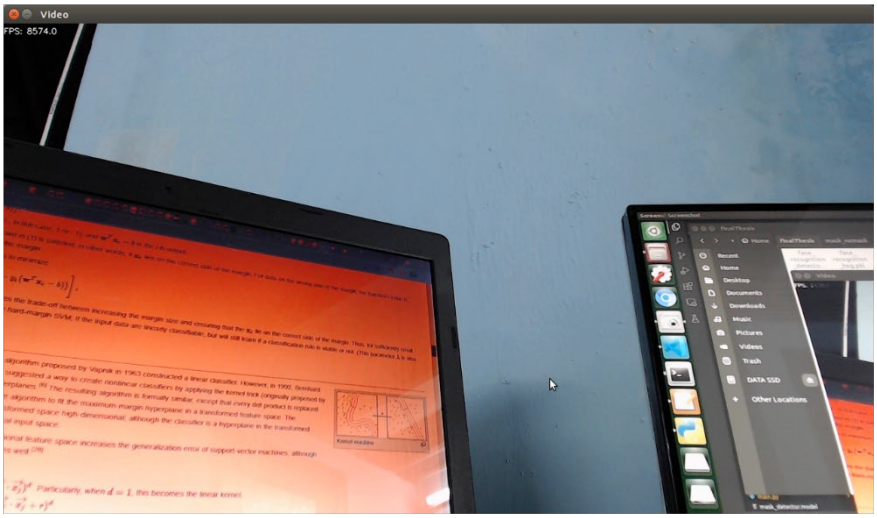
Hình 3.38: Sơ đồ nghiên cứu xử lý nhận dạng khuôn mặt.

CHƯƠNG 4: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

4.1 Kết quả thực nghiệm

4.1.1 Khi xử lý tăng tốc độ khung hình

Kết quả thực nghiệm sau khi chương trình thực thi với chức năng tăng fps ở khung hình 1280x720 thì fps đạt được là 8574.0 như hình 4.1.



Hình 4.1: Kết quả thực nghiệm khi xử lý tăng FPS ở khung hình 1280x720.

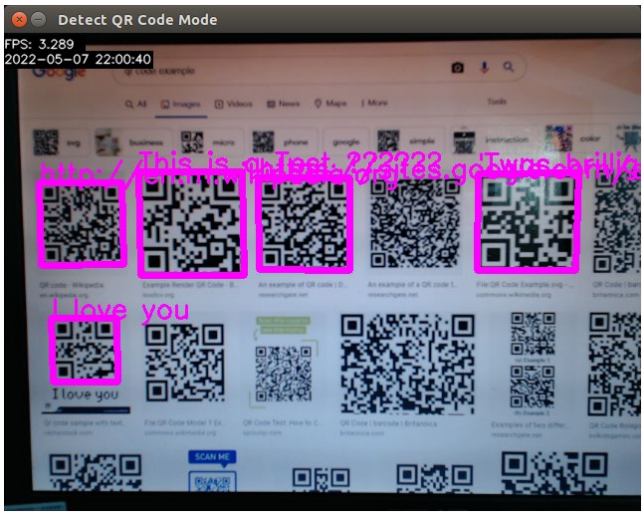
4.1.2 Khi phát hiện màu sắc

Kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện màu sắc ở khung hình 1280x720 thì đã phát hiện được các màu sắc như hình 4.2.

Hình 4.2: Kết quả thực nghiệm khi phát hiện màu sắc ở khung hình 1280x720.

4.1.3 Khi xử lý phát hiện mã QR

Kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện mã QR và giải mã nội dung mã QR ở khung hình 1280x720 thì đạt được 3fps.



Hình 4.3: Kết quả thực nghiệm khi xử lý phát hiện mã QR ở khung hình 1280x720.

4.1.4 Khi đối tượng là rau củ quả với tốc độ fps mặc định là 30fps.

Kết quả thực nghiệm trên trên tập dữ liệu rau củ quả, tốc độ fps mặc định là 30fps trong khoảng cách 70cm từ camera đến đối tượng.

Bảng 4.1: Kết quả thực nghiệm trên trên tập dữ liệu rau củ quả, với tốc độ fps mặc định là 30fps.

STT	Lớp	True Positive	False Negative	False Positive
1	carrot	16	4	2
2	cucumber	11	9	5
3	daikon	18	2	1
4	jicama	11	9	3
5	lime	20	0	0
6	potato	18	2	1
7	tomato	20	0	0
Tổng		**114**	**26**	**12**

Dựa vào kết quả đánh giá ở trên, các chỉ số của mô hình có thể được tính như sau:

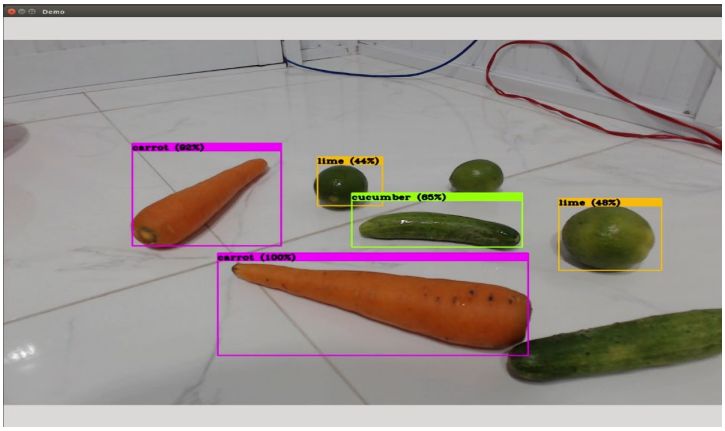
$$\text{Prediction} = \frac{TP}{TP + FP} = \frac{114}{114 + 12} \approx 0{,}90 \quad (4.1)$$

$$\text{Recall} = \frac{TP}{TP + FN} = \frac{114}{114 + 26} \approx 0{,}81 \quad (4.2)$$

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} = 2 \times \frac{0{,}90 \times 0{,}81}{0{,}90 + 0{,}81} \approx 0{,}85 \quad (4.3)$$

4.1.4.1 Hình thực nghiệm đạt được khi xử lý trên GPU không kết hợp DeepStream

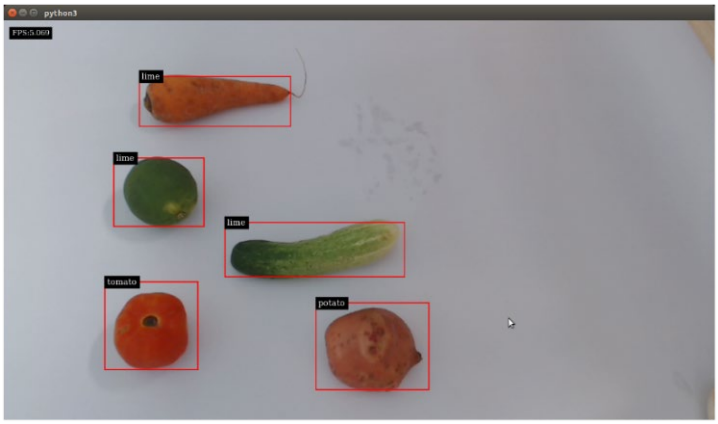
Kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện rau củ quả, khi xử lý trên GPU không kết hợp DeepStream.



Hình 4.4: Kết quả thực nghiệm phát hiện rau củ quả trên GPU không kết hợp DeepStream.

4.1.4.2 Hình thực nghiệm đạt được khi xử lý trên GPU kết hợp DeepStream

Kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện rau củ quả khi xử lý trên GPU có kết hợp DeepStream.



Hình 4.5: Kết quả thực nghiệm xử lý phát hiện rau củ quả trên GPU kết hợp DeepStream.

4.1.5 Khi đối tượng là người có mang và không mang khẩu trang với tốc độ fps mặc định là 30fps

Kết quả thực nghiệm trên trên tập dữ liệu có mang và không mang khẩu trang, tốc độ fps mặc định là 30fps trong khoảng cách 60cm từ camera đến người.

Bảng 4.2: Kết quả thực nghiệm trên trên tập dữ liệu có mang và không mang khẩu trang, với tốc độ fps mặc định là 30fps.

STT	Lớp	True Positive	False Negative	False Positive
1	face_with_mask	17	3	1
2	face_without_mask	16	4	0
	Tổng	**33**	**7**	**1**

Dựa vào kết quả đánh giá ở trên, các chỉ số của mô hình có thể được tính như sau:

$$\text{Prediction} = \frac{TP}{TP + FP} = \frac{33}{33 + 1} \approx 0{,}97 \quad (4.4)$$

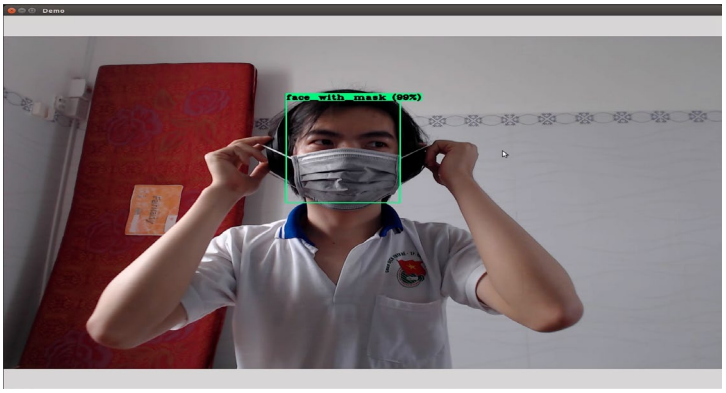
$$\text{Recall} = \frac{TP}{TP + FN} = \frac{33}{33 + 7} \approx 0{,}83 \quad (4.5)$$

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} = 2 \times \frac{0{,}97 \times 0{,}83}{0{,}97 + 0{,}83} \approx 0{,}895 \quad (4.6)$$

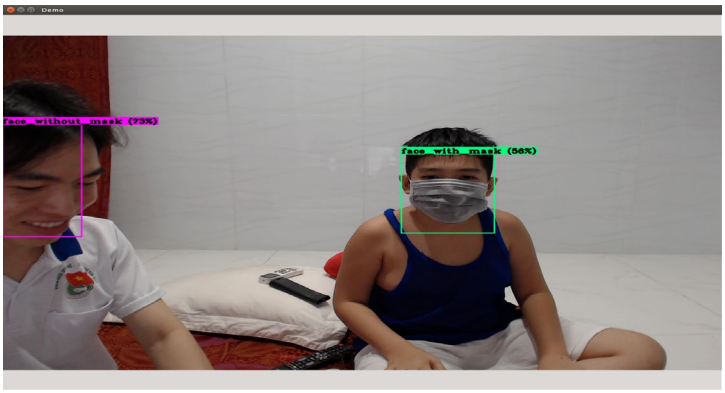
4.1.7.1 Hình thực nghiệm đạt được khi xử lý trên GPU không kết hợp DeepStream

Một vài kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện đối tượng mang hay không mang khâu trang khi xử lý trên GPU không kết hợp DeepStream.

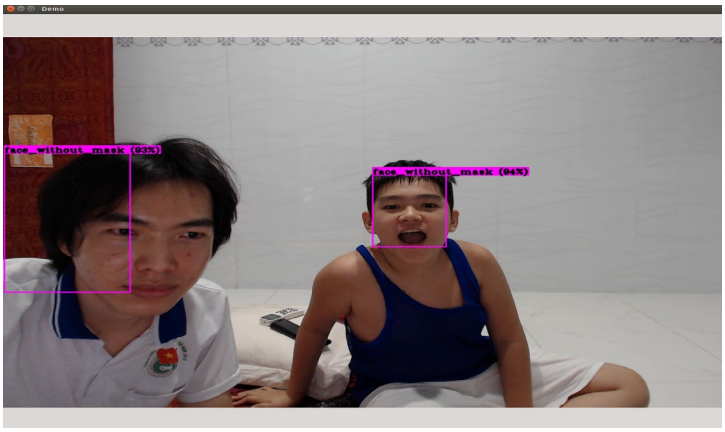
Hình 4.6: Kết quả thực nghiệm khi xử lý trên GPU không kết hợp DeepStream khi người mang khẩu trang ở xa.



Hình 4.7: Kết quả thực nghiệm khi xử lý trên GPU không kết hợp DeepStream khi người mang khẩu trang ở gần.



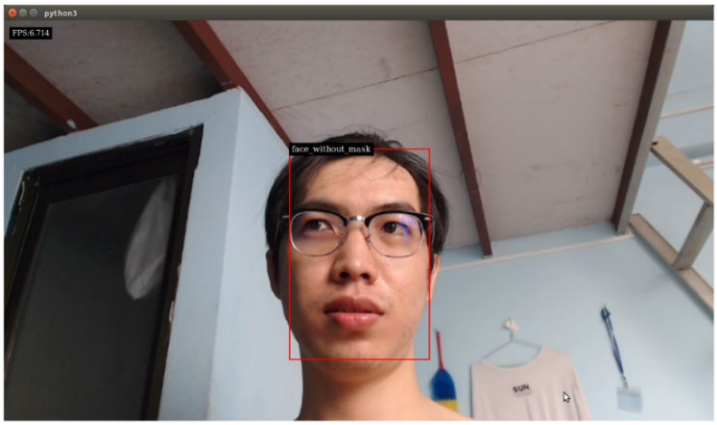
Hình 4.8: Kết quả thực nghiệm khi xử lý trên GPU không kết hợp DeepStream khi có một người mang khẩu trang và một người không mang khẩu trang.



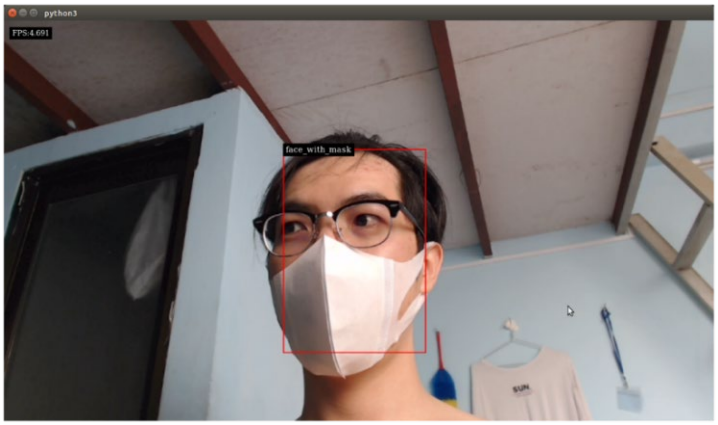
Hình 4.9: Kết quả thực nghiệm khi xử lý trên GPU không kết hợp DeepStream khi có hai người không mang khẩu trang.

4.1.5.2 Hình thực nghiệm đạt được khi xử lý trên GPU kết hợp DeepStream

Một vài kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện đối tượng mang hay không mang khâu trang khi xử lý trên GPU có kết hợp DeepStream.



Hình 4.10: Kết quả thực nghiệm khi xử lý trên GPU kết hợp DeepStream khi có một người không mang khẩu trang.



Hình 4.11: Kết quả thực nghiệm khi xử lý trên GPU kết hợp DeepStream khi có một người mang khẩu trang.

4.1.6 Khi so sánh hiện đối tượng xử lý trên GPU có và không có DeepStream hỗ trợ.

Kết quả thực nghiệm khi nghiên cứu xử lý trên GPU có và không có DeepStream hỗ trợ.

Bảng 4.3: Kết quả thực nghiệm trên GPU có và không có DeepStream hỗ trợ.

STT	GPU	Khung hình	FPS	Độ chính xác
1	Không kết hợp DeepStream	1280x720	1	85%-98%
2	Có kết hợp DeepStream	1280x720	4.6-10	85.7%-85.9%

4.1.7 Nhận dạng khuôn mặt đạt được sử dụng trích xuất đặc trưng HOG và Linear SVM với tolerance là 0.6

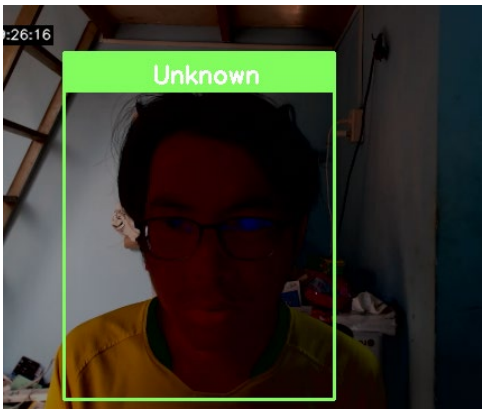
4.1.7.1. Ở khung hình 320x240

Kết quả thực nghiệm sau khi chương trình thực thi với chức năng nhận dạng khuôn mặt khi sử dụng trích xuất đặc trưng HOG ở khung hình 320x240 với tốc độ fps mặc định là 30fps, thì fps đạt được là 26.074.

Hình 4.12: Kết quả thực nghiệm nhận dạng khuôn mặt trên khung hình 320x240 với tốc độ 30fps.

4.1.7.2. Với một nhãn trên nhiều khung hình

Một vài các kết quả thực nghiệm sau khi chương trình thực thi với chức năng phát hiện khuôn mặt trên một lớp trên nhiều khung hình khác nhau, với tốc độ fps mặc định là 30fps và ba mức sáng ở các giá trị khác nhau.



Hình 4.13: Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên khung hình 640x360 với mức sáng 0-18.

Bảng 4.4: Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên các khung hình khác nhau với mức sáng 0-18.

STT	Khung hình	Khoảng cách (cm)	FPS đạt được	True Positive	False Negative	False Positive
1	1920x1080	50-100	1-2.5	0	20	0
2	1280x720	50-100	4-5.2	0	20	0
3	960x540	50-62	6.4-7.3	0	20	0
4	720x480	50-62	9.7-15.3	0	20	0
5	720x360	50-62	10.3-16.4	0	20	0
6	640x480	50-62	9.0-14.7	0	20	0
7	640x360	50-62	7.8-16.4	0	20	0
8	480x360	50-62	8.3-16.4	0	20	0
9	320x240	40-47	18-30	0	20	0
Tổng				**0**	**180**	**0**

Dựa vào kết quả đánh giá ở trên, các chỉ số của mô hình có thể được tính như sau:

$$\text{Prediction} = \frac{TP}{TP+FP} = \frac{0}{0+0} \approx 0 \quad (4.7)$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{0}{0+180} \approx 0 \quad (4.8)$$

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} = 2 \times \frac{0 \times 0}{0 + 0} \approx 0 \quad (4.9)$$

Hình 4.14: Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên các khung hình 640x360 với mức sáng 100.

Bảng 4.5 : Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên các khung hình khác nhau với mức sáng 100.

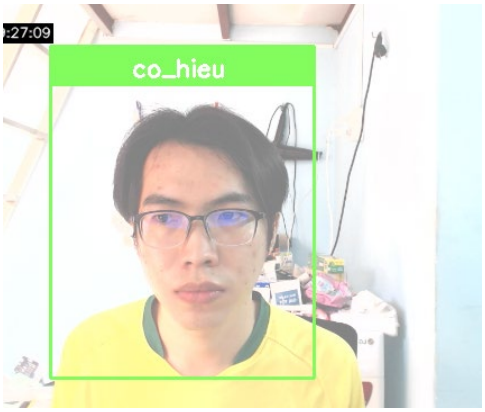
STT	Khung hình	Khoảng cách (cm)	FPS đạt được	True Positive	False Negative	False Positive
1	1920x1080	50-100	2-2.6	18	0	2
2	1280x720	50-100	3-4.4	17	0	3
3	960x540	50-62	4.5-5.6	15	0	5
4	720x480	50-62	4.8-7.5	13	0	7
5	720x360	50-62	6.8-13.4	12	0	8
6	640x480	50-62	7-15.1	14	0	4
7	640x360	50-62	17-21	13	0	7
8	480x360	50-62	20-24	13	0	5
9	320x240	40-47	24-26	17	0	3
Tổng				**132**	**0**	**44**

Dựa vào kết quả đánh giá ở trên, các chỉ số của mô hình có thể được tính như sau:

$$\text{Prediction} = \frac{TP}{TP+FP} = \frac{128}{128+44} \approx 0{,}75 \quad (4.10)$$

$$\text{Recall} = \frac{TP}{TP+FN} = \frac{132}{132+0} \approx 1 \quad (4.11)$$

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} = 2 \times \frac{0{,}75 \times 1}{0{,}75 + 1} \approx 0{,}857 \quad (4.12)$$



Hình 4.15: Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên các khung hình 640x360 với mức sáng 255.

Bảng 4.6 : Kết quả thực nghiệm khi xử lý nhận dạng khuôn mặt trên các khung hình khác nhau với mức sáng 255.

STT	Khung hình	Khoảng cách (cm)	FPS đạt được	True Positive	False Negative	False Positive
1	1920x1080	50-100	1-2.2	20	0	0
2	1280x720	50-100	3-4.1	16	0	4
3	960x540	50-62	4-5	14	0	6
4	720x480	50-62	5-6	14	0	6
5	720x360	50-62	5.4-7.2	15	0	5
6	640x480	50-62	4.8-7	16	1	4
7	640x360	50-62	5-11	16	3	6
8	480x360	50-62	12-16	16	1	4
9	320x240	40-47	18-24	16	1	6
Tổng				**143**	**6**	**41**

Dựa vào kết quả đánh giá ở trên, các chỉ số của mô hình có thể được tính như sau:

$$\text{Prediction} = \frac{TP}{TP+FP} = \frac{143}{143+41} \approx 0{,}777 \quad (4.13)$$

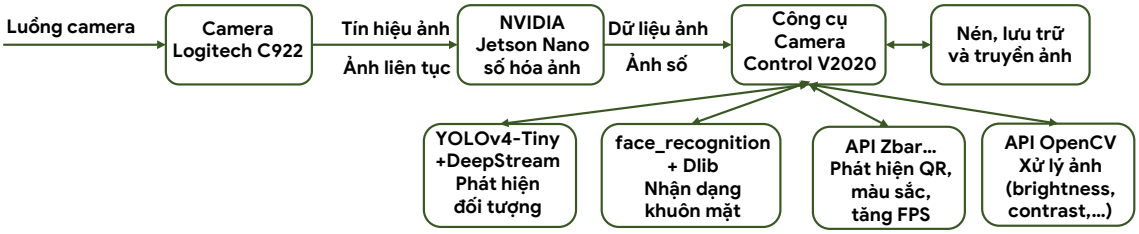
$$\text{Recall} = \frac{TP}{TP+FN} = \frac{143}{143+6} \approx 0{,}96 \quad (4.14)$$

$$F_1 = 2 \times \frac{\text{Prediction} \times \text{Recall}}{\text{Prediction} + \text{Recall}} = 2 \times \frac{0{,}75 \times 0{,}978}{0{,}75 + 0{,}978} \approx 0{,}859 \quad (4.15)$$

4.2 Đánh giá

4.2.1 Kết quả đạt được

Sơ đồ kết quả sau một quá trình dài của luận văn.



Hình 4.16: Kết quả đạt được của nghiên cứu.

Dựa trên các bảng số liệu kết quả ở trên đã nghiên cứu được. Với thiết bị NVIDIA Jetson Nano kết hợp với Camera Logitech C922 Pro Stream thì nghiên cứu khuyến khích để tối ưu nhất cho nghiên cứu xử lý và có thể đạt được tốc độ quét tối đa cho:

- Hạng mục nghiên cứu phát hiện đối tượng, tăng tốc độ khung hình, phát hiện màu sắc, phát hiện mã QR nên sử dụng ở độ phân giải (resolution) 1280x720 hay bất kỳ kích thước khung hình nào để tối ưu cho việc hiển thị vì tốc độ xử lý không thay đổi.
- Hạng mục nhận dạng khuôn mặt, nên sử dụng ở độ phân giải (resolution) ở 320x240 với sáng mặc định là 100 hoặc mức sáng tối đa là 255 trong khoảng cách từ 40cm đến 100cm để có thể cho ra kết quả chính xác và tốc độ xử lý cũng được tối ưu.

Dữ liệu đào tạo phù hợp có thể là một phần quan trọng trong quá trình huấn luyện để đạt được kết quả lý tưởng.

4.2.2 Đánh giá mô hình thực nghiệm.

4.2.2.1 Phát hiện đối tượng

Từ các kết quả đánh giá ở trên của hai tập dữ liệu, nghiên cứu đã có thể dự đoán được nguyên nhân dẫn đến những sai số của mô hình:

- Khi đối tượng được quét vào camera thì kích thước của đối tượng nhỏ hơn so với kích thước ảnh được huấn luyện hoặc ảnh đầu vào lớn hơn ảnh được huấn luyện.
- Mô hình nhận dạng bị kém và dẫn đến sai là do có một vài lớp bị dữ liệu bị hạn chế.
- Bên cạnh đó, cũng có hạn chế từ mô hình YOLOv4-Tiny. Vì YOLOv4-Tiny là mô hình rút gọn của mô YOLOv4.
- Nghiên cứu cũng sử dụng DeepStream để tham chiếu và gọi trực tiếp kiến trúc TensorRT nên cũng ảnh hưởng một phần.

4.2.2.2 Nhận dạng khuôn mặt.

Từ các kết quả đánh giá ở trên của tập dữ liệu với các khung hình khác nhau, mức sáng khác nhau, khoảng cách khác nhau với cùng tốc độ fps; nghiên

cứu đã có thể dự đoán được nguyên nhân dẫn đến những sai số của mô hình như sau:

- Khi đối tượng được quét vào camera thì kích thước của đối tượng nhỏ hơn so với kích thước ảnh được huấn luyện hoặc ảnh đầu vào lớn hơn ảnh được huấn luyện do tỉ lệ khung hình được cài đặt ngay lúc đầu vào.
- Mô hình không thể nhận dạng được khuôn mặt chính xác nếu một khuôn mặt nhìn về các hướng khác nhau hay các góc khác nhau so với camera mà không quay về hướng chính diện (hướng nhìn thẳng vào camera) vì mô hình sử dụng nhận dạng khuôn mặt chính diện 2 chiều.
- Mô hình dễ nhận dạng nhầm với các khuôn mặt có quan hệ huyết thống trong gia đình vì tập dữ liệu sử dụng các khuôn mặt có quan hệ trong gia đình với nhau là chủ yếu.
- Với mức sáng khác nhau thì độ chính xác của mô hình phát hiện khuôn mặt cũng khác nhau.
- Với khoảng cách khác nhau và khung hình khác nhau thì cũng ảnh hưởng đến mô hình và làm cho mô hình phát hiện chính xác hoặc phát hiện sai.

Mô hình sẽ hoạt động tốt trong môi trường ánh sáng đầy đủ, mức sáng ở cấu hình mặc định và khoảng cách đặt vị trí camera trong phạm vi nghiên cứu ở trên thì kết quả sẽ cho độ chính xác sẽ dao động trong khoảng 85,7% - 85,9% và nhận diện được khuôn mặt ít bị sai xót.

4.2.3 Đánh giá chung

Nghiên cứu về quản lý camera vẫn chưa đạt đến hiệu quả tối ưu nhất cho chương trình nghiên cứu. Những sự cố về việc đọc dữ liệu camera từ máy tính dẫn đến gián đoạn hệ thống làm cho nghiên cứu không thể tinh chỉnh được hình ảnh. Bên cạnh đó, vẫn còn khá nhiều lỗi liên quan đến việc xử lý hình ảnh, xử lý khung hình, xử lý các bộ lọc.

Nghiên cứu chưa tối ưu được camera về đường truyền tín hiệu, do nghiên cứu sử dụng usb camera gắn ngoài mà không dùng MPI CSI camera do tốc độ truyền tải của usb có giới hạn.

Nghiên cứu xử lý phát hiện màu sắc phát hiện không chính xác màu sắc khi ở các điều kiện ánh sáng khác nhau.

Nghiên cứu quét mã hay phát hiện mã chỉ hiện thị rồi biến mất tức thời.

OpenCV có thể nói là thư viện tuyệt vời trong quá trình nghiên cứu, phù hợp với hệ thống nhúng với hiệu suất cao, hiệu quả, chất lượng hiển thị hình ảnh sắc nét.

CHƯƠNG 5: KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

5.1 Kết luận

5.1.1 Tổng kết

Nghiên cứu đã mang tạo ra chương trình xử lý thuật toán AI vào máy tính nhúng. Qua đó, tạo ra một chương trình khai thác máy tính cở nhỏ nhưng vẫn có thể chạy được các thuật toán ML hay DL có hỗ trợ GPU.

Mặc dù nghiên cứu còn nhiều nhược điểm về tính bảo mật, độ chính xác, độ nhạy, v.v…. Bên cạnh đó, những ý tưởng của nghiên cứu ngày càng được hoàn thiện hơn trong tương lai gần. Mặc dù, nghiên cứu hiện tại chỉ mới là bước đầu trong việc tạo dựng một chương trình có khả năng tích hợp nhiều thuật toán AI và cũng như nhiều bài toán xử lý hình ảnh và đạt đến sự ổn định.

5.1.2 Ưu điểm

Nghiên cứu tập trung vào xử lý camera và áp dụng các mô hình mạng của trí tuệ nhân tạo vào trong phần cứng thiết bị nhúng với giá thành rẻ mà vẫn có thể xử lý các thuật toán AI như phát hiện đối tượng với YOLOv4-Tiny kết hợp với DeepStream có thể phát hiện các vật thể có kích thước nhỏ ở gần camera, định danh khuôn mặt với face_recognition và dlib nhanh nhờ vào sử dụng trích xuất đặc HOG với Linear SVM, quản lý camera hiệu quả với OpenCV, v.v...

NVIDIA Jetson Nano là một thiết bị hỗ trợ hữu ích cho các ứng dụng tiên tiến đòi hỏi sức mạnh xử lý với hệ sinh thái phát triển mạnh mẽ của NVIDIA và đi kèm theo điều đó là sự hỗ trợ từ cộng đồng người dùng của NVIDIA.

5.1.3 Nhược điểm

Một vài tính năng mà nghiên cứu chưa thể tích hợp vào chương trình, mà phải chạy rời rạc do những tính năng này được viết dưới dạng từng mô-đun riêng lẻ.

Những tính năng mà nghiên cứu đã xây dựng cũng như thử nghiệm vẫn chưa thực sự hoàn thiện vẫn còn nhiều lỗi và độ ổn định vẫn chưa được ổn định khi hoạt động.

Hiệu suất của chương trình nghiên cứu cũng bị phụ thuộc vào thời tiết nếu sử dụng thiết bị ngoài trời hay nhiệt độ phòng do quạt làm mát của phần cứng có giới hạn.

Ngoài những điều trên, nghiên cứu cũng bị giới hạn khi giải quyết các bài toán AI như hình dạng, vận tốc, khoảng cách, v.v… khi phát hiện các loại đối tượng hay nhận dạng khuôn mặt.

Camera không thể khôi phục lại cài đặt ban đầu khi thay đổi các chức năng như tăng giảm độ sáng, độ bão hòa, v.v… mà phải cần đến việc khởi động lại cả nguyên một thiết bị máy tính.

Nghiên cứu xử lý hình ảnh vẫn chưa hoạt động được hết hoàn toàn, thi thoảng vẫn gặp sự cố không thay đổi được.

Nghiên cứu tăng tốc khung hình chưa có tận dụng được để tối ưu các hạng mục nghiên cứu khác mặt dù kết quả mang lại rất nổi trội.

Nghiên cứu nhận diện màu sắc vẫn chưa nhận diện chính xác màu.

Nghiên cứu quét mã hay phát hiện mã QR vẫn chưa có thể lưu lại nội dung đã quét được vào một số tệp lưu trữ bất kỳ trên máy tính.

Chất lượng những video chất lượng ở đầu vào sẽ được quyết định với phần cứng camera khi cắm vào.

Điều kiện ánh sáng bị thiếu sáng hay ánh sáng quá sáng cũng ảnh hưởng đến chất lượng video đầu vào và sẽ làm cho chương trình nghiên cứu không thể nhận dạng chính xác.

Khoảng cách đối tượng ở xa hay gần cũng ảnh hưởng kết quả chính xác hay ít chính xác, thậm chí là không thể nhận dạng được của chương trình nghiên cứu.

Những đối tượng mang khẩu trang như học sinh, sinh viên, hành khách, bệnh nhân, công nhân viên, hay người bình thường với hình dáng của chiếc khẩu trang với một thiết kế mới hay chưa có trong dữ liệu huấn luyện cũng sẽ làm cho chương trình nghiên cứu không thể nhận dạng được.

Những đối tượng như rau củ quả với kích thước dị thường hay màu sắc khác nhau hoặc chưa có trong dữ liệu huấn luyện cũng sẽ làm cho chương trình nghiên cứu không thể nhận dạng được.

Chương trình nghiên cứu sẽ không nhận dạng được nếu có vật hay bàn tay người cố tình che lại camera.

Kết quả thực nghiệm của nghiên cứu xử lý phát hiện đối tượng trên GPU với sự hỗ trợ từ DeepStream thì vẫn chưa hiển thị được độ chính xác như kết quả thực nghiệm khi xử lý trên GPU.

Việc dùng kiến trúc DeepStream để tham chiếu trực tiếp đến kiến trúc TensorRT cũng đã làm giảm độ chính xác của mô hình.

Nhận dạng khuôn mặt còn tồn tại hàng loạt các vấn đề sau:

- Thi thoảng, chương trình nghiên cứu không thể quét hết tất cả các khuôn mặt do hạn chế của camera, cũng như không thể phát hiện những khuôn mặt ở quá xa camera.
- Chương trình nghiên cứu khó nhận dạng được khuôn mặt nếu một khuôn mặt đang quay về hướng không chính diện (hướng nhìn thẳng vào camera) mà nhìn về các hướng khác hay trong điều kiện ánh sáng kém.
- Tìm ra đặc trưng duy nhất của khuôn mặt để phân biệt mọi người với nhau nếu như người đó bị một số sự cố trên khuôn mặt như: mắt to hay

nhỏ hơn sau khi đi thức dậy hay có đeo len hoặc bị tai nạn, mặt dày hay nhỏ khi họ trang điểm (tô khối trên khuôn mặt) hay đi phẫu thuật thẩm mỹ về và khác hơn so với mẫu huấn luyện chưa được cập nhật lại...

- Chương trình nghiên cứu khó nhận dạng các khuôn mặt khi người đó cố tính dùng các chậu cây để che khuôn mặt, mang mắt kính có màu khác mắt kính trong suốt (hoặc màu trắng trong), hay cố tình che lại camera bằng bất cứ vật thể nào hay bằng bàn tay.
- Kết quả thực nghiệm khi xử lý nhận diện khuôn mặt vẫn chưa hiển thị được độ chính xác như những mô hình nhận diện khuôn mặt khác.
- Mô hình sử dụng trích xuất đặc trưng HOG và Linear SVM dẫn đến việc làm cho kết quả nhận dạng có thể ít chính xác hơn mô hình sử dụng thuật toán CNN.
- Khuôn mặt khi xuất hiện, hướng xuất hiện của khuôn mặt khác với vị trí đặt camera giám sát hay góc đặt camera hoặc do khuôn mặt đã được tô khối trang điểm. Từ đó, dẫn đến một vài thành phần như trên khuôn mặt như: mắt, mũi, miệng có thể bị che khuất hoặc ảnh chụp bị biến dạng do góc chụp làm ảnh hưởng đến kết quả và giảm đi khả năng nhận dạng chính xác khuôn mặt.
- Góc chụp của camera rất quan trọng vì nó quyết định phạm vi nhận dạng đối tượng cũng như khung hình chứa khuôn mặt khi chụp.
- Khuôn mặt xuất hiện trong khung hình bị che khuất bởi một số vật thể hoặc một hay một vài đối tượng khác cũng sẽ làm cho chương trình nghiên cứu không thể nhận dạng được hay nhận dạng sai.
- Cảm xúc và biểu cảm trên khuôn mặt khi xuất hiện trong khung hình cũng ảnh hưởng lớn đến việc rút trích đặt trưng từ khuôn mặt.
- Hình ảnh trong điều kiện thiếu ánh sáng hoặc thừa ánh sáng sẽ khó tránh được sự ảnh hưởng lớn đến việc rút trích đặt trưng và nhận dạng so với hình ảnh ở điều kiện ánh sáng vừa phải.
- Camera với độ phân giải cao sẽ cho ra những hình ảnh với chất lượng cao và ngược lại camera với chất lượng thấp sẽ làm cho hình ảnh chỉ dừng lại ở chất lượng thấp. Những hình ảnh ở chất lượng cao, chất lượng tốt ở đầu vào sẽ làm cho việc nhận dạng dễ dàng hơn. Còn những hình ảnh chất lượng đầu vào thấp, mờ, vỡ thì sẽ ảnh hưởng đến việc chương trình nghiên cứu không nhận dạng được hay nhận dạng sai.

5.1.4 Những đóng góp mới

Nghiên cứu đã xây dựng một thiết bị nhúng NVIDIA Jetson Nano từ những tính năng cơ bản cho đến việc áp dụng các thuật toán cao cấp từ AI vào nghiên cứu và thực nghiệm.

Nghiên cứu tích hợp nhiều ứng dụng nhỏ và xử lý để tạo ra một ứng dụng tổng hợp phục vụ cho việc quản lý camera và quản lý GPU trên máy tính để xử lý các thuật toán AI ở thời gian thực.

Nghiên cứu cũng tối ưu hóa tốc độ xử lý giải quyết các bài toán dùng ML hay DL với các hạng mục nghiên cứu phát hiện đối tượng và nhận diện khuôn mặt dù tốc độ cải thiện không đáng kể.

Nghiên cứu có thể mở ra nhiều hi vọng cho việc sử dụng những thiết bị nhúng đầy sức mạnh từ hệ sinh thái của NVIDIA.

5.2 Hướng phát triển

Nghiên cứu sẽ phát triển thêm một số bộ lọc để xử lý HDR cho hình ảnh.

Nghiên cứu sẽ hiển thị được độ chính xác (accurracy) trong hai hạng mục nghiên cứu là phát hiện đối tượng và nhận dạng khuôn mặt.

Chương trình sẽ chạy đa tác vụ (multiple threads), tách tiến trình (process) xử lý camera riêng biệt ra với các tiến trình khác để chạy ngầm và có thể chuyển đổi liên tục mượt mà.

Chương trình sẽ được tích hợp thêm hệ quản trị cơ sở dữ liệu (MySQL, MongoDB hay MindsDB) để lưu trữ những dữ liệu cần thiết cho việc sử dụng hay tái huấn luyện dữ liệu cũ và dữ liệu mới hoặc lưu trữ các nội dung đã quét được hay phát hiện mã QR hay lưu các dữ liệu cho các hạng mục nghiên cứu khác.

Nghiên cứu sẽ có thể cho phép các thiết bị camera gắn vào và tự động nhận các cấu hình có liên quan đến camera.

Nghiên cứu sẽ hỗ trợ nhiều thiết bị camera chạy cùng một thời điểm và xử lý cùng lúc. Nghiên cứu sẽ tối ưu với camera MPI CSI hỗ trợ cảm biến IR (công nghệ quét độ sâu của hình ảnh) hay camera cảm biến nhiệt độ hoặc camera có độ nét cao để nhận dạng hay phát hiện mục tiêu tăng độ chính xác.

Nghiên cứu về thu phóng hình ảnh sẽ được cải thiện về chất lượng hình ảnh.

Nghiên cứu về phát hiện màu sắc sẽ được tối ưu độ chính xác với thuật toán KNN.

Nghiên cứu về mã QR sẽ tiến thêm một bước nữa, với việc có thể ấn vào đường dẫn truy cập và hiển thị đường dẫn đó trên trình duyệt web khi ấn vào nội dung mà mã QR đã phát hiện ra hay quét được.

Tất cả các hạng mục nghiên cứu sẽ được tăng tốc độ khung hình để cải thiện về tốc độ FPS và tối ưu hiệu suất.

Chương trình sẽ được quản lý tốt hơn trên một ứng dụng từ thiết bị điện thoại thông minh có hệ điều hành Android hay iOS.

Nghiên cứu với hạng mục phát hiện đối tượng là rau củ quả sẽ được tiến hành phần tiếp theo với đầu bếp AI với phần việc cung cấp cơ sở dữ liệu chứa các món ăn, nguyên liệu, cách chế biến và tích hợp xử lý ngôn ngữ tự nhiên (Natural Language Processing - NLP). Việc tích hợp thêm xử lý ngôn ngữ tự nhiên và đặc biệt là xử lý tiếng Việt vào mục đích phát hiện giọng nói khi gia chủ hay đầu bếp nấu ăn không chuyên nghiệp muốn yêu cầu nấu một món ăn nào đó. Tiếp đó, đầu bếp AI sẽ trả lời và hiển thị tất cả các món ăn này lên ứng dụng hay trên trình duyệt trên điện thoại có hệ điều hành Android hay iOS.

Nghiên cứu với hạng mục nhận dạng sẽ được cải thiện về cơ sở dữ liệu và có thể được tích hợp sâu vào những hệ thống điểm danh cho học sinh, sinh viên hay hệ thống chấm công nhân viên hoặc sẽ được tích hợp vào những cửa ra vào cho các nông trại, trang tại hay những nơi cần khóa bảo mật ít an toàn, dễ dàng thi công, dễ sử dụng và tiết kiệm.

Nghiên cứu sẽ dùng mô hình mạng Swin Transformer, mô hình mạng YOLOv6 hay mô hình mạng PP-YOLOE vào hạng mục phát hiện đối tượng hoặc cải tiến lại thuật toán của mô hình YOLOv4-Tiny.

Các hạng mục viết rời rạc dưới dạng mô-đun sẽ được kết nối vào chương trình nghiên cứu.

Các nghiên cứu về thiết bị nhúng như NVIDIA Jetson Nano sẽ được tích hợp với flycam hay drone để giải quyết các bài toán phát hiện và nhận dạng những mục tiêu ở xa hơn.

TÀI LIỆU THAM KHẢO

[1] Akhil Kumar, Arvind Kalia, Aayushi Kalia, ETL-YOLO v4: A face mask detection algorithm in era of COVID-19 pandemic, 2022.

[2] Adam Geitgey, https://medium.com/@ageitgey/build-a-face-recognition-system-for-60-with-the-new-nvidia-jetson-nano-2gb-and-python-46edbddd7264, 2020.

[3] Nataliya Boyko, Oleg Basystiuk, Nataliya Shakhovska, Performance Evaluation and Comparison of Software for Face Recognition, based on Dlib and Opencv Library, 2018.

[4] Shrikant Jagannath Patro, Prof. Nisha V M, Real Time Video Analytics for Object Detection and Face Identification using Deep Learning, 2019.

[5] Md. Rakibul Hasan, Effectiveness of Face Recognition System with Human Face Detection Learning Technique for Students, 2020.

[6] Shashank Reddy Boyapally, Facial Recognition and Attendance System using dlib and face_recognition libraries, 2021.

[7] Seok-Yeol Heo, Kang Min Kim, Wan-Jik Lee, Design and Implementation of Visitor Access Control System using Deep learning Face Recognition, 2021.

[8] Jianming Zhang, Manting Huang, Xiaokang Jin và Xudong Li, A Real-Time Chinese Traffic Sign Detection Algorithm Based on Modified YOLOv2, 2017.

[9] Rachel Huang, Jonathan Pedoeem, Cuixian Chen, YOLO-LITE: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers, 2018.

[10] 조원영O, 박승렬, 김현수, 윤태진, Development of AI Systems for Counting Visitors and Check of Wearing Masks Using Deep Learning Algorithms, 2020.

[11] Jimin Yu và Wei Zhang, Face Mask Wearing Detection Algorithm Based on Improved YOLO-v4, 2021.

[12] Naeem Ayoub và Peter Schneider-Kamp, Real-Time On-Board Deep Learning Fault Detection for Autonomous UAV Inspections, 2021.

[13] Han Wu, Chenjie Du, Zhongping Ji, Mingyu Gao và Zhiwei He, SORT-YM: An Algorithm of Multi-Object Tracking with YOLOv4-Tiny and Motion Prediction, 2021.

[14] Sergio Saponara, Abdussalam Elhanashi, Alessio Gagliardi, Implementing a real-time, AI-based, people detection and social distancing measuring system for Covid-19, 2021.

[15] Dujin Wang, Yizhong Wang, Ming Li, Xinting Yang, Jianwei Wu, Wenyong Li, Using an Improved YOLOv4 Deep Learning Network for Accurate Detection of Whitefly and Thrips on Sticky Trap Images, 2021.

[16] Issac Sim, Ju-Hyung Lim, Young-Wan Jang, JiHwan You, SeonTaek Oh, Young-Keun Kim, Developing a Compressed Object Detection Model based on YOLOv4 for Deployment on Embedded GPU Platform of Autonomous System, 2021.

[17] Dong-Jin Shin và Jeong-Joon Kim, A Deep Learning Framework Performance Evaluation to Use YOLO in Nvidia Jetson Platform, 2022.

[18] Haogang Feng, Gaoze Mu, Shida Zhong, Peichang Zhang và Tao Yuan, Benchmark Analysis of YOLO Performance on Edge Intelligence Devices, 2022.

[19] Muhammad Pandu Dwi Cahyo, Fitri Utaminingrum, Autonomous Robot System Based on Room Nameplate Recognition Using YOLOv4 Method on Jetson Nano 2GB, 2022.
[20] Sergio Saponara, Abdussalam Elhanashi, Qinghe Zheng, Developing a real-time social distancing detection system based on YOLOv4-tiny and bird-eye view for COVID-19, 2022.
[21] Sangeeta Yadav, Preeti Gulia, Nasib Singh Gill và Jyotir Moy Chatterjee, A Real-Time Crowd Monitoring and Management System for Social Distance Classification and Healthcare UsingDeep Learning, 2022.
[22] AIZOO, https://github.com/AIZOOTech/FaceMaskDetection, 2020.
[23] Google images, https://images.google.com/, 2022.
[24] Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi, You Only Look Once: Unified, Real-Time Object Detection, 2016.
[25] Alexey Bochkovskiy, Chien-Yao Wang, Hong-Yuan Mark Liao, YOLOv4: Optimal Speed and Accuracy of Object Detection, 2020.
[26] Davis E. Kin, Dlib-ml: A Machine Learning Toolkit, 2009.
[27] Adam Geitgey, https://github.com/ageitgey/face_recognition, 2021.
[28] Vũ Hữu Tiệp, https://machinelearningcoban.com/2016/12/26/introduce/, 2016.
[29] AI For Everyone - AI4E, https://www.facebook.com/nttuan8.AI4E/photos/a.112595893653935/526304232283097/, 2022.
[30] Qingkai Kong, http://qingkaikong.blogspot.com/2016/11/machine-learning-3-artificial-neural.html, 2016.
[31] Alexis Merlaud, Development and use of compact instruments for tropospheric investigations based on optical spectroscopy from mobile platforms, 2013
[32] Wikipedia, https://en.wikipedia.org/wiki/Pixel, 2022.
[33] Shermal Fernando , https://www.opencv-srf.com/2010/09/object-detection-using-color-seperation.html, 2010.
[34] Wikipedia, https://en.wikipedia.org/wiki/HSL_and_HSV, 2022.
[35] NVIDIA GeForce , https://www.youtube.com/watch?v=uJxxCgKa0mU, 2019.
[36] Mayanh24h, https://mayanh24h.com/thuat-ngu-co-ban-trong-nhiep-anh.html, 2019.
[37] Wikipedia , https://en.wikipedia.org/wiki/PAL, 2022.
[38] Saina Ghosh, https://medium.com/clique-org/how-to-create-a-face-recognition-model-using-facenet-keras-fd65c0b092f1, 2020.
[39] Victor Herzog Damke, https://medium.com/cwi-software/reconhecimento-facial-com-raspberry-pi-c4f8579dd640, 2019.
[40] Jiankang Deng, Anastasios Roussos, Grigorios Chrysos, Evangelos Ververas, Irene Kotsia, Jie Shen và Stefanos Zafeiriou, The Menpo Benchmark for Multi-pose 2D and 3D Facial Landmark Localisation and Tracking, 2018.
[41] Kunihiko Fukushima, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position, 1980.
[42] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, 1988.

[43] Afshine Amidi và Shervine Amidi , https://stanford.edu/~shervine/l/vi/teaching/cs-230/cheatsheet-convolutional-neural-networks, 2020.
[44] Adam Geitgey, https://medium.com/@ageitgey/machine-learning-is-fun-part-3-deep-learning-and-convolutional-neural-networks-f40359318721, 2016.
[45] Joseph Redmon, Ali Farhadi, YOLO9000: Better, Faster, Stronger, 2016.
[46] Ayoosh Kathuria, https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b, 2018.
[47] Mingxing Tan và Quoc V. Le, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks, 2019.
[48] Zhanchao Huang và Jianlin Wang, DC-SPP-YOLO: Dense Connection and Spatial Pyramid Pooling Based YOLO for Object Detection, 2019.
[49] Jonathan Hui, https://jonathan-hui.medium.com/yolov4-c9901eaa8e61, 2020.
[50] Duy Hưng, https://www.pandaml.com/computer vision/practical/nhan-dien-khuon-mat-hog-deeplearning/, 2021.
[51] NVIDIA, https://developer.nvidia.com/deepstream-sdk, 2022.
[52] NVIDIA AI IOT, https://github.com/NVIDIA-AI-IOT/deepstream_python_apps, 2021.
[53] Shashank Prasanna, https://on-demand.gputechconf.com/gtcdc/2017/presentation/dc7172-shashank-prasanna-deep-learning-deployment-with-nvidia-tensorrt.pdf, 2017.
[54] NVIDIA, https://developer.nvidia.com/blog/tensorrt-3-faster-tensorflow-inference/, 2017.
[55] NVIDIA, https://developer.nvidia.com/embedded/jetson-nano-developer-kit, 2022.
[56] Logitech, https://www.logitech.com/vi-vn/products/webcams/c922-pro-stream-webcam.html, 2022.
[57] Wiktor Jakub Maj, Matrix operations on License Plate Detector and Recognizer (LPDR), 2020.
[58] OpenCV, https://docs.opencv.org/4.4.0/d3/dc1/tutorial_basic_linear_transform.html, 2022
[59] Wikipedia, https://en.wikipedia.org/wiki/Queue_(abstract_data_type), 2022.
[60] THỊ GIÁC MÁY TÍNH, https://thigiacmaytinh.com/loc-mau-anh-filter-color/, 2017.
[61] István Szentandrási, Adam Herout, Markéta Dubská, Fast Detection and Recognition of QR codes in High-Resolution Images, 2012.
[62] NVIDIA, https://developer.nvidia.com/tensorrt, 2022.
[63] Adam Geitgey, https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78, 2016.
[64] Ayaz Saiyed, https://blog.yudiz.com/ai-ml-based-facial-detection-and-recognition/, 2020.
[65] Martin Anderson, https://www.width.ai/post/facial-detection-and-recognition-with-dlib/, 2021.

Hết./.