

**TRƯỜNG ĐẠI HỌC AN GIANG
KHOA CÔNG NGHỆ THÔNG TIN**

THỰC TẬP CUỐI KHÓA NGÀNH CÔNG NGHỆ THÔNG TIN

XÂY DỰNG ỨNG DỤNG DI ĐỘNG NHẬN DẠNG CÔN TRÙNG

**Đơn vị thực tập
Khoa Công Nghệ Thông Tin**

HUỲNH CƯỜNG

AN GIANG, 04-2024

TRƯỜNG ĐẠI HỌC AN GIANG
KHOA CÔNG NGHỆ THÔNG TIN

THỰC TẬP CUỐI KHÓA NGÀNH CÔNG NGHỆ THÔNG TIN

XÂY DỰNG ỨNG DỤNG DI ĐỘNG NHẬN
DẠNG CÔN TRÙNG

HUỲNH CƯỜNG
DTH215840

GIẢNG VIÊN HƯỚNG DẪN
PGS.TS. ĐOÀN THANH NGHỊ

AN GIANG, 04-2025

NHẬN XÉT CỦA GIẢNG VIÊN HƯỚNG DẪN

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Giảng viên hướng dẫn

(Ký và ghi rõ họ tên)

Nội dung nhận xét:

- **Đồng ý** hay **không đồng ý** cho sinh viên báo cáo TTCK; Nếu không đồng ý cần ghi rõ lý do.
- Kết quả đạt được so với yêu cầu;
- Ý kiến khác (nếu có)

LỊCH LÀM VIỆC

Họ và sinh viên: Huỳnh Cường

Cơ quan thực tập: Văn Phòng Khoa Công Nghệ Thông Tin

Họ và tên giảng viên hướng dẫn: PGS.TS. Đoàn Thanh Nghị

Thời gian thực tập: từ ngày 24 tháng 2 năm 2025 đến ngày 20 tháng 4 Năm 2025

Tuần	Nội dung công việc được giao	Tự nhận xét về mức độ hoàn thành	Nhận xét của giảng viên hướng dẫn	Chữ ký của giảng viên hướng dẫn
01 Từ ngày 24/02 đến ngày 02/03	- Tham quan nơi thực tập và tiến hành nghiên cứu về thuật toán nhận diện YOLO.	Hoàn thành		
02 Từ ngày 03/03 đến ngày 09/03	- Tiếp tục tìm hiểu về YOLO và tiến hành huấn luyện mô hình YOLOv5 cùng các phiên bản Yolo	Hoàn thành		
03 Từ ngày 10/03 đến ngày 16/03	- Phân tích chức năng ứng dụng. - Bắt đầu thiết kế giao diện. - Xây dựng chức năng cơ bản. - Xây dựng cơ sở dữ liệu - Viết báo cáo	Hoàn thành		

04 Từ ngày 17/03 đến ngày 23/03	- Tiếp tục xây dựng chức năng. - Tiếp tục viết báo cáo	Hoàn thành		
05 Từ ngày 24/03 đến ngày 30/03	- Tiếp tục xây dựng chức năng và tiếp tục viết báo cáo.	Hoàn thành		
06 Từ ngày 31/03 đến ngày 06/04	Kiểm tra lỗi, hoàn thiện chương trình hệ thống. Viết báo cáo	Hoàn thành		
07 Từ ngày 07/04 đến ngày 13/04	Tiếp tục Kiểm tra và hoàn thiện chương trình	Hoàn thành		
07 Từ ngày 14/04 đến ngày 20/04	Hoàn thiện ứng dụng			

LỜI CẢM ƠN

Lời đầu tiên, em xin được bày tỏ lòng biết ơn sâu sắc đến Trường Đại học An Giang, đặc biệt là Khoa Công nghệ Thông tin – nơi đã chấp cánh ước mơ, tạo điều kiện và môi trường học tập tuyệt vời để em có thể theo đuổi đam mê với ngành Công nghệ Thông tin trong suốt 4 năm qua. Em vô cùng cảm kích trước sự tận tâm, nhiệt huyết và giàu kinh nghiệm của quý thầy cô – những người không chỉ truyền đạt kiến thức chuyên môn mà còn dạy cho em những bài học quý giá về đạo đức nghề nghiệp và tinh thần cầu tiến.

Em xin chân thành cảm ơn các quý thầy cô tại trường Đại học An Giang. Đặc biệt là thầy Đoàn Thanh Nghị tại khoa Công nghệ thông tin đã luôn tận tình hướng dẫn và chỉ dạy em hoàn thành đồ án thực tập trong suốt 2 tháng thực tập cuối khóa tại khoa Công nghệ thông tin này.

Những chia sẻ và hướng dẫn tận tâm của thầy và khoa Công Nghệ Thông Tin có ý nghĩa hết sức quan trọng và giúp em rất nhiều trong việc hoàn thành đề tài thực tập cuối khóa, bên cạnh đó còn là hành trang tiếp bước cho em trong cả quãng đường dài sau này.

Lời cuối cùng, em xin được gửi lời cảm ơn chân thành nhất đến gia đình, ba mẹ, bạn bè và toàn thể lớp DH22TH3, những người đã luôn sẵn sàng chia sẻ và hỗ trợ nhau cả trong học tập và cuộc sống.

Xin chân thành cảm ơn tất cả mọi người!

Tp Long Xuyên ngày 2 tháng 04 năm 2025

Sinh viên thực hiện

Huỳnh Cường

TÓM TẮT

Việt Nam là một quốc gia nông nghiệp, trong đó nông nghiệp đóng vai trò then chốt đối với nền kinh tế quốc dân. Tuy nhiên, do tình hình biến đổi khí hậu diễn biến ngày càng phức tạp, sự xuất hiện và gia tăng của nhiều loài côn trùng gây hại đã gây ảnh hưởng nghiêm trọng đến sản lượng cây trồng. Trước thực trạng này, việc xác định chính xác các loài côn trùng gây hại trở nên cấp thiết.

Tuy nhiên, trong bối cảnh số lượng loài côn trùng ngày càng đa dạng, phương pháp nhận diện dựa trên kinh nghiệm truyền thống không còn đảm bảo độ chính xác cao. Để góp phần giải quyết vấn đề này, trong bài luận này, tôi đề xuất và phát triển một ứng dụng nhận diện côn trùng trên thiết bị di động, được xây dựng bằng ngôn ngữ lập trình Kotlin và IDE Android Studio.

Ứng dụng này không chỉ hỗ trợ nâng cao độ chính xác trong việc xác định các loài côn trùng gây hại mà còn cung cấp thông tin chi tiết về đặc điểm sinh học, phân bố, hình thái và các biện pháp phòng trừ tương ứng. Qua đó, người dân có thể lựa chọn các giải pháp sinh học phù hợp, giúp giảm thiểu việc lạm dụng thuốc bảo vệ thực vật, hướng tới nền nông nghiệp bền vững và an toàn.

Đề tài gồm ba chương với các nội dung cụ thể như sau:

Chương 1: Tổng quan về đề tài. Chương này trình bày tổng quan về đề tài nghiên cứu, bao gồm việc làm rõ tầm quan trọng và ý nghĩa thực tiễn của đề tài trong bối cảnh hiện nay. Bên cạnh đó, chương cũng nêu rõ mục tiêu nghiên cứu, phạm vi thực hiện, cũng như phương pháp nghiên cứu được áp dụng trong suốt quá trình xây dựng và phát triển hệ thống.

Chương 2: Cơ sở lý thuyết và các nghiên cứu liên quan. Chương này cung cấp nền tảng lý thuyết phục vụ cho việc giải quyết vấn đề nghiên cứu. Cụ thể, chương trình bày các kiến thức về kỹ thuật phát hiện đối tượng, giới thiệu thuật toán nhận diện YOLO, NMS, IoU, phương pháp học chuyển giao (Transfer Learning), cùng với các công nghệ và nền tảng hỗ trợ như Android Studio, cơ sở dữ liệu PostgreSQL và thư viện TensorFlow Lite.

Chương 3: Xây dựng hệ thống nhận diện côn trùng trên thiết bị di động. Chương này tập trung mô tả quá trình xây dựng hệ thống nhận diện côn trùng trên nền tảng Android. Nội dung bao gồm thiết kế kiến trúc hệ thống nhận diện, cùng các bước triển khai cụ thể. Phần cuối chương là phần đánh giá, kết luận và đề xuất các hướng phát triển tiếp theo của hệ thống nhằm nâng cao hiệu quả và tính ứng dụng trong thực tiễn.

CHƯƠNG 1	1
TỔNG QUAN	1
1.1. GIỚI THIỆU CƠ QUAN THỰC TẬP.....	1
1.2. TÍNH CẤP THIẾT CỦA ĐỀ TÀI.....	1
1.2.1 Nhu cầu cấp bách.....	1
1.2.2 Tiềm năng ứng dụng.....	2
1.2.3. Mục tiêu nghiên cứu	Error! Bookmark not defined.
1.2.4 Phạm vi nghiên cứu	2
1.2.5 Phương pháp nghiên cứu	3
1.3. LÝ DO CHỌN ĐỀ TÀI	3
1.4. MỤC TIÊU NGHIÊN CỨU	4
1.5. PHẠM VI ĐỀ TÀI.....	4
CHƯƠNG 2	5
CƠ SỞ LÝ THUYẾT	5
2.1. PHÁT HIỆN ĐỐI TƯỢNG.....	5
2.2. GIỚI THIỆU VỀ YOLO (YOU ONLY LOOK ONCE)	6
2.2.1 Mạng YOLO là gì?	6
2.2.2 Kiến trúc của YOLO.....	6
2.2.3 Sự phát triển của các phiên bản từ khi ra mắt cũng như mới nhất và dễ tiếp cận nhất của YOLO	7
2.3 PHƯƠNG PHÁP TRANSFER LEARNING	14
2.3.1 Giới thiệu về Transfer Learning	14
2.3.2 Transfer Learning	16
2.3.3 Những điều cần lưu ý khi áp dụng transfer learning.....	18
2.3.4 Cách train YOLO	19
2.4 SƠ LƯỢC VỀ ANDROID.....	20
2.4.1 Giới thiệu	20
2.4.2 Kiến trúc của ứng dụng Android	22
2.4.3 Môi trường phát triển của ứng dụng	24
2.4.4 Ngôn ngữ lập trình Kotlin	25
2.4.5 Giới thiệu Supabase.....	26
2.4.6 Cơ sở dữ liệu PosgreSQL.....	27
2.5 TÌM HIỂU VỀ CÔN TRÙNG.....	30
2.5.1 Giới thiệu về côn trùng	30
2.5.2 Hình thái phát triển.....	30
2.5.3 Vòng đời của côn trùng.....	32
2.5.4 Tập tính của côn trùng	33
2.5.5 Giác quan của côn trùng	34
2.5.6 Phân loại côn trùng	35
2.5.7 Côn trùng có lợi và côn trùng gây hại.....	36

2.5 NHỮNG NGHIÊN CỨU TƯƠNG TỰ	38
2.5.1 Trong nước.....	38
2.5.2 Ngoài nước.....	39
CHƯƠNG 3	44
HỆ THỐNG NHẬN DẠNG CÔN TRÙNG TRÊN THIẾT BỊ DI ĐỘNG	44
3.1 TỔNG QUAN HỆ THỐNG.....	44
3.2 MÔ TẢ HỆ THỐNG	46
3.2.1 Danh sách Actor	46
3.2.2 Danh sách User case	46
3.2.3 Sơ đồ use case	47
3.2.4 Đặc tả chi tiết use case	47
3.3 THIẾT KẾ CƠ SỞ DỮ LIỆU	53
3.3.1 Thiết kế chi tiết các bảng dữ liệu.....	53
3.3.2 Các lớp đối tượng	55
3.4 THỰC NGHIỆM VÀ BÀN LUẬN.....	55
3.4.1 Mô tả tập dữ liệu của 10 lớp.....	55
3.4.2 Độ chính xác khi nhận diện.....	58
3.4.3 Chế độ hoạt động của ứng dụng	59
3.4.4 Độ chính xác khi xác định côn trùng trên tập dữ liệu hình ảnh.....	60
3.5 GIAO DIỆN CHƯƠNG TRÌNH	63
3.5.1 Giao diện màn hình đăng nhập	63
3.5.2 Giao diện màn hình đăng ký	64
3.5.3 Giao diện home	65
3.5.4 Giao diện chatbot.....	66
3.5.5 Giao diện detect côn trùng từ camera	67
3.5.6 Giao diện detect côn trùng từ ảnh trong thư viện	68
3.5.7 Giao diện xem thông tin chi tiết côn trùng.....	69
3.5.8 Giao diện Admin đăng bài viết.....	70
3.6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN.....	71
3.6.1 Kết luận.....	71
3.6.2 Hướng phát triển.....	72
HƯỚNG DẪN SỬ DỤNG ỨNG DỤNG.....	73
TÀI LIỆU THAM KHẢO.....	74

DANH SÁCH HÌNH VẼ

Hình 1: Kiến trúc của YOLO.....	7
Hình 2: YOLOv1 chia bức ảnh thành SxS ô lưới	8
Hình 3: Kiến trúc của YOLOv2 (<i>Figure 6. The Architecture of YOLOv2., n.d.</i>).....	9
Hình 4: Sơ đồ mạng Darknet 53.....	10

Hình 5: Biểu đồ hiệu suất của Yolov9.....	13
Hình 6: Sơ đồ so sánh hiệu suất của mô hình trước và sau khi áp dụng Transfer Learning (Khánh, n.d.)	15
Hình 7: Các đặc trưng học được từ mạng CNN.....	16
Hình 8: Kiến trúc của mạng VGG16 được sử dụng làm base network trong transfer learning.	17
Hình 9: Kiến trúc based network kết hợp với fully connected layers...	17
Hình 10: Chiến lược áp dụng transfer learning.....	18
Hình 11: Chọn runtime	19
Hình 12: Cấu trúc thư mục.....	20
Hình 13: Kiến trúc của Android.....	23
Hình 14: Hình ảnh về đa dạng côn trùng.....	30
Hình 15: Vòng đời biến thái không hoàn toàn của Châu Châu	32
Hình 16: Vòng đời biến thái hoàn toàn của Bướm	33
Hình 17: Hệ thống của ứng dụng nhận diện dịch hại của Sri Lanka	39
Hình 18: Xác định đúng vị trí của đối tượng.	41
Hình 19: Quy trình công việc cơ bản của Faster R-CNN để phát hiện và phân loại lớp đối tượng mục tiêu, tức là các loài gây hại cây trồng (trong nghiên cứu của tác giả).	42
Hình 20: Sơ đồ tổng quát hệ thống.....	44
Hình 21: Sơ đồ tổng quát quá trình huấn luyện mô hình và nơi lưu trữ mô hình	45
Hình 22: Sơ đồ usecase.....	47
Hình 23: Hình ảnh lược đồ quan hệ cơ sở dữ liệu	55
Hình 24: Độ chính xác khi nhận diện.....	58
Hình 25: Miêu tả area of overlap	59
Hình 26: Giao diện đăng nhập	63
Hình 27: Giao diện đăng ký	64
Hình 28: Giao diện home.....	65
Hình 29: Giao diện chatbot.....	66

Hình 30: Giao diện nhận dạng camera	67
Hình 31: Giao diện nhận dạng bằng ảnh	68
Hình 32: Giao diện xem chi tiết côn trùng	69
Hình 33: Giao diện tạo bài viết admin.....	70
Hình 34: Giao diện sửa thông tin người dùng	71

DANH SÁCH SÁCH CÁC BẢNG

Bảng 1: Các phiên bản Android	21
Bảng 2: So sánh Kotlin và Java.....	26
Bảng 3: So sánh PostgreSQL, Firebase và SQLite	28
Bảng 4: Hiệu suất của phương pháp Faster R-CNN với các phương pháp khác trong tài liệu.....	42
Bảng 5: Danh sách Actor của hệ thống	46
Bảng 6: Danh sách các use case của hệ thống	46
Bảng 7: Đặc tả use case “Đăng Nhập”	47
Bảng 8: Đặc tả use case “Đăng Ký”	48
Bảng 9: Đặc tả use case “User Login”	49
Bảng 10: Đặc tả use case “Nhận Diện”	50
Bảng 11: Đặc tả use case “Nhận Diện Loài Mới”	50
Bảng 12: Đặc tả use case “Xem danh sách hình ảnh theo tên côn trùng”	51
Bảng 13: Đặc tả use case đăng bài viết côn trùng.....	52
Bảng 14: Bảng thông tin côn trùng	53
Bảng 15: Bảng người dùng	53
Bảng 16: Bảng bài viết	53
Bảng 17: Bảng bài viết yêu thích	54
Bảng 18: Côn trùng trong tập dữ liệu 10 lớp	55
Bảng 19: Bảng độ chính xác sau khi nhận diện của tập dữ liệu 10 lớp	60
Bảng 20: So sánh các phiên bản Yolov9	61

DANH SÁCH CÁC TỪ VIẾT TẮT

STT	Từ viết tắt	Từ viết đủ nghĩa
1	ĐBSCL	Đồng Bằng Sông Cửu Long
2	CNN	Convolutional Neural Network
3	GPU	Graphics Processing Unit
4	ReLU	Rectified Linear Unit
5	R-CNN	Regions with CNN feature
6	CPU	Central Processing Unit
7	SSD	Single Shot Detector
8	YOLO	You Only Look Once
9	IoU	Intersection Over Union
10	FPS	Frames Per Second
11	GT	Ground Truth
12	AI	Artificial Intelligence
13	TPUs	Tensor Processing Units
14	ResNet	Residual Network
15	NMS	Non-Maximum Suppression
16	IDE	Integrated development environment
17	IoU	Intersection over Union

CHƯƠNG 1

TỔNG QUAN

1.1. GIỚI THIỆU CƠ QUAN THỰC TẬP

Khoa Công nghệ Thông tin (CNTT) – Trường Đại học An Giang được thành lập ngày 14/08/2017 theo Quyết định số 1461/QĐ-ĐHAG. Trước đó, lĩnh vực CNTT được đào tạo và nghiên cứu dưới hình thức bộ môn trực thuộc các khoa khác. Việc thành lập Khoa CNTT nhằm đáp ứng nhu cầu đào tạo nguồn nhân lực chất lượng cao trong bối cảnh chuyển đổi số và cách mạng công nghiệp 4.0. Khoa hiện có hai bộ môn chính: Công nghệ phần mềm và Hệ thống thông tin, với đội ngũ giảng viên trình độ sau đại học, nhiều người đang làm nghiên cứu chuyên sâu. Khoa cung cấp chương trình đào tạo cử nhân CNTT đạt chuẩn quốc gia, chú trọng năng lực thực hành và kỹ năng mềm (*Giới Thiệu - Khoa Công Nghệ Thông Tin - ĐHAG*, n.d.).

1.2. TÍNH CẤP THIẾT CỦA ĐỀ TÀI

1.2.1 Nhu cầu cấp bách

Trong bối cảnh hiện nay, nền nông nghiệp Việt Nam đang phải đối mặt với nhiều thách thức, đặc biệt là vấn đề côn trùng gây hại làm ảnh hưởng nghiêm trọng đến năng suất và chất lượng cây trồng. Để đối phó với tình trạng này, người nông dân chủ yếu sử dụng thuốc trừ sâu hóa học. Tuy nhiên, việc sử dụng bừa bãi và thiếu kiểm soát các loại thuốc này không chỉ gây ô nhiễm môi trường mà còn đe dọa trực tiếp đến sức khỏe con người và hệ sinh thái nông nghiệp bền vững.

Bên cạnh đó, sự thiếu hụt các công cụ hỗ trợ nhận diện và phân loại côn trùng gây hại một cách chính xác khiến việc kiểm soát và phòng trừ trở nên kém hiệu quả. Điều này đòi hỏi phải có các giải pháp thay thế mang tính ứng dụng cao, thân thiện với môi trường và dễ tiếp cận đối với người dùng, đặc biệt là trong khu vực nông thôn.

Trên cơ sở đó, đề tài **“Xây dựng ứng dụng di động nhận dạng côn trùng”** mang ý nghĩa thực tiễn rõ rệt. Việc phát triển một công cụ hỗ trợ nhận dạng côn trùng nhanh chóng, chính xác, hoạt động được trên điện thoại thông minh sẽ góp phần nâng cao hiệu quả trong công tác quản lý dịch hại, giảm thiểu sự phụ thuộc vào thuốc trừ sâu, hướng đến một nền nông nghiệp hiện đại, an toàn và bền vững.

1.2.2 Tiềm năng ứng dụng

Bên cạnh tính cấp thiết về mặt thực tiễn, đề tài còn mang lại giá trị ứng dụng cao nhờ tích hợp công nghệ trí tuệ nhân tạo, đặc biệt là thuật toán nhận dạng đối tượng YOLO (You Only Look Once). Ứng dụng này có thể hỗ trợ người nông dân trong nhiều khía cạnh quan trọng:

Phát hiện sớm và chính xác các loại côn trùng gây hại, giúp chủ động phòng trừ ngay từ giai đoạn đầu.

Cung cấp thông tin chi tiết về đặc điểm sinh học, tập tính sinh trưởng, cũng như các biện pháp phòng trừ phù hợp đối với từng loại côn trùng.

Hỗ trợ người dân sử dụng thuốc trừ sâu một cách hợp lý và hiệu quả, nhờ nhận diện chính xác loài gây hại, từ đó giảm thiểu lãng phí, hạn chế tác động tiêu cực đến môi trường, sức khỏe con người và hệ sinh thái.

Với việc tích hợp trí tuệ nhân tạo và khả năng triển khai trên các thiết bị di động phổ biến, ứng dụng không chỉ nâng cao hiệu quả sản xuất nông nghiệp mà còn góp phần thúc đẩy quá trình chuyển đổi số trong ngành nông nghiệp Việt Nam.

1.2.3 Phạm vi nghiên cứu

Về mặt lý thuyết:

Nghiên cứu thuật toán nhận dạng đối tượng YOLO (You Only Look Once) cùng các phiên bản cải tiến (YOLOv8, YOLOv9, YOLOv10, v.v.), nhằm lựa chọn phiên bản phù hợp nhất cho bài toán nhận dạng côn trùng.

Tìm hiểu các kỹ thuật xây dựng ứng dụng di động trên nền tảng Android, bao gồm giao diện người dùng, tích hợp mô hình học máy và tối ưu hóa hiệu suất hoạt động của ứng dụng.

Khảo sát các phương pháp xử lý ảnh và kỹ thuật học máy, đặc biệt là học sâu (deep learning), phục vụ cho quá trình huấn luyện và triển khai mô hình nhận dạng.

Về mặt ứng dụng:

Phát triển một ứng dụng di động có khả năng nhận dạng hình ảnh các loại côn trùng gây hại phổ biến trên cây trồng trong điều kiện thực tế.

Xây dựng và chuẩn hóa cơ sở dữ liệu hình ảnh côn trùng với đa dạng chủng loại, góc chụp và điều kiện môi trường nhằm đảm bảo độ chính xác của mô hình học máy.

Tích hợp các chức năng cung cấp thông tin sinh học của từng loại côn trùng và gợi ý biện pháp phòng trừ phù hợp, hỗ trợ người nông dân trong việc giám sát và quản lý côn trùng gây hại một cách hiệu quả.

1.2.4 Phương pháp nghiên cứu

Nghiên cứu cơ sở lý thuyết: Tham khảo các tài liệu khoa học, bài báo chuyên ngành và sách chuyên khảo liên quan đến lĩnh vực nhận dạng hình ảnh, thuật toán YOLO và các phiên bản cải tiến.

Tổng hợp và phân tích thông tin:

+ Thu thập thông tin từ các nguồn tài liệu đáng tin cậy về các loại côn trùng gây hại phổ biến trong nông nghiệp, đặc điểm nhận dạng, vòng đời và các biện pháp phòng trừ hiệu quả.

+ Tổ chức và phân loại thông tin để phục vụ cho việc xây dựng cơ sở dữ liệu và nội dung trong ứng dụng.

Huấn luyện mô hình nhận dạng:

+ Lựa chọn phiên bản YOLO phù hợp và tiến hành huấn luyện mô hình nhận dạng côn trùng trên cơ sở dữ liệu đã xây dựng.

+ Đánh giá và điều chỉnh mô hình nhằm đạt được hiệu quả cao nhất về tốc độ xử lý và độ chính xác.

Phát triển ứng dụng di động:

+ Thiết kế và lập trình ứng dụng di động trên nền tảng Android, tích hợp mô hình YOLO đã huấn luyện để thực hiện chức năng nhận dạng.

+ Bổ sung các tính năng hỗ trợ như hiển thị thông tin chi tiết về côn trùng, khuyến nghị biện pháp phòng trừ và lưu trữ lịch sử nhận dạng.

1.3. LÝ DO CHỌN ĐỀ TÀI

Trong lĩnh vực nông nghiệp, sâu bệnh hại là một trong những nguyên nhân chính gây ra tổn thất nghiêm trọng về năng suất và chất lượng cây trồng. Theo Tổ chức Lương thực và Nông nghiệp Liên Hợp Quốc (FAO), mỗi năm có từ 20% đến 40% sản lượng cây trồng toàn cầu bị phá hoại bởi các loài côn trùng gây hại, dẫn đến thiệt hại kinh tế ước tính lên tới khoảng 220 tỷ USD trên toàn thế giới. Để đối phó với tình trạng này, người nông dân thường xuyên sử dụng các loại thuốc trừ sâu nhằm kiểm soát và phòng ngừa sự phát triển của sâu bệnh.

Tuy nhiên, việc lạm dụng thuốc bảo vệ thực vật trong thời gian dài không những gây ô nhiễm môi trường mà còn làm gia tăng nguy cơ xuất hiện các vấn đề nghiêm trọng về sức khỏe cộng đồng, như ung thư, bệnh hô hấp và các rối

loạn di truyền ở người tiêu dùng. Do đó, việc phát triển các giải pháp kỹ thuật tiên tiến để nhận dạng sớm và chính xác các loại sâu bệnh hại là nhu cầu cấp thiết trong bối cảnh hiện nay.

Trên cơ sở đó, đề tài “Xây dựng ứng dụng nhận dạng côn trùng cho các thiết bị di động” được lựa chọn nhằm hướng tới mục tiêu hỗ trợ người nông dân trong việc phát hiện sớm các loại côn trùng gây hại, giảm thiểu việc sử dụng thuốc trừ sâu không cần thiết, góp phần nâng cao năng suất cây trồng và bảo vệ sức khỏe cộng đồng. Bằng cách ứng dụng các thành tựu mới trong lĩnh vực trí tuệ nhân tạo, đặc biệt là công nghệ nhận dạng hình ảnh với thuật toán YOLO, đề tài hứa hẹn mang lại giải pháp hiệu quả, tiện dụng và dễ tiếp cận cho người dùng trong lĩnh vực nông nghiệp thông minh.

1.4. MỤC TIÊU NGHIÊN CỨU

- Nghiên cứu các phương pháp và quy trình xây dựng ứng dụng di động, đặc biệt trên nền tảng Android, từ thiết kế giao diện đến tích hợp các chức năng nhận diện thông minh.

- Nghiên cứu tổng quan về thuật toán YOLO (You Only Look Once) và khả năng ứng dụng của thuật toán này trong việc nhận dạng đối tượng từ hình ảnh, đặc biệt là côn trùng.

- Phân tích và thiết kế kiến trúc phần mềm của ứng dụng, bao gồm các thành phần chức năng, quy trình xử lý dữ liệu hình ảnh, tích hợp mô hình học sâu và các tính năng hỗ trợ người dùng.

- Xây dựng và hiện thực hóa một ứng dụng di động có khả năng nhận dạng nhanh chóng và chính xác các loại côn trùng gây hại, đồng thời cung cấp thông tin mô tả chi tiết và các khuyến nghị phòng trừ phù hợp dành cho người nông dân.

1.5. PHẠM VI ĐỀ TÀI

- Về lý thuyết:

- + Tìm hiểu thuật toán nhận diện đối tượng YOLO (You Only Look Once), bao gồm nguyên lý hoạt động, các phiên bản cải tiến và khả năng ứng dụng trong lĩnh vực thị giác máy tính.

- + Nghiên cứu nền tảng phát triển ứng dụng di động trên hệ điều hành Android, bao gồm thiết kế giao diện, quản lý bộ nhớ và tích hợp mô hình học máy.

+ Tìm hiểu kỹ thuật Transfer Learning nhằm tận dụng các mô hình đã được huấn luyện sẵn, giúp giảm thời gian đào tạo và nâng cao hiệu quả nhận dạng côn trùng.

Về ứng dụng:

+ Xây dựng một ứng dụng di động có khả năng nhận dạng các loại côn trùng gây hại phổ biến trên cây trồng thông qua hình ảnh được chụp từ thiết bị di động.

+ Tích hợp vào ứng dụng các tính năng cung cấp thông tin sinh học của từng loài côn trùng như đặc điểm nhận dạng, tập tính, môi trường phân bố và các biện pháp phòng trừ phù hợp.

Đối tượng nghiên cứu: Các loài côn trùng gây hại đến cây trồng trong nông nghiệp, được phân loại theo hình dáng, đặc điểm sinh học và mức độ phổ biến. Dữ liệu hình ảnh thu thập sẽ bao gồm nhiều góc độ và điều kiện ánh sáng khác nhau để đảm bảo tính đa dạng và độ chính xác cho mô hình nhận dạng.

CHƯƠNG 2

CƠ SỞ LÝ THUYẾT

2.1. PHÁT HIỆN ĐỐI TƯỢNG

Nhận dạng đối tượng (Object Recognition):

là một thuật ngữ tổng quát dùng để chỉ một tập hợp các nhiệm vụ trong lĩnh vực thị giác máy tính, liên quan đến việc xác định và phân tích các đối tượng xuất hiện trong hình ảnh kỹ thuật số. Để hiểu rõ hơn về nhiệm vụ Phát hiện Đối tượng (Object Detection), cần phân biệt rõ ràng với hai nhiệm vụ nền tảng khác là Phân loại Hình ảnh (Image Classification) và Định vị Vật thể (Object Localization) (Liu et al., 2016).

Để hiểu rõ hơn về Phát hiện Đối tượng (Object Detection), ta cần phân biệt nó với hai nhiệm vụ khác trong lĩnh vực thị giác máy tính: Phân loại Hình ảnh (Image Classification) và Định vị Vật thể (Object Localization).

Phân loại Hình ảnh (Image Classification):

Đây là nhiệm vụ cơ bản nhất trong thị giác máy tính, với mục tiêu gán một hoặc nhiều nhãn lớp cho toàn bộ hình ảnh. Đầu vào là một hình ảnh có thể chứa một hoặc nhiều đối tượng, và đầu ra là các nhãn phân loại tương ứng thể hiện các loại đối tượng có mặt trong ảnh. Ví dụ, hệ thống có thể phân loại hình ảnh là "mèo", "chó" hoặc "chim" dựa trên nội dung chính của ảnh (Krizhevsky et al., 2017).

Định vị Vật thể (Object Localization):

Nhiệm vụ này không chỉ xác định loại đối tượng có trong ảnh mà còn xác định chính xác vị trí của chúng thông qua việc vẽ các hộp giới hạn (bounding boxes). Mỗi bounding box được định nghĩa bởi tọa độ trung tâm, chiều cao và chiều rộng, nhằm xác định rõ khu vực mà đối tượng chiếm dụng trong ảnh. Ví dụ, hệ thống có thể vẽ khung quanh một chiếc xe hoặc một người đang đứng trong ảnh (Girshick, 2015).

Phát hiện Đối tượng (Object Detection):

Đây là một nhiệm vụ nâng cao, kết hợp cả hai nhiệm vụ trên: vừa phân loại, vừa định vị đối tượng. Đầu vào là một hình ảnh chứa nhiều đối tượng, và đầu ra là tập hợp các hộp giới hạn đi kèm với nhãn phân loại tương ứng cho từng đối tượng. Ví dụ, hệ thống không chỉ phát hiện người trong ảnh mà còn phân loại họ thành nam hoặc nữ (Redmon et al., 2016a).

2.2. GIỚI THIỆU VỀ YOLO (YOU ONLY LOOK ONCE)

2.2.1 Mạng YOLO là gì?

YOLO (You Only Look Once) là một trong những mô hình phát hiện đối tượng (object detection) tiên tiến nhất hiện nay, nổi bật nhờ khả năng phát hiện nhanh và chính xác các đối tượng trong hình ảnh chỉ thông qua một lần xử lý duy nhất. Khác với các phương pháp truyền thống như R-CNN hay Fast R-CNN vốn cần nhiều bước xử lý (region proposal, feature extraction, classification...), YOLO tiếp cận bài toán theo hướng hoàn toàn mới: nó coi việc phát hiện đối tượng là một bài toán hồi quy duy nhất, trực tiếp từ pixel ảnh đến bounding box và nhãn lớp. Nhờ đó, YOLO có thể xử lý hình ảnh theo thời gian thực với tốc độ vượt trội mà vẫn duy trì độ chính xác cao (Redmon et al., 2016a).

2.2.2 Kiến trúc của YOLO

Kiến trúc của YOLO có thể được chia thành ba thành phần chính:

Backbone (Mạng trích xuất đặc trưng):

Đây là thành phần chịu trách nhiệm trích xuất đặc trưng từ ảnh đầu vào. Trong YOLO phiên bản đầu tiên, backbone là một mạng CNN được thiết kế riêng gồm 24 lớp tích chập (convolutional layers) và 2 lớp kết nối đầy đủ (fully connected) (Redmon et al., 2016b). Trong các phiên bản sau như YOLOv3 và YOLOv4, các backbone mạnh hơn như **Darknet-53** hoặc **CSPDarknet** đã được sử dụng nhằm tăng độ chính xác và khả năng trích xuất đặc trưng (Bochkovskiy et al., 2020a).

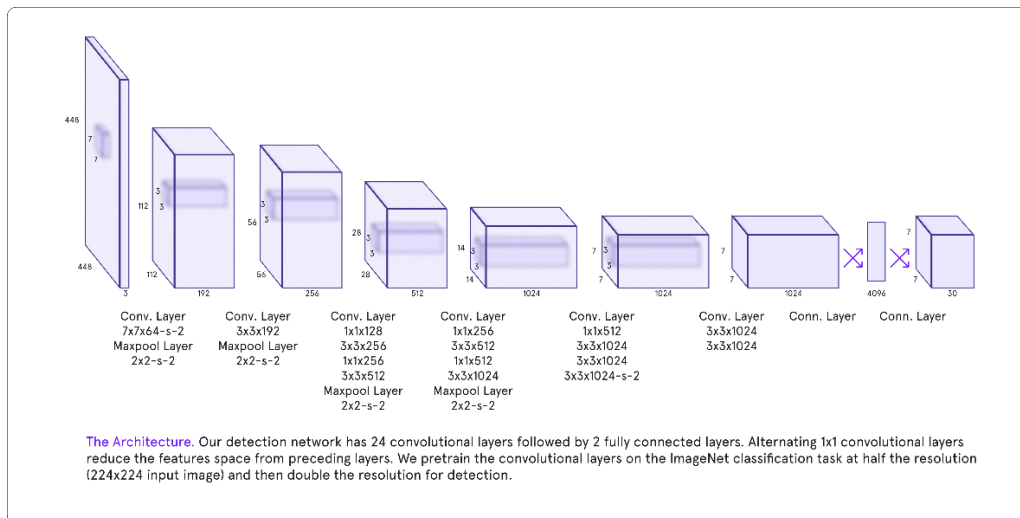
Neck (Tăng cường đặc trưng):

Neck là phần trung gian giữa backbone và head, giúp kết hợp thông tin ở nhiều

cấp độ khác nhau bằng các kỹ thuật như **FPN (Feature Pyramid Network)** và **PANet (Path Aggregation Network)**, từ đó nâng cao khả năng phát hiện các đối tượng ở nhiều kích thước (Bochkovski et al., 2020a).

Head (Đầu ra dự đoán):

Đây là phần cuối của mạng, thực hiện nhiệm vụ dự đoán bounding box, xác suất của từng lớp và độ tin cậy (objectness score). YOLO chia hình ảnh đầu vào thành một lưới (grid), mỗi ô lưới chịu trách nhiệm phát hiện các đối tượng nằm trong phạm vi của nó. Mỗi ô sẽ đưa ra các dự đoán bao gồm tọa độ bounding box (x, y, w, h), xác suất có đối tượng, và phân phối xác suất theo các lớp (Redmon et al., 2016a).



Hình 1: Kiến trúc của YOLO

2.2.3 Sự phát triển của các phiên bản từ khi ra mắt cũng như mới nhất và dễ tiếp cận nhất của YOLO

2.2.3.1 YOLOv1: Thuật toán phát hiện đối tượng thời gian thực đầu tiên.

YOLOv1 (You Only Look Once version 1) là phiên bản đầu tiên trong chuỗi mô hình phát hiện đối tượng YOLO, được đề xuất bởi Joseph Redmon và các cộng sự vào năm 2016. Đây là một trong những bước đột phá lớn trong lĩnh vực thị giác máy tính khi lần đầu tiên kết hợp hai tác vụ phân loại và định vị đối tượng thành một quá trình duy nhất, cho phép phát hiện đối tượng trong thời gian thực với độ chính xác đáng kể (Redmon et al., 2016a).

Không giống như các phương pháp phát hiện đối tượng trước đó như R-CNN hay Fast R-CNN vốn phụ thuộc vào các bước trích xuất đặc trưng riêng biệt và sử dụng bộ đề xuất vùng (region proposals), YOLOv1 xử lý toàn bộ hình

ảnh đầu vào thông qua một mạng nơ-ron duy nhất. Cụ thể, hình ảnh được chia thành một lưới $S \times S$ (thường là 7×7), mỗi ô lưới chịu trách nhiệm dự đoán các bounding box và xác suất thuộc về từng lớp đối tượng nếu có đối tượng xuất hiện trong ô đó. Mỗi dự đoán bao gồm tọa độ của bounding box, độ tin cậy (confidence score) và xác suất phân loại.

Mặc dù YOLOv1 chưa đạt độ chính xác cao như các phương pháp hai giai đoạn (two-stage detectors), nhưng mô hình lại nổi bật ở khả năng xử lý nhanh, phù hợp cho các ứng dụng yêu cầu tốc độ thời gian thực như giám sát video, xe tự hành và robot. Việc coi phát hiện đối tượng là một bài toán hồi quy tổng quát cũng giúp đơn giản hóa đáng kể kiến trúc mô hình, từ đó giảm độ phức tạp và dễ dàng triển khai trong thực tế.

YOLOv1 đã đặt nền móng cho các phiên bản cải tiến sau này như YOLOv2, YOLOv3, YOLOv4 và các nhánh hiện đại như YOLOv5–YOLOv9, đồng thời góp phần thay đổi tư duy thiết kế các hệ thống phát hiện đối tượng hiện đại.



$S \times S$ grid on input

Hình 2: YOLOv1 chia bức ảnh thành $S \times S$ ô lưới

2.2.3.2 YOLOv2

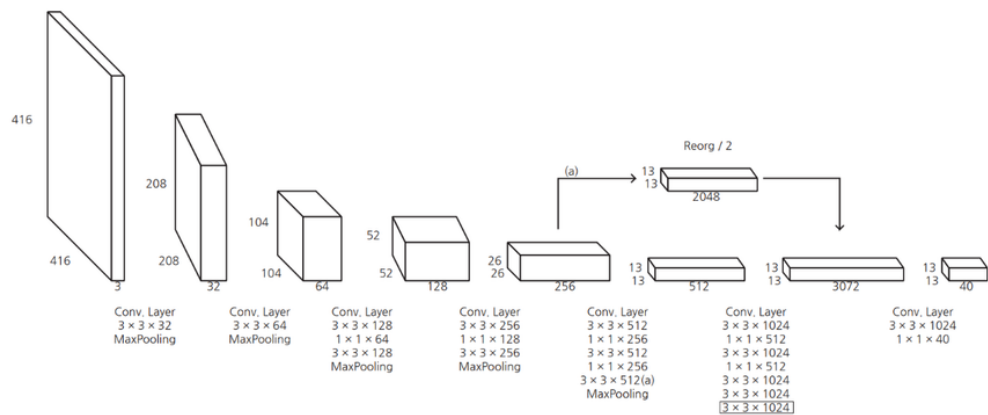
YOLOv2 đánh dấu một bước tiến vượt bậc so với mô hình YOLO ban đầu nhờ những cải tiến quan trọng sau (Redmon & Farhadi, 2016).

Tăng Cường Độ Ổn Định và Hiệu Năng: YOLOv2 áp dụng chuẩn hóa theo lô (batch normalization) cho tất cả các lớp tích chập. Kỹ thuật này giúp

giảm thiểu hiện tượng quá khớp (overfitting) - vấn đề thường gặp khi mô hình học quá tốt dữ liệu huấn luyện và gặp khó khăn với những ví dụ mới. Nhờ giảm thiểu hiện tượng quá khớp, chuẩn hóa theo lô giúp mô hình hoạt động ổn định và hiệu quả hơn (Redmon & Farhadi, 2016).

Tập Trung Tốt Hơn Vào Các Đối Tượng Nhỏ: Khả năng xử lý hình ảnh độ phân giải cao của YOLOv2 là một lợi thế lớn. Nhờ vậy, mô hình có thể xử lý các chi tiết nhỏ hơn, dẫn đến việc phát hiện các đối tượng nhỏ trong ảnh tốt hơn (Redmon & Farhadi, 2016). Trước đây, những đối tượng nhỏ này có thể hoàn toàn bị bỏ sót.

Khai Thác Sức Mạnh: Vai Trò Của Anchor Box: YOLOv2 tiếp thu khái niệm anchor box từ Faster R-CNN. Các hộp được xác định trước này có kích thước và tỷ lệ khung hình khác nhau, đóng vai trò như điểm tham chiếu cho mô hình trong quá trình phát hiện đối tượng. Bằng cách tận dụng anchor box, YOLOv2 đạt được độ chính xác cao hơn trong việc dự đoán kích thước và hình dạng của các đối tượng trong ảnh (Redmon & Farhadi, 2016).

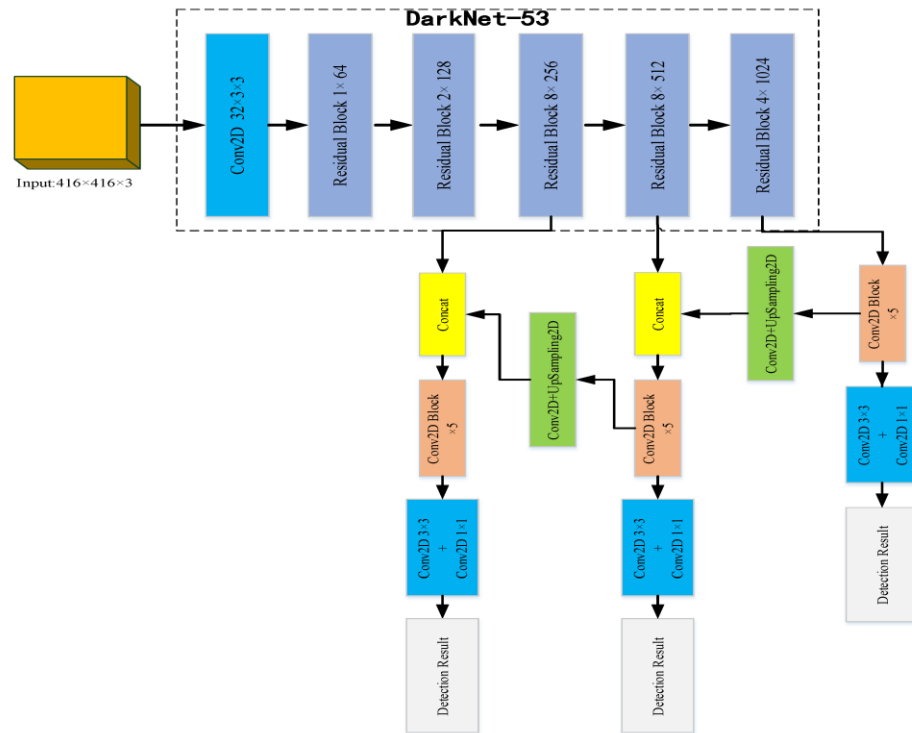


Hình 3: Kiến trúc của YOLOv2 (Figure 6. The Architecture of YOLOv2., n.d.)

2.2.3.3 YOLOv3

YOLOv3 (You Only Look Once version 3) là một thuật toán phát hiện đối tượng mạnh mẽ và hiệu quả cao, được đề xuất bởi Joseph Redmon và cộng sự vào năm 2018 (Redmon & Farhadi, 2018). Mô hình này cải tiến từ các phiên bản trước bằng cách sử dụng kiến trúc mạng sâu Darknet-53 – một mạng tích chập gồm 53 lớp, kết hợp với các kết nối tắt (residual connections) giúp cải thiện hiệu suất huấn luyện. Một trong những điểm nổi bật của YOLOv3 là khả năng phát hiện đối tượng ở ba thang đo khác nhau (multi-scale prediction), giúp tăng cường khả năng nhận diện các vật thể có kích thước nhỏ, trung bình và lớn trong cùng một ảnh. Mô hình sử dụng anchor boxes để tăng độ chính xác khi dự đoán bounding boxes, đồng thời áp dụng hàm kích hoạt sigmoid để tính toán

xác suất tồn tại đối tượng trong mỗi ô lưới (grid cell). YOLOv3 mang lại tốc độ xử lý nhanh, phù hợp với các ứng dụng thời gian thực như giám sát an ninh, xe tự hành, và phân tích video. Tuy nhiên, trong một số trường hợp phức tạp, độ chính xác của YOLOv3 vẫn có thể thấp hơn so với các phương pháp như Faster R-CNN, đặc biệt khi gặp các đối tượng chồng lấn hoặc trong điều kiện ánh sáng kém (Redmon & Farhadi, 2018).



Hình 4: Sơ đồ mạng Darknet 53

2.2.3.4 YOLO4

YOLOv4 là một phiên bản nâng cấp của họ mạng YOLO, được giới thiệu bởi Bochkovskiy và cộng sự vào năm 2020, với mục tiêu cải thiện hiệu suất phát hiện đối tượng trong khi vẫn duy trì tốc độ xử lý cao (Bochkovskiy et al., 2020b). Mô hình này sử dụng CSPDarknet53 làm kiến trúc trích xuất đặc trưng, kết hợp các kỹ thuật như Cross-Stage Partial Connections (CSP) và Spatial Pyramid Pooling (SPP) nhằm tăng cường khả năng học đặc trưng mà không làm tăng đáng kể số lượng tham số. Ngoài ra, YOLOv4 tích hợp nhiều cải tiến từ các kỹ thuật hiện đại như Bag of Freebies và Bag of Specials, bao gồm Mosaic Data Augmentation, DropBlock, Mish activation và CIoU loss. Những kỹ thuật này giúp cải thiện khả năng tổng quát hóa và độ chính xác của mô hình trên các tập dữ liệu lớn như MS COCO. Nhờ đó, YOLOv4 đạt được hiệu năng tốt hơn cả YOLOv3 và các mô hình cùng thời điểm, trở thành một lựa chọn phổ biến trong các hệ thống giám sát an ninh, phát hiện giao thông và các ứng dụng thời gian thực khác.

2.2.3.5 YOLOv5

YOLOv5 là một phiên bản cải tiến không chính thức của họ mô hình YOLO, được phát triển và công bố bởi công ty Ultralytics vào năm 2020. Khác với các phiên bản trước được công bố qua các bài báo học thuật trên arXiv, YOLOv5 được phát hành trực tiếp trên nền tảng GitHub và được viết hoàn toàn bằng ngôn ngữ lập trình Python, sử dụng framework PyTorch thay vì Darknet như các phiên bản trước (Jocher, 2020). Mô hình YOLOv5 được thiết kế linh hoạt với nhiều phiên bản khác nhau như YOLOv5s, YOLOv5m, YOLOv5l và YOLOv5x, nhằm cân bằng giữa tốc độ xử lý và độ chính xác. Một số cải tiến nổi bật của YOLOv5 bao gồm sử dụng **auto-learning bounding box anchors**, **Mosaic data augmentation**, **CIoU loss function**, và kỹ thuật **transfer learning** mạnh mẽ. Nhờ đó, YOLOv5 được đánh giá cao trong các bài toán nhận dạng đối tượng thời gian thực với hiệu suất vượt trội và dễ triển khai trong môi trường sản xuất. Mặc dù không phải là phiên bản chính thức được nhóm tác giả ban đầu phát hành, YOLOv5 vẫn nhanh chóng trở thành một trong những mô hình phổ biến và được sử dụng rộng rãi trong cộng đồng nghiên cứu và ứng dụng công nghiệp.

2.2.3.6 YOLOv6

YOLOv6 là một framework phát hiện đối tượng đơn giai đoạn được thiết kế đặc biệt cho các ứng dụng công nghiệp, với mục tiêu tối ưu hóa cả độ chính xác và tốc độ xử lý. Mô hình này giới thiệu kiến trúc **EfficientRep Backbone** và **Rep-PAN Neck**, giúp cải thiện khả năng trích xuất và tổng hợp đặc trưng một cách hiệu quả. Ngoài ra, YOLOv6 áp dụng kỹ thuật **self-distillation** và các chiến lược huấn luyện tiên tiến để nâng cao hiệu suất mà không tăng chi phí tính toán (Li et al., 2022). Kết quả đánh giá trên tập dữ liệu COCO cho thấy YOLOv6-N đạt 37.5% AP với tốc độ 1187 FPS, trong khi YOLOv6-S đạt 45.0% AP tại 484 FPS, vượt trội so với các mô hình cùng phân khúc như YOLOv5-S và YOLOX-S.

2.2.3.7 YOLOv7

YOLOv7 là một mô hình phát hiện đối tượng thời gian thực tiên tiến, được phát triển nhằm cải thiện cả tốc độ và độ chính xác so với các phiên bản trước đó (Wang et al., 2022). Mô hình này giới thiệu kiến trúc **Extended Efficient Layer Aggregation Network (E-ELAN)**, giúp tăng cường khả năng học sâu và hiệu quả tính toán. Ngoài ra, YOLOv7 tích hợp các kỹ thuật huấn luyện như **Trainable Bag-of-Freebies** và **Coarse-to-Fine Lead Head**, nhằm tối ưu hóa hiệu suất mà không làm tăng chi phí tính toán. Theo đánh giá trên bộ dữ liệu MS COCO, YOLOv7 đạt độ chính xác trung bình (AP) 56.8% với tốc

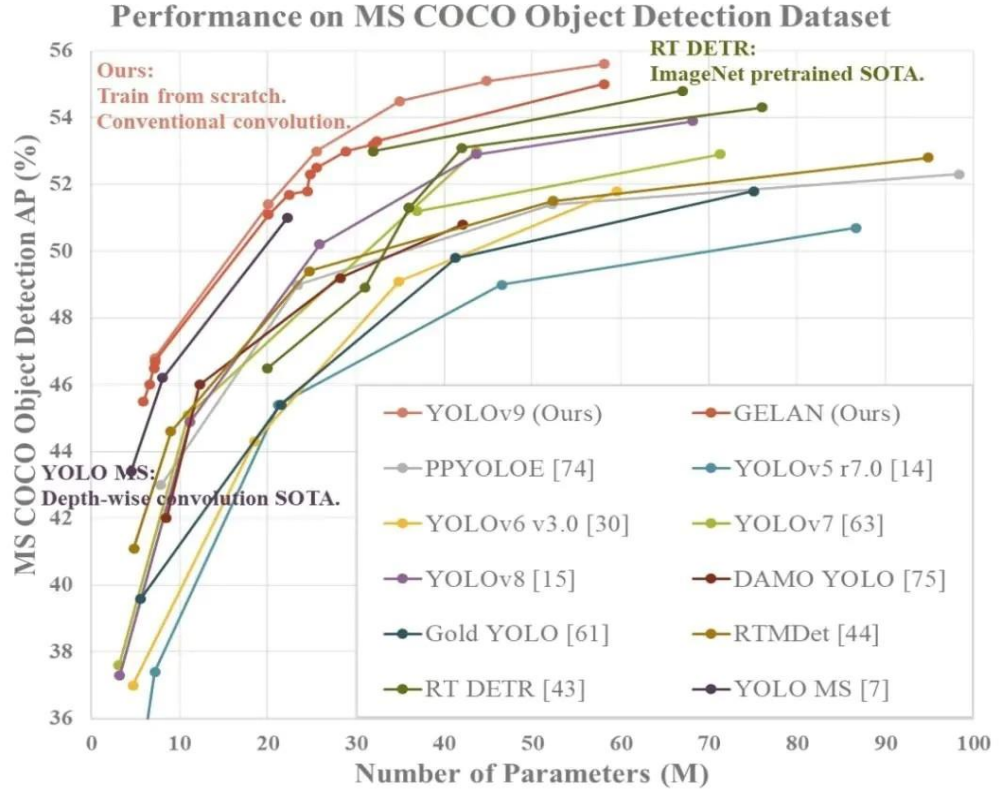
độ xử lý từ 5 FPS đến 160 FPS, vượt trội hơn các mô hình phát hiện đối tượng thời gian thực khác như YOLOv5 và YOLOX.

2.2.3.6 YOLOv8

YOLOv8 là phiên bản mới nhất trong dòng mô hình phát hiện đối tượng You Only Look Once (YOLO), được phát hành bởi Ultralytics vào năm 2023. Mô hình này giới thiệu nhiều cải tiến đáng kể về kiến trúc và hiệu suất, bao gồm việc chuyển sang thiết kế anchor-free, sử dụng module C2f thay cho CSP trong phần backbone, và áp dụng kỹ thuật SPPF (Spatial Pyramid Pooling – Fast) để tăng cường khả năng trích xuất đặc trưng (Ultralytics, n.d.). Ngoài ra, YOLOv8 hỗ trợ nhiều tác vụ thị giác máy tính như phát hiện đối tượng, phân đoạn ảnh, ước lượng tư thế và phân loại hình ảnh, đồng thời cung cấp các phiên bản mô hình với kích thước khác nhau (nano, small, medium, large, x-large) để phù hợp với các yêu cầu về tài nguyên và độ chính xác. Theo đánh giá trên tập dữ liệu COCO, YOLOv8 đạt hiệu suất mAP cao hơn so với các phiên bản trước như YOLOv5 và YOLOv7, đồng thời cải thiện tốc độ suy luận, đặc biệt khi triển khai trên GPU.

2.2.3.7 YOLOv9

YOLOv9 là phiên bản mới nhất trong dòng mô hình phát hiện đối tượng You Only Look Once (YOLO), được phát hành bởi Ultralytics vào năm 2024 (Yaseen, 2024). Mô hình này giới thiệu nhiều cải tiến đáng kể về kiến trúc và hiệu suất, bao gồm việc sử dụng kiến trúc Generalized Efficient Layer Aggregation Network (GELAN) để tối ưu hóa khả năng trích xuất đặc trưng và kỹ thuật Programmable Gradient Information (PGI) nhằm cải thiện quá trình huấn luyện. Ngoài ra, YOLOv9 tích hợp các kỹ thuật như Depthwise Convolutions và kiến trúc nhẹ C3Ghost, giúp giảm độ phức tạp tính toán mà vẫn duy trì độ chính xác cao. Theo đánh giá trên tập dữ liệu COCO, YOLOv9 đạt hiệu suất mAP cao hơn so với các phiên bản trước như YOLOv8, đồng thời cải thiện tốc độ suy luận, đặc biệt khi triển khai trên các thiết bị từ biên đến GPU hiệu năng cao.



Hình 5: Biểu đồ hiệu suất của Yolov9

2.2.3.7 YOLOv10

YOLOv10 là phiên bản mới nhất trong dòng mô hình phát hiện đối tượng You Only Look Once (YOLO), được phát triển bởi nhóm nghiên cứu tại Đại học Thanh Hoa và công bố tại hội nghị NeurIPS 2024 (Perrin, 2024). Mô hình này giới thiệu nhiều cải tiến đáng kể về kiến trúc và hiệu suất, bao gồm việc sử dụng chiến lược gán nhãn kép nhất quán (Consistent Dual Assignments) để loại bỏ nhu cầu sử dụng Non-Maximum Suppression (NMS) trong quá trình huấn luyện, giúp giảm độ trễ suy luận và tăng hiệu quả triển khai. Ngoài ra, YOLOv10 áp dụng thiết kế mô hình dựa trên hiệu suất-độ chính xác (Efficiency-Accuracy Driven Model Design), tối ưu hóa các thành phần như downsampling tách biệt không gian-kênh (Spatial-Channel Decoupled Downsampling) và thiết kế hộp giới hạn hướng dẫn theo thứ hạng (Rank-Guided Box Design), nhằm cải thiện hiệu suất mà không tăng chi phí tính toán. Theo đánh giá trên tập dữ liệu COCO, YOLOv10-S nhanh hơn 1,8 lần và có số lượng tham số cũng như FLOPs ít hơn 2,8 lần so với RT-DETR-R18 với độ chính xác tương đương. So với YOLOv9-C, YOLOv10-B giảm 46% độ trễ và 25% số lượng tham số trong khi vẫn duy trì hiệu suất tương đương. Những cải tiến này giúp YOLOv10 trở thành một giải pháp hiệu quả cho các ứng dụng phát hiện đối tượng theo thời gian thực trên các thiết bị từ biên đến GPU hiệu năng cao.

2.2.3.7 YOLOv11

YOLOv11 là phiên bản mới nhất trong dòng mô hình phát hiện đối tượng You Only Look Once (YOLO), được phát triển bởi nhóm Ultralytics và công bố vào cuối năm 2024 (Khanam & Hussain, 2024). Mô hình này giới thiệu nhiều cải tiến đáng kể về kiến trúc và hiệu suất, bao gồm việc sử dụng các khối C3k2 (Cross Stage Partial với kích thước kernel 2), SPPF (Spatial Pyramid Pooling - Fast) và C2PSA (Convolutional block with Parallel Spatial Attention) để nâng cao khả năng trích xuất đặc trưng và hiệu quả huấn luyện (Khanam & Hussain, 2024). YOLOv11 hỗ trợ nhiều tác vụ thị giác máy tính như phát hiện đối tượng, phân đoạn thể hiện, ước lượng tư thế và phát hiện đối tượng định hướng (OBB), với các phiên bản mô hình từ Nano đến Extra-Large phù hợp cho cả thiết bị biên và hệ thống GPU hiệu năng cao (*Ultralytics YOLO11 - Ultralytics YOLO Tài Liệu*, n.d.). Theo đánh giá trên tập dữ liệu COCO, YOLOv11 đạt hiệu suất mAP cao hơn so với các phiên bản trước như YOLOv10 và YOLOv9, đồng thời cải thiện tốc độ suy luận, đặc biệt khi triển khai trong các hệ thống giao thông thông minh và giám sát thời gian thực.

2.3 PHƯƠNG PHÁP TRANSFER LEARNING

2.3.1 Giới thiệu về Transfer Learning

Trong quá trình xây dựng mô hình chắc hẳn chúng ta đã từng gặp trường hợp mô hình của mình dự đoán không chuẩn. Thường thì mọi người sẽ nghi ngờ nguyên nhân xuất phát từ việc chúng ta gán nhãn dữ liệu bị sai, nhưng khi kiểm tra lại thì vấn đề không nằm ở nhãn dữ liệu. Vậy vấn đề xuất phát từ đâu? Trong chương này thì tôi sẽ trình bày nguyên nhân dẫn đến mô hình không hiệu quả, đồng thời sẽ đề xuất phương pháp được áp dụng phổ biến giúp cải thiện độ chính xác và giảm chi phí, tiết kiệm thời gian. Đó là phương pháp học chuyển giao (Transfer Learning) (Brownlee, 2017).

Nguyên nhân của mô hình dự báo kém, nếu chúng ta bỏ qua những yếu tố như gán nhãn ảnh sai, ảnh bị mập mờ, bị che khuất, ... thì thông thường sẽ có bốn yếu tố như sau:

- Dữ liệu nhỏ: bộ dữ liệu có kích thước quá bé, do đó mô hình huấn luyện không học được các đặc trưng tổng quát để áp dụng vào tác vụ phân loại. Ví dụ như với cùng một bài toán phân loại chó và mèo thì nếu đối với tập dữ liệu chó và mèo của Việt Nam mình có 100 ảnh, số lượng ảnh này còn quá ít khi đó nếu đem mô hình huấn luyện với tập dữ liệu 100 ảnh của Việt Nam mà sử dụng cho những bộ dữ liệu lớn hơn thì sẽ dẫn đến khả năng là dự báo sai.

- Mất cân bằng dữ liệu: khi mô hình mất cân bằng dữ liệu thì việc dự đoán các mẫu thuộc nhóm thiểu số sẽ khó khăn hơn.

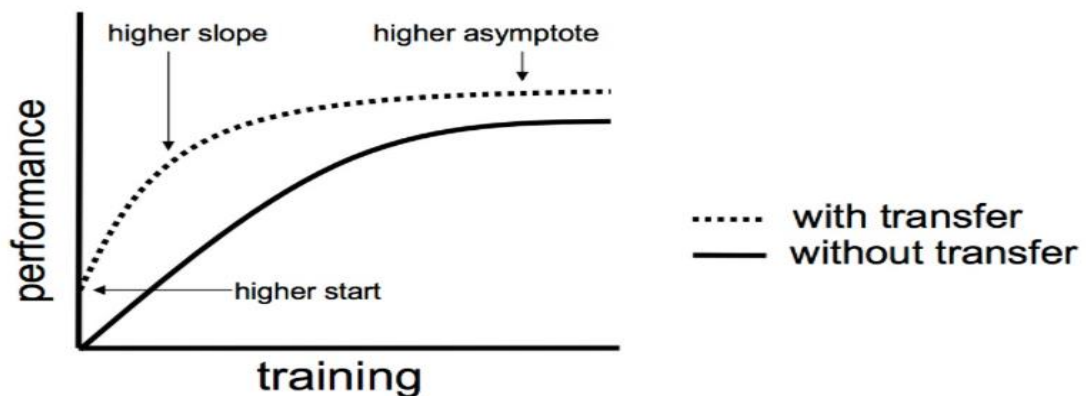
- Kiến trúc mô hình lựa chọn phức tạp so với bộ dữ liệu: đối với những bộ dữ liệu lớn lên tới vài triệu ảnh thì việc lựa chọn mô hình có kiến trúc phức tạp sẽ rất phù hợp vì khi đó sẽ mạng lại độ chính xác cao. Còn đối với bộ dữ liệu có kích thước nhỏ thì việc lựa chọn kiến trúc mô hình phức tạp sẽ làm giảm đi độ chính xác. Nguyên nhân có thể là do những mô hình có kiến trúc phức tạp thường xảy ra overfitting.

- Quá trình tối ưu hóa gặp khó khăn: có thể bạn đã thiết lập learning rate, chưa tốt nên khiến mô hình huấn luyện lâu hội tụ hoặc chưa đạt tới điểm global optimal.

Vậy vai trò của học chuyển giao là gì? Có ba vai trò mà chúng ta có thể dễ nhìn thấy nhất ở phương pháp học chuyển giao đó là:

- Chuyển giao tri thức: Tận dụng lại các tri thức của mô hình trước đó và cải thiện nó lại để tốt hơn.

- Cải thiện độ chính xác (Brownlee, 2019) và tiết kiệm chi phí: ví dụ như trong bài toán phân loại chó và mèo, nếu chúng ta huấn luyện ngay từ đầu bạn sẽ tốn nhiều epochs huấn luyện hơn để đạt được độ chính xác cao. Tuy nhiên nếu chúng ta tận dụng pretrained-model thì cần ít epochs huấn luyện hơn để đạt được một độ chính xác mong đợi. Thậm chí độ chính xác của nó còn có thể cao hơn khi không áp dụng transfer learning.



Hình 6: Sơ đồ so sánh hiệu suất của mô hình trước và sau khi áp dụng Transfer Learning (Khánh, n.d.)

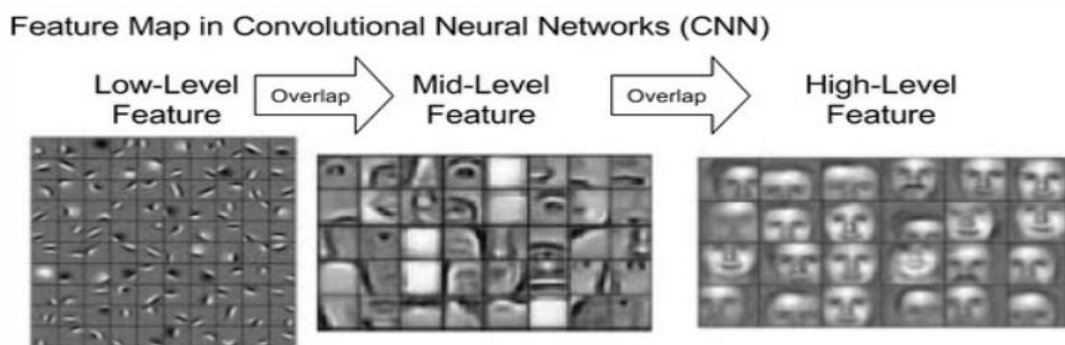
- Hiệu quả với dữ liệu nhỏ: trong trường hợp mà bộ dữ liệu có kích thước quá nhỏ và khó có thể tìm kiếm và mở rộng thêm thì các mô hình được huấn luyện từ chúng sẽ khó có thể dự báo tốt, tận dụng lại tri thức từ các tác vụ phân loại sẽ giúp các mô hình được huấn luyện dự báo tốt hơn với dữ liệu mới vì mô hình được học trên cả hai nguồn tri thức đó là dữ liệu huấn luyện và dữ liệu mà nó đã được học từ trước đó.

- Từ đồ thị cho thấy khi sử dụng transfer learning sẽ có 3 lợi thế là có điểm khởi đầu của accuracy tốt hơn, accuracy có tốc độ nhanh hơn, đường tiệm cận của độ chính xác tối ưu cao hơn.

2.3.2 Transfer Learning

Quá trình áp dụng tri thức đã được học từ mô hình trước sang bài toán hiện tại được gọi là transfer learning (*A Gentle Introduction to Transfer Learning for Deep Learning - MachineLearningMastery.Com*, n.d.).

Như chúng ta đã biết thì layers CNN về bản chất là một feature extraction mà mỗi một layer CNN sẽ có tác dụng trích lọc đặc trưng theo những level khác nhau.



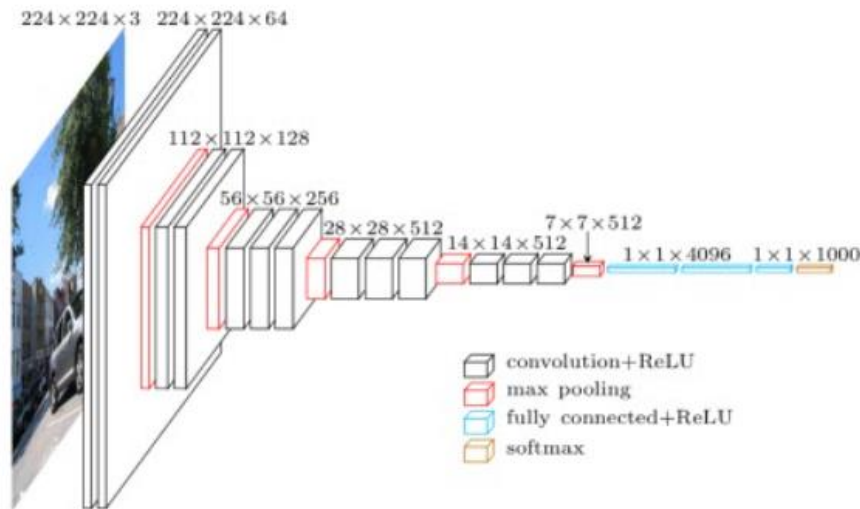
Hình 7: Các đặc trưng học được từ mạng CNN

Ở những Convolutional Layers đầu tiên, các bộ lọc phát hiện được các chi tiết chung dưới dạng các nét ngang, dọc và các cạnh tranh của ảnh. Đây là những đặc trưng bậc thấp (low level feature) và khá chung chung. Chúng ta chưa thể nhận biết được vật thể dựa trên những nét này. Ở những Convolutional Layers cuối cùng là những đặc trưng bậc cao (high level feature) được tổng hợp từ đặc trưng bậc thấp. Đây là những đặc trưng tốt và có sức mạnh phân loại các classes.

Quá trình transfer learning sẽ tận dụng lại các đặc trưng được học từ những pretrained-model. Để hiểu hơn về cách thức chuyển giao, chúng ta cùng tìm hiểu về kiến trúc của mô hình sử dụng transfer learning:

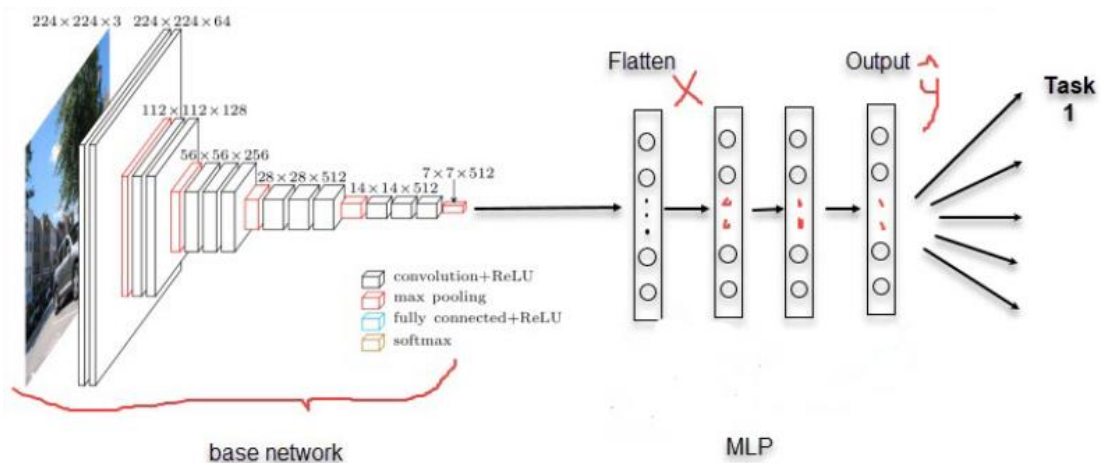
- Phrase 1: Là một mạng Base Network có tác dụng trích lọc đặc trưng được cấu tạo từ các Convolutional 2D Layers. Base Network sẽ được trích xuất từ một phần của pretrained-model sau khi loại bỏ các top fully connected layers. Để dễ hình dung mình giả định model pretrained được sử dụng là VGG16, một kiến trúc khá tốt được google phát triển vào năm 2014. Điểm cải tiến của VGG16 so với các kiến trúc CNN trước đó là sử dụng nhiều Convolutional 2D Layers nối tiếp nhau. Cụ thể các layers có cấu trúc $[[Conv]_n - MaxPool]$ m thay

vì [Conv-MaxPool] m, với m, n là tần suất xuất hiện của các khối mạng được lặp lại bao bọc trong ngoặc vuông.



Hình 8: Kiến trúc của mạng VGG16 được sử dụng làm base network trong transfer learning.

- Phrase 2: Là các Fully Connected Layers giúp giảm chiều dữ liệu và tính toán phân phối xác suất ở output. Bản chất Fully Connected Layers chính là một mạng MLP (Multiple Layer Perceptron), một kiến trúc nguyên thủy nhất của thuật toán neural network. Số lượng các units ở output chính bằng với số lượng classes của bài toán phân loại. Các hệ số của fully connected layers sẽ được khởi tạo một cách ngẫu nhiên.



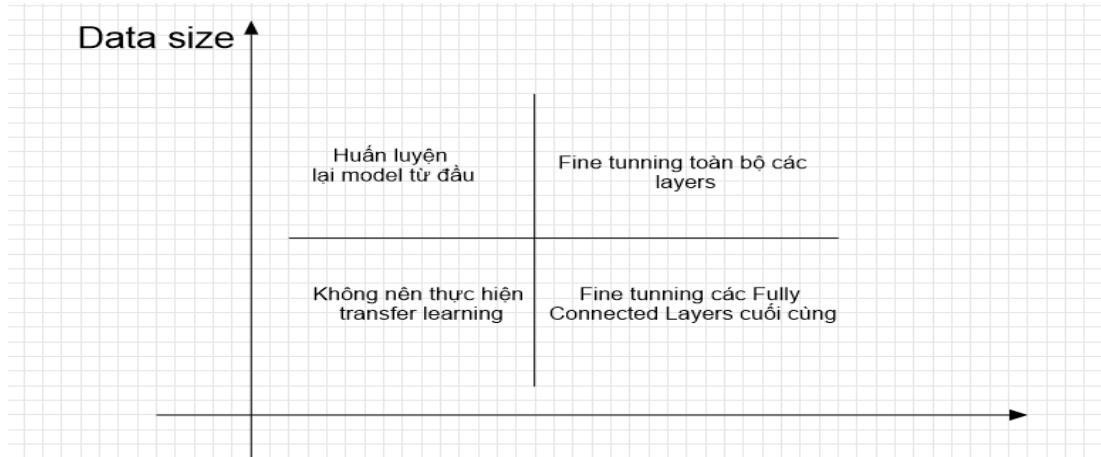
Hình 9: Kiến trúc based network kết hợp với fully connected layers

Quá trình khởi tạo mô hình chúng ta sẽ tận dụng lại các weight của base network. Dữ liệu ảnh sau khi đi qua based network sẽ tạo ra những đặc trưng tốt, những đặc trưng này chính là đầu vào X cho mạng MLP để dự báo y^{\wedge} .

Hệ số W và b được khởi tạo ngẫu nhiên. Các hệ số của base network được load lại từ pretrain model.

2.3.3 Những điều cần lưu ý khi áp dụng transfer learning

Transfer learning theory kích thước của dữ liệu



Hình 10: Chiến lược áp dụng transfer learning.

- Đối với dữ liệu nhỏ: Train lại toàn bộ các layers sẽ làm mất đi các đặc trưng đã được học từ model pretrained và dẫn tới mô hình dự báo sẽ không chính xác. Chúng ta chỉ nên train lại các fully connected layers cuối
- Đối với dữ liệu lớn và giống domain: Có thể train lại model trên toàn bộ layers. Nhưng để quá trình huấn luyện nhanh hơn thì chúng ta sẽ thực hiện bước khởi động (warm up) và sau đó mới fine tuning lại mô hình.
- Đối với dữ liệu lớn và khác domain: Chúng ta nên huấn luyện lại model từ đầu vì pretrain-model không tạo ra được các đặc trưng tốt cho dữ liệu khác domain.

Khi nào thì thực hiện Transfer Learning

- Chỉ nên transfer learning giữa 2 mô hình có cùng domain. pretrained-model A và mô hình cần huấn luyện B không có chung domain về dữ liệu thì các đặc trưng học được từ bộ feature extractor của A sẽ không thực sự hữu ích trong việc phân loại của mô hình B. Cụ thể hơn. Nếu bạn muốn xây dựng một ứng dụng âm thanh đánh thức trợ lý ảo của google bằng tiếng Việt khi nói từ: "dậy đi google". Bạn đã có sẵn pretrained-model A đối với tác vụ speech to text nhưng huấn luyện trên Tiếng Anh. Như vậy bạn không nên thực hiện transfer learning trong trường hợp này. Như trong ví dụ của mình thì pretrained-model của imagenet đã bao gồm 2 classes dog and cat.

- Dữ liệu huấn luyện pretrained-model A phải lớn hơn so với mô hình B. Nếu chúng ta transfer hệ số từ một pretrained-model được huấn luyện trên dữ liệu có kích thước nhỏ thì các đặc trưng học được từ mô hình A sẽ không tổng quát để giúp ích phân loại dữ liệu mô hình B.

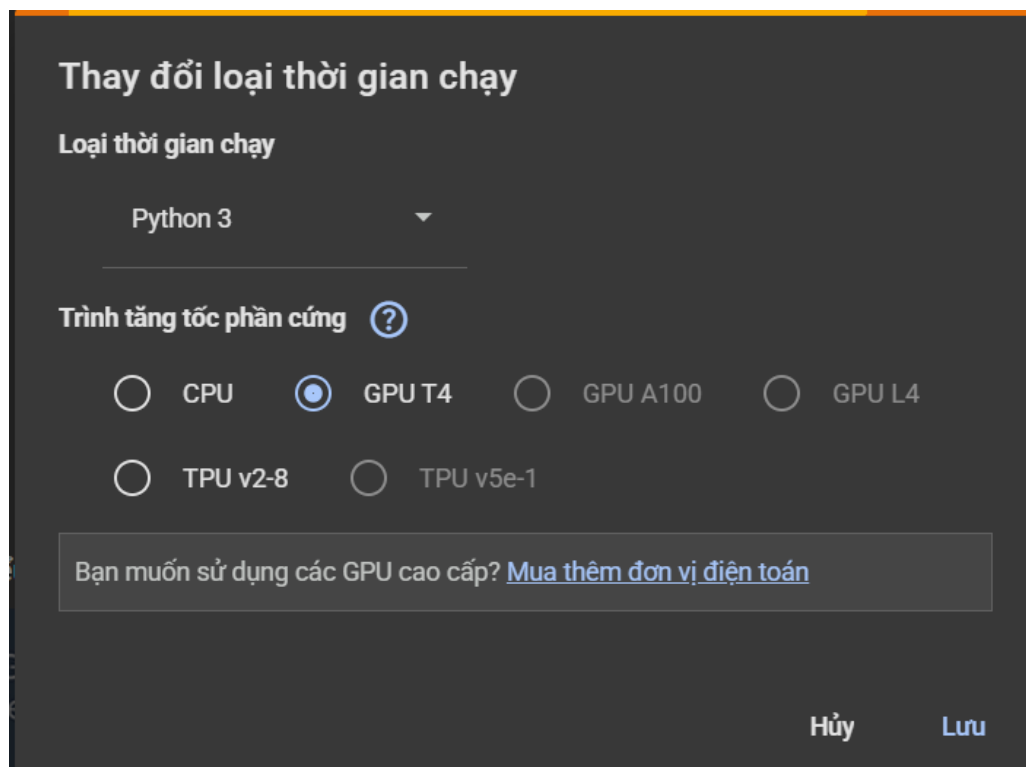
- Pretrained-model A phải là mô hình có phẩm chất tốt. Đây là một yêu cầu hiển nhiên vì mô hình tốt mới tạo ra được những đặc trưng tốt.

2.3.4 Cách train YOLO

Để train YOLO v9 trên Google Colab, bạn có thể làm theo các bước sau:

Bước 1: Chuẩn bị Google Colab

- Mở Google Colab và chọn **GPU runtime**:
- Vào Runtime > Change runtime type > chọn GPU.



Hình 11: Chọn runtime

Bước 2: Cài đặt YOLOv9

Lệnh cài đặt YOLOv9

```
!git clone https://github.com/WongKinYiu/yolov9
```

```
%cd yolov9
```

```
!pip install -r requirements.txt
```

Bước 3: Chuẩn bị dữ liệu huấn luyện (dạng YOLO)

Name	Date modified	Type	Size
images	11/24/2024 4:06 PM	File folder	
labels	11/24/2024 4:06 PM	File folder	
classes	8/27/2021 11:27 PM	Tài liệu văn bản	1 KB
data	11/14/2024 3:00 PM	Yaml Source File	1 KB

Hình 12: Cấu trúc thư mục

Image: Chứa ảnh huấn luyện

Labels: Chứa nhãn của ảnh

Classes: Số lớp cần huấn luyện

Data: File cấu hình

Bước 4: Huấn luyện YOLOv9

- Tải model backbone

Lệnh:

```
!wgethttps://github.com/WongKinYiu/yolov9/releases/download/yolov9-c/yolov9-c.pt
```

- Chạy huấn luyện

Lệnh:

```
!python train.py --img 640 --batch 16 --epochs 50 --data my_data.yaml -
-cfg cfg/yolov9-c.yaml --weights yolov9-c.pt --name my_yolov9_model
```

Bước 5: Kiểm tra kết quả

Sau khi huấn luyện xong, mô hình sẽ lưu ở:

```
runs/train/my_yolov9_model/weights/best.pt
```

2.4 SƠ LƯỢC VỀ ANDROID

2.4.1 Giới thiệu

Android là một hệ điều hành có dạng mã nguồn mở, nó hoạt động dựa trên nền tảng Linux và được thiết kế dành riêng cho những thiết bị di động cảm ứng hoặc máy tính bảng. Trước đây, hệ điều hành này được phát triển bởi tổng công ty Android và được tài trợ bởi Google. Cho đến năm 2005 thì Google đã mua lại hệ điều hành này và cho ra mắt người dùng vào năm 2007. Android này sở hữu mã nguồn mở nên lập trình viên có thể dễ dàng điều chỉnh và phân phối nó một cách tự do. Đây chính là một trong những yếu tố đã giúp cho Android trở thành nền tảng xây dựng điện thoại thông minh phát triển nhất trên thế giới.

Hiện tại, Android đã chiếm 65% so với thị phần điện thoại thông minh trên toàn thế giới vào quý 3 năm 2012. Theo điều tra thì đã có khoảng 500 triệu thiết bị được kích hoạt và có đến “1.3 triệu lượt được hoạt mỗi ngày”. Vào tháng 10/2020 thì android đã có hơn 700.000 ứng dụng và số lượng tải từ Google Play ước tính lên khoảng 25 tỷ lượt. Mặc dù có sự ra đời của iOS của Apple thì khiến Android có phần nào ảnh hưởng. Tuy nhiên, Android vẫn đứng ở vị trí đầu tiên trong thị phần thế giới (*Android Là Gì - Tổng Quan Về Hệ Điều Hành Di Động Số 1*, n.d.).

Bảng 1: Các phiên bản Android

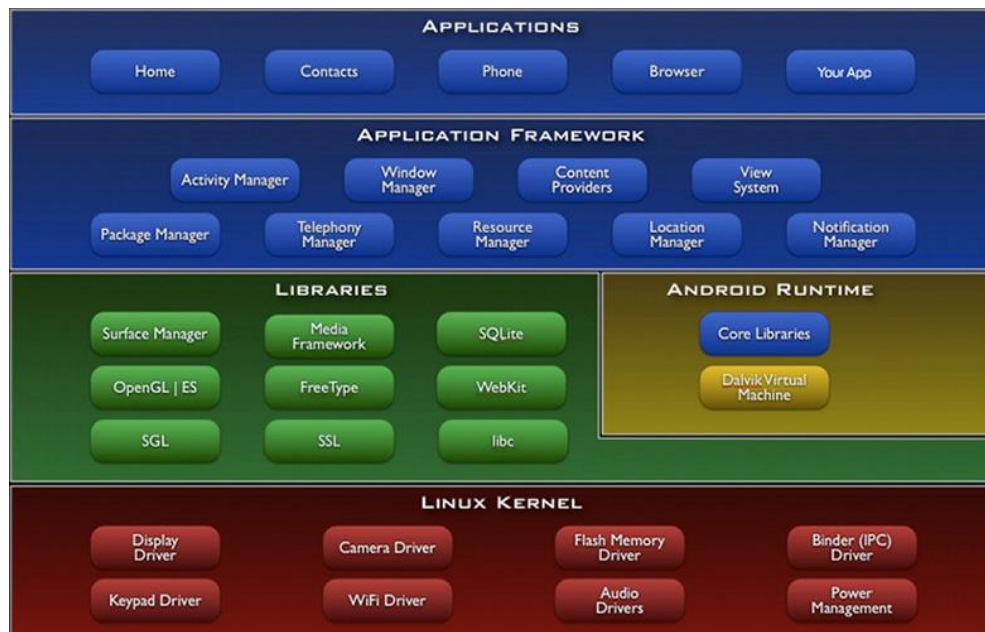
Code name	Version numbers	API level	Release date
No codename	1.0	1	September 23, 2008
No codename	1.1	2	February 9, 2009
Cupcake	1.5	3	April 27, 2009
Donut	1.6	4	September 15, 2009
Eclair	2.0 - 2.1	5 - 7	October 26, 2009
Froyo	2.2 - 2.2.3	8	May 20, 2010
Gingerbread	2.3 - 2.3.7	9 - 10	December 6, 2010
Honeycomb	3.0 - 3.2.6	11 - 13	February 22, 2011
Ice Cream Sandwich	4.0 - 4.0.4	14 - 15	October 18, 2011
Jelly Bean	4.1 - 4.3.1	16 - 18	July 9, 2012
KitKat	4.4 - 4.4.4	19 - 20	October 31, 2013
Lollipop	5.0 - 5.1.1	21- 22	November 12, 2014

Marshmallow	6.0 - 6.0.1	23	October 5, 2015
Nougat	7.0	24	August 22, 2016
Nougat	7.1.0 - 7.1.2	25	October 4, 2016
Oreo	8.0	26	August 21, 2017
Oreo	8.1	27	December 5, 2017
Pie	9.0	28	August 6, 2018
Android 10	10.0	29	September 3, 2019
Android 11	11	30	September 8, 2020
Android 12	12	31	October 4, 2021
Android 12L	12.1	32	March 7, 2022
Android 13	13	33	August 15, 2022
Android 14	14	34	October 4, 2023
Android 15	15	35	September 3, 2024
Android 16	16	36	April 2, 2025

2.4.2 Kiến trúc của ứng dụng Android

Các thành phần chính của kiến trúc android (*Android Là Gì - Tổng Quan Về Hệ Điều Hành Di Động Số 1*, n.d.) như

- Application
- Khung ứng dụng
- Android Runtime
- Thư viện Android
- Linux Kernel



Hình 13: Kiến trúc của Android

Application: Đây là tầng cao nhất trong kiến trúc Android, nơi chứa các ứng dụng mà người dùng tương tác hằng ngày như Danh bạ (Contacts), Tin nhắn (Messages), Trình duyệt web (Browser), và các ứng dụng cài đặt từ Google Play. Mỗi ứng dụng Android được xây dựng bằng Java, Kotlin hoặc ngôn ngữ tương thích, với cấu trúc đặc trưng bao gồm Activity, Service, Broadcast Receiver và Content Provider. Tầng này tương tác trực tiếp với người dùng nhưng phụ thuộc vào tầng Khung ứng dụng để sử dụng các tài nguyên hệ thống.

Application Framework (Khung ứng dụng): Khung ứng dụng cung cấp một tập hợp API cho các nhà phát triển sử dụng để xây dựng ứng dụng. Nó quản lý vòng đời của các thành phần ứng dụng, tài nguyên, giao diện người dùng, thông báo, và quản lý quyền truy cập dữ liệu.

Android Runtime (ART): ART là môi trường thực thi ứng dụng trên Android, thay thế cho Dalvik kể từ Android 5.0 (Lollipop). ART sử dụng Ahead-of-Time Compilation (AOT) để biên dịch mã bytecode sang mã máy trong quá trình cài đặt, giúp cải thiện hiệu suất và giảm mức tiêu thụ năng lượng. Ngoài ra, ART còn cung cấp bộ thu gom rác (Garbage Collector), quản lý bộ nhớ và hỗ trợ gỡ lỗi.

Android Libraries (Thư viện Android): Là tập hợp các thư viện C/C++ cấp thấp cung cấp các tính năng cốt lõi cho hệ điều hành. Các thư viện này bao gồm:

- Surface Manager – hiển thị giao diện người dùng.

- Media Framework – hỗ trợ định dạng âm thanh và video (MP3, AAC, MPEG4...).
- SQLite – cung cấp cơ sở dữ liệu nhẹ tích hợp sẵn.
- OpenGL ES – hỗ trợ đồ họa 2D và 3D cho các ứng dụng. Các thư viện này đóng vai trò là cầu nối giữa hệ điều hành và phần mềm ứng dụng.

Linux Kernel: Đây là tầng thấp nhất của kiến trúc Android, đóng vai trò nền tảng hệ điều hành. Android sử dụng kernel Linux (hiện tại là nhánh riêng của Linux Long-Term Support) để quản lý bộ nhớ, tiến trình, mạng, và nhất là bảo mật hệ thống. Kernel cung cấp giao diện trừu tượng giữa phần cứng và các thành phần phần mềm phía trên. Ngoài ra, nó còn hỗ trợ trình điều khiển thiết bị (device drivers) như màn hình cảm ứng, camera, GPS, Bluetooth và Wi-Fi.

2.4.3 Môi trường phát triển của ứng dụng

Môi trường phát triển ứng dụng Android đóng vai trò then chốt trong quá trình xây dựng các ứng dụng di động hiện đại trên nền tảng hệ điều hành Android. Với hệ sinh thái phát triển phong phú và hỗ trợ mạnh mẽ từ Google, các nhà phát triển có thể dễ dàng tiếp cận và triển khai ứng dụng từ ý tưởng đến sản phẩm hoàn chỉnh thông qua bộ công cụ Android được cung cấp chính thức.

Android Studio là môi trường phát triển tích hợp (IDE – Integrated Development Environment) chính thức được Google phát hành, thay thế cho Eclipse Android Development Tools từ năm 2014. Dựa trên IntelliJ IDEA, Android Studio cung cấp đầy đủ các công cụ phục vụ cho việc lập trình, thiết kế giao diện, mô phỏng thiết bị, và kiểm thử ứng dụng. IDE này hỗ trợ các ngôn ngữ lập trình chính như Java, Kotlin và XML – trong đó Kotlin đã được Google công nhận là ngôn ngữ chính thức từ năm 2017. Giao diện thiết kế trực quan (UI Designer) trong Android Studio cho phép lập trình viên kéo–thả các thành phần giao diện và theo dõi sự thay đổi theo thời gian thực.

Bên cạnh IDE, Android Software Development Kit (SDK) đóng vai trò trung tâm, cung cấp các API, thư viện, trình biên dịch và công cụ dòng lệnh cần thiết để biên dịch, đóng gói và triển khai ứng dụng. SDK này thường được tích hợp sẵn trong Android Studio, giúp lập trình viên dễ dàng thiết lập môi trường làm việc mà không cần cấu hình phức tạp.

Trình giả lập (AVD – Android Virtual Device) là một phần quan trọng của môi trường phát triển, cho phép nhà phát triển kiểm tra ứng dụng trên nhiều cấu hình thiết bị ảo khác nhau như kích thước màn hình, phiên bản Android, và độ phân giải. Điều này đặc biệt hữu ích khi kiểm thử giao diện người dùng và tính năng trên nhiều loại thiết bị mà không cần sở hữu phần cứng thực tế.

Ngoài ra, các công cụ hỗ trợ như Gradle – công cụ tự động hóa biên dịch và quản lý thư viện phụ thuộc, cũng giúp tối ưu quá trình phát triển và triển khai ứng dụng. Đồng thời, hệ thống kiểm thử tự động (Instrumentation Tests, Unit Tests) được tích hợp giúp đảm bảo chất lượng phần mềm.

Nhìn chung, môi trường phát triển Android là một hệ thống mạnh mẽ, linh hoạt và liên tục được cải tiến để đáp ứng nhu cầu phát triển ứng dụng di động hiện đại, hỗ trợ lập trình viên xây dựng ứng dụng hiệu quả với khả năng mở rộng và bảo trì tốt.

2.4.4 Ngôn ngữ lập trình Kotlin

Để có thể phát triển ứng dụng Android chúng ta thường sử dụng các ngôn ngữ như: Java, JavaScript, Kotlin. Trong khuôn khổ đề tài này, tôi sẽ xây dựng một ứng dụng Android sử dụng Kotlin – một ngôn ngữ lập trình hiện đại, linh hoạt và mạnh mẽ. Kotlin được thiết kế với mục tiêu mang lại trải nghiệm lập trình tối ưu, giúp các nhà phát triển tạo ra những ứng dụng chất lượng cao với thời gian phát triển được rút ngắn. Với cú pháp gọn gàng, rõ ràng và các tính năng hỗ trợ lập trình hướng đối tượng cũng như lập trình hàm, Kotlin trở thành một lựa chọn lý tưởng để phát triển ứng dụng Android. Việc sử dụng ngôn ngữ này không chỉ giúp tôi tiếp cận các công nghệ tiên tiến, mà còn đảm bảo rằng sản phẩm cuối cùng sẽ đáp ứng tốt các tiêu chuẩn hiện đại và nhu cầu của người dùng.

Kotlin là một ngôn ngữ lập trình đa nền tảng (cross-platform) được phát triển bởi JetBrains, một công ty phát triển phần mềm có trụ sở tại Nga. Kotlin được thiết kế để chạy trên Java Virtual Machine (JVM) và có thể sử dụng để phát triển ứng dụng di động, web và backend (TopDev, 2023).

Kotlin được thiết kế với mục tiêu giải quyết những hạn chế của Java như cú pháp rườm rà, lỗi null (NullPointerException) và thiếu hỗ trợ cho lập trình hàm. Kotlin vẫn đảm bảo khả năng tương thích hoàn toàn với Java, cho phép các nhà phát triển chuyển đổi hoặc tích hợp dần dần mà không ảnh hưởng đến các dự án hiện tại.

Kotlin mang đến nhiều ưu điểm nổi bật như:

- Cú pháp ngắn gọn, dễ đọc: Giúp giảm thiểu số dòng mã, nâng cao hiệu quả viết và bảo trì mã nguồn.
- An toàn với Null: Kotlin áp dụng hệ thống kiểu an toàn null tại thời điểm biên dịch, hạn chế lỗi phổ biến “null pointer exception”.
- Hỗ trợ lập trình hàm: Hỗ trợ các biểu thức lambda, extension function, higher-order function, giúp lập trình viên viết mã linh hoạt và hiệu quả hơn.

- Tương thích hoàn toàn với Java: Kotlin có thể sử dụng toàn bộ thư viện Java hiện có, đồng thời cho phép Java gọi lại mã Kotlin một cách tự nhiên.
- Được Google hỗ trợ chính thức: Từ năm 2019, Google đã tuyên bố Android sẽ chuyển sang ưu tiên Kotlin làm ngôn ngữ chính ("Kotlin-first").

Bảng 2: So sánh Kotlin và Java

Tiêu chí	Kotlin	Java
Độ ngắn gọn của cú pháp	Ngắn gọn, dễ đọc, giảm boilerplate code	Cú pháp dài, nhiều đoạn mã mẫu (boilerplate)
An toàn với Null	Có hệ thống kiểm soát null tại compile-time	Có thể phát sinh lỗi NullPointerException
Hỗ trợ lập trình hàm	Mạnh mẽ: lambda, extension, higher-order	Giới hạn, chỉ có từ Java 8 trở đi
Hỗ trợ IDE	Hỗ trợ đầy đủ trong Android Studio	Hỗ trợ đầy đủ

2.4.5 Giới thiệu Supabase

Supabase là một nền tảng Backend-as-a-Service (BaaS) mã nguồn mở, được phát triển nhằm cung cấp một giải pháp thay thế cho Firebase của Google (*Supabase*, n.d.). Supabase cung cấp một bộ công cụ backend hoàn chỉnh bao gồm cơ sở dữ liệu PostgreSQL, hệ thống xác thực người dùng, API tự động, đăng ký thời gian thực, lưu trữ tệp và nhúng vector (*Supabase Tính Năng, Ưu Điểm, Nhược Điểm và Trường Hợp Sử Dụng*, n.d.).

Khác với Firebase, Supabase được xây dựng hoàn toàn dựa trên các công nghệ mã nguồn mở và tuân thủ các tiêu chuẩn mở (Khánh, n.d.). Điều này mang lại cho các nhà phát triển sự linh hoạt trong việc kiểm soát hạ tầng và khả năng tùy chỉnh cao hơn. Supabase hoạt động bằng cách cung cấp một cơ sở dữ liệu PostgreSQL đầy đủ, có thể được quản lý thông qua một giao diện trực quan

tương tự như bảng tính, giúp dễ tiếp cận ngay cả với những người không có chuyên môn sâu về cơ sở dữ liệu.

Một trong những điểm nổi bật của Supabase là khả năng tạo API RESTful và GraphQL tự động từ cơ sở dữ liệu chỉ trong vài giây sau khi thiết kế lược đồ (schema). Bên cạnh đó, hệ thống xác thực người dùng của Supabase hỗ trợ đa dạng phương thức như đăng nhập bằng email, mật khẩu, OTP, OAuth2 (Google, Facebook, GitHub, v.v.), phù hợp với nhiều loại ứng dụng thực tế.

Supabase đặc biệt phù hợp với các nhà phát triển muốn tập trung vào phần giao diện và trải nghiệm người dùng mà không cần tốn quá nhiều thời gian vào việc xây dựng backend từ đầu. Việc tích hợp Supabase với các framework phổ biến như React, Next.js, Flutter hay Android (Kotlin) cũng rất thuận tiện thông qua các thư viện SDK chính thức.

Ngoài ra, do được xây dựng dựa trên PostgreSQL – một hệ quản trị cơ sở dữ liệu quan hệ nổi tiếng với tính ổn định và khả năng mở rộng cao – Supabase mang lại cho nhà phát triển sự linh hoạt trong việc thao tác dữ liệu, kết hợp với tính năng phát sự kiện thời gian thực (real-time subscription) rất thích hợp cho các ứng dụng có tính tương tác cao như chat, dashboard hoặc các hệ thống theo dõi dữ liệu theo thời gian thực.

Với mô hình phát triển mã nguồn mở và cộng đồng hỗ trợ tích cực, Supabase đang ngày càng trở thành một giải pháp backend tiềm năng cho các cá nhân, nhóm phát triển khởi nghiệp cũng như các doanh nghiệp mong muốn tối ưu hoá chi phí phát triển phần mềm mà vẫn đảm bảo hiệu năng và bảo mật hệ thống.

2.4.6 Cơ sở dữ liệu PostgreSQL

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ (Relational Database Management System – RDBMS) mã nguồn mở mạnh mẽ, được phát triển bởi cộng đồng và duy trì dưới sự điều phối của PostgreSQL Global Development Group. Với lịch sử phát triển từ năm 1986, PostgreSQL đã chứng minh được độ ổn định, khả năng mở rộng và mức độ bảo mật cao, được các doanh nghiệp, tổ chức chính phủ và cộng đồng phát triển phần mềm trên toàn thế giới tin dùng (*PostgreSQL: Documentation*, n.d.).

Trong lĩnh vực phát triển ứng dụng di động, đặc biệt là ứng dụng Android, PostgreSQL đóng vai trò quan trọng trong việc quản lý dữ liệu phía máy chủ (backend). Trong mô hình client-server hiện đại, ứng dụng Android thường chỉ đảm nhận phần giao diện người dùng (frontend), trong khi phần lưu trữ, xử lý và bảo vệ dữ liệu người dùng được đảm nhiệm bởi một hệ thống backend – nơi PostgreSQL là lựa chọn phổ biến. Việc tách biệt này giúp đảm

bảo khả năng mở rộng, đồng bộ hóa dữ liệu giữa nhiều thiết bị và tăng cường khả năng kiểm soát bảo mật hệ thống.

PostgreSQL được đánh giá cao nhờ khả năng hỗ trợ phong phú các kiểu dữ liệu, bao gồm các loại dữ liệu cơ bản (chuỗi, số, ngày giờ), dữ liệu mảng, dữ liệu địa lý (spatial), dữ liệu phi cấu trúc (JSON, JSONB) và đặc biệt là hỗ trợ các tiện ích mở rộng như PostGIS (xử lý dữ liệu không gian), pgvector (truy vấn vector cho trí tuệ nhân tạo) hay full-text search (tìm kiếm toàn văn). Những đặc điểm này khiến PostgreSQL trở thành một nền tảng lý tưởng cho các ứng dụng Android hiện đại, nơi yêu cầu xử lý dữ liệu lớn, phức tạp, và có tính năng thông minh ngày càng phổ biến (*PostgreSQL: About*, n.d.).

Trong kiến trúc phát triển Android sử dụng PostgreSQL, dữ liệu thường không được truy cập trực tiếp từ thiết bị người dùng. Thay vào đó, ứng dụng sẽ giao tiếp với máy chủ thông qua các giao diện lập trình ứng dụng (API), chẳng hạn như RESTful hoặc GraphQL. Các thư viện phổ biến như Retrofit, Volley hoặc Ktor (đối với Kotlin) thường được sử dụng để thiết lập và quản lý kết nối này. Việc sử dụng API không chỉ giúp quản lý truy cập dữ liệu một cách hiệu quả mà còn cho phép kiểm soát quyền hạn người dùng, theo dõi truy cập và ghi nhận lịch sử hoạt động – những tính năng đặc biệt quan trọng trong các ứng dụng có tính bảo mật cao như ngân hàng, y tế hoặc thương mại điện tử.

Ngoài cách triển khai trực tiếp, PostgreSQL còn có thể được sử dụng gián tiếp thông qua các nền tảng backend trung gian như Supabase, Hasura hoặc Directus. Các nền tảng này cung cấp lớp giao tiếp tự động với PostgreSQL, hỗ trợ xác thực người dùng, đăng ký thời gian thực, lưu trữ tệp và API tự sinh, từ đó giúp rút ngắn đáng kể thời gian phát triển backend cho ứng dụng Android. Trong đó, Supabase – một nền tảng mã nguồn mở được xem là “Firebase mã nguồn mở” – đang ngày càng phổ biến nhờ vào sự tích hợp chặt chẽ với hệ cơ sở dữ liệu PostgreSQL (Supabase, 2025).

So sánh PostgreSQL với Firebase và SQLite trong phát triển Android

Trong quá trình phát triển ứng dụng Android, nhà phát triển thường phân vân giữa nhiều lựa chọn về cơ sở dữ liệu, phổ biến nhất là PostgreSQL, Firebase và SQLite. Bảng dưới đây tóm tắt một số điểm so sánh quan trọng giữa ba giải pháp.

Bảng 3: So sánh PostgreSQL, Firebase và SQLite

Tiêu chí	PostgreSQL	Firebase Realtime/Firestore	SQLite (cơ sở dữ liệu cục bộ)
Loại hệ quản trị	Quan hệ	NoSQL (Realtime hoặc Document-based)	Quan hệ
Vị trí triển khai	Backend (cloud/on-premise server)	Backend (Google Cloud)	Thiết bị người dùng (offline)
Khả năng mở rộng	Cao, có thể phân cụm (cluster) và nhân bản	Rất cao, theo hạ tầng Google	Hạn chế theo phần cứng thiết bị
Hỗ trợ thời gian thực	Có, thông qua Supabase hoặc giải pháp tùy chỉnh	Có sẵn	Không hỗ trợ thời gian thực
Kiểu dữ liệu hỗ trợ	Rất phong phú (JSON, spatial, vector,...)	JSON	Kiểu cơ bản (chuỗi, số, blob)

Trong quá trình tìm hiểu về các nền tảng lưu trữ như Supabase hay Firebase tôi lựa chọn PostgreSQL thông qua Supabase thay vì Firebase, với lý do chính là Supabase cung cấp dịch vụ lưu trữ tệp miễn phí ở mức sử dụng cơ bản, giúp tiết kiệm đáng kể chi phí vận hành. Cụ thể, Supabase hỗ trợ miễn phí 1 GB lưu trữ và 2 GB băng thông mỗi tháng, đủ dùng cho các ứng dụng trong giai đoạn phát triển hoặc thử nghiệm.

Ngược lại, Firebase Storage của Google có giới hạn nghiêm ngặt về dung lượng miễn phí và tính phí theo từng MB lưu trữ hoặc truyền tải, điều này có thể gây áp lực tài chính khi lượng người dùng tăng hoặc dữ liệu phát sinh nhiều.

Bên cạnh yếu tố chi phí, Supabase còn sử dụng PostgreSQL – một hệ quản trị cơ sở dữ liệu mạnh mẽ, ổn định và dễ mở rộng. Điều này mang lại lợi thế trong việc quản lý dữ liệu có cấu trúc và tích hợp linh hoạt với API backend.

Với các ưu điểm về chi phí và tính linh hoạt, Supabase là lựa chọn phù hợp cho ứng dụng Android của tôi trong giai đoạn hiện tại.

2.5 TÌM HIỂU VỀ CÔN TRÙNG

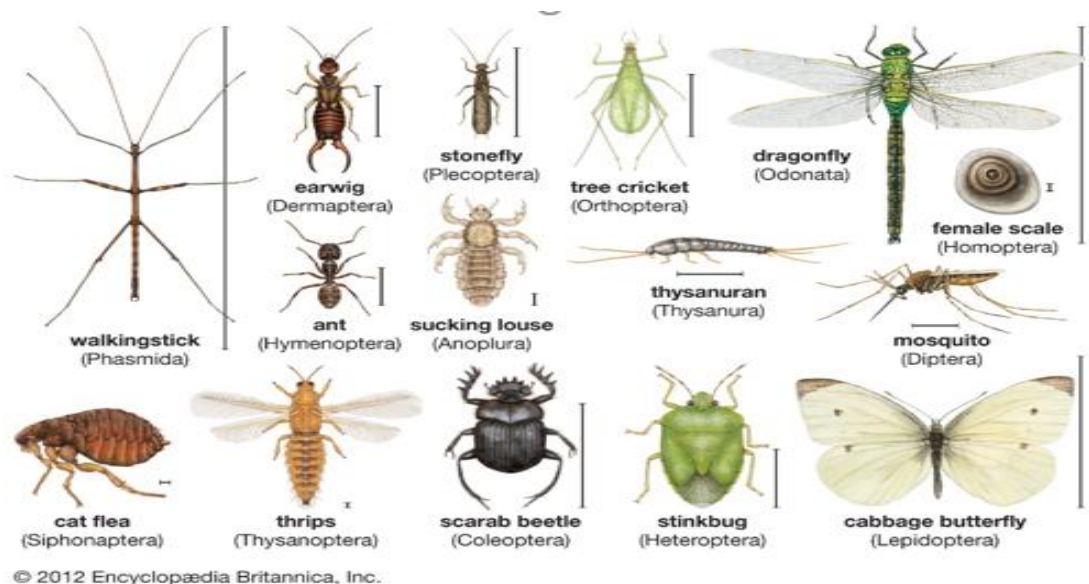
2.5.1 Giới thiệu về côn trùng

Côn trùng (*CON TRUNG HAI LƯA*, n.d.) là những động vật không xương sống có tên khoa học là insecta, là lớp lớn nhất trên Trái Đất và cung là lớp phân bố rộng rãi nhất trong số các đại diện của ngành chân khớp. Côn trùng là nhóm đa dạng nhất trên Trái Đất, với hơn 1 triệu loài đã được mô tả chiếm hơn một nửa tổng số tất cả các loài sinh vật sống mà con người biết đến với ước lượng về số loài chưa được mô tả lên tới 30 triệu, và do đó có thể đại diện cho hơn 90% các dạng sống khác nhau trên hành tinh

Côn trùng thật sự có các đặc điểm sau:

- Thứ nhất cơ thể của một côn trùng trưởng thành phải phân thành 3 phần đầu, ngực, bụng.
- Thứ hai, thành trùng phải có tất cả 3 đôi chân phải gắn vào các đốt ngực, hai đôi râu trên đầu và phần bụng được phân chia thành nhiều đốt (nhỏ hơn bằng 11 đốt).

Phần lớn côn trùng trưởng thành đều có cánh.



Hình 14: Hình ảnh về đa dạng côn trùng

2.5.2 Hình thái phát triển

Kích thước côn trùng dao động khoảng từ trên dưới 1mm tới khoảng 180mm về chiều dài. Côn trùng có cơ thể phân đốt và được bảo vệ bởi một bộ xương ngoài, một lớp cứng được cấu tạo chủ yếu bởi kitin.

Cơ thể có thể được chia thành đầu, ngực và bụng.

- Đầu có một cặp râu là cơ quan cảm giác, một cặp mắt kép và 2 mắt đơn (ở giai đoạn non có thể là 6 mắt đơn) và một miệng.

- Ngực có 6 chân (mỗi đốt một cặp chân) và 2-4 cánh (ở các loài có cánh).

- Bụng có cơ quan bài tiết và cơ quan sinh sản. Côn trùng có một hệ tiêu hóa hoàn chỉnh, gồm một ống liên tục từ miệng tới hậu môn, khác với nhiều động vật chân khớp đơn giản khác có hệ tiêu hóa chưa hoàn chỉnh. Cơ quan bài tiết gồm các ống Malpighi (Malpighian), với chức năng thải các chất thải chứa nitơ, ruột sau làm nhiệm vụ điều hoà áp suất thẩm thấu, đoạn cuối ruột sau có khả năng tái hấp thu nước cùng với muối Natri và Kali. Vì vậy, côn trùng thường không bài tiết nước ra cùng với phân, thực tế thì chúng cho phép dự trữ nước trong cơ thể. Quá trình tái hấp thu này giúp chúng có thể chịu đựng được với điều kiện môi trường khô và nóng.

- Hầu hết côn trùng có hai cặp cánh liên kết với đốt ngực 2 và 3. Côn trùng là động vật không xương sống duy nhất đã tiến hoá theo hướng bay lượn và chính điều này đóng một vai trò quan trọng trong sự thành công của chúng. Các côn trùng có cánh, và những côn trùng không cánh thứ sinh đã tạo nên nhóm có cánh (Pterygota). Cơ chế bay của côn trùng cho đến nay vẫn chưa được tìm hiểu một cách đầy đủ, người ta cho rằng nó phụ thuộc rất lớn vào khối không khí nhiễu loạn do cánh tạo ra. Ở những côn trùng nguyên thủy lại dựa chủ yếu vào tác động của hệ cơ lên cánh và cấu trúc của cánh. Ở những bộ tiến hoá hơn như Neoptera, cánh thường gập lại trên lưng khi chúng nghỉ ngơi. Ở những côn trùng này, cánh được hoạt động bởi các cơ bay gián tiếp mà giúp cánh vận động bằng cách ép mạnh lên thành ngực. Những cơ này có thể co lại khi bị căng ra mà không cần sự điều khiển của hệ thần kinh, điều này cho phép chúng tạo ra tần số co dẫn cơ tương đối cao.

- Hầu hết côn trùng có hai cặp cánh liên kết với đốt ngực 2 và 3. Côn trùng là động vật không xương sống duy nhất đã tiến hoá theo hướng bay lượn và chính điều này đóng một vai trò quan trọng trong sự thành công của chúng. Các côn trùng có cánh, và những côn trùng không cánh thứ sinh đã tạo nên nhóm có cánh (Pterygota). Cơ chế bay của côn trùng cho đến nay vẫn chưa được tìm hiểu một cách đầy đủ, người ta cho rằng nó phụ thuộc rất lớn vào khối không khí nhiễu loạn do cánh tạo ra. Ở những côn trùng nguyên thủy lại dựa chủ yếu vào tác động của hệ cơ lên cánh và cấu trúc của cánh. Ở những bộ tiến hoá hơn như Neoptera, cánh thường gập lại trên lưng khi chúng nghỉ ngơi. Ở những côn trùng này, cánh được hoạt động bởi các cơ bay gián tiếp mà giúp cánh vận động bằng cách ép mạnh lên thành ngực. Những cơ này có thể co lại khi bị căng ra

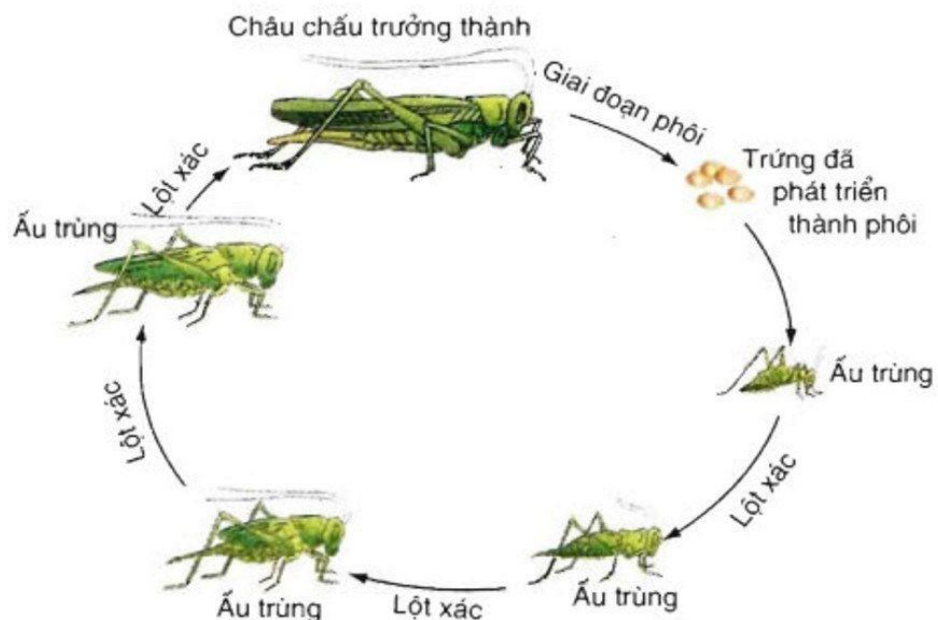
mà không cần sự điều khiển của hệ thần kinh, điều này cho phép chúng tạo ra tần số co giãn cơ tương đối cao.

2.5.3 Vòng đời của côn trùng

Côn trùng nở từ trứng, trải qua nhiều lần lột xác trước khi đạt tới kích thước trưởng thành của loài. Cách sinh trưởng này là bắt buộc vì chúng có bộ xương cứng bên ngoài, được cấu tạo chủ yếu bởi kitin (chitin). Lột xác là quá trình mà con vật thoát khỏi lớp xương ngoài cũ để tăng lên về kích thước, sau đó hình thành nên bộ xương ngoài mới, vì lớp xương ngoài bằng kitin hoặc đá vôi của các loài chân khớp không thể tăng lên về kích cỡ, trong khi cơ thể của chúng luôn luôn lớn lên cho tới lúc trưởng thành. Ở hầu hết các loài côn trùng, giai đoạn trẻ được gọi là ấu trùng (nymph). Ấu trùng có thể có cấu tạo tương tự như Thành trùng như ở châu chấu (mặc dù cánh vẫn chưa chỉ phát triển đầy đủ cho đến giai đoạn trưởng thành). Đây là những côn trùng biến thái không hoàn toàn.

Côn trùng có hai kiểu biến thái đó là kiểu biến thái không hoàn toàn (không trải qua giai đoạn nhộng) và kiểu biến thái hoàn toàn (có trải qua giai đoạn nhộng).

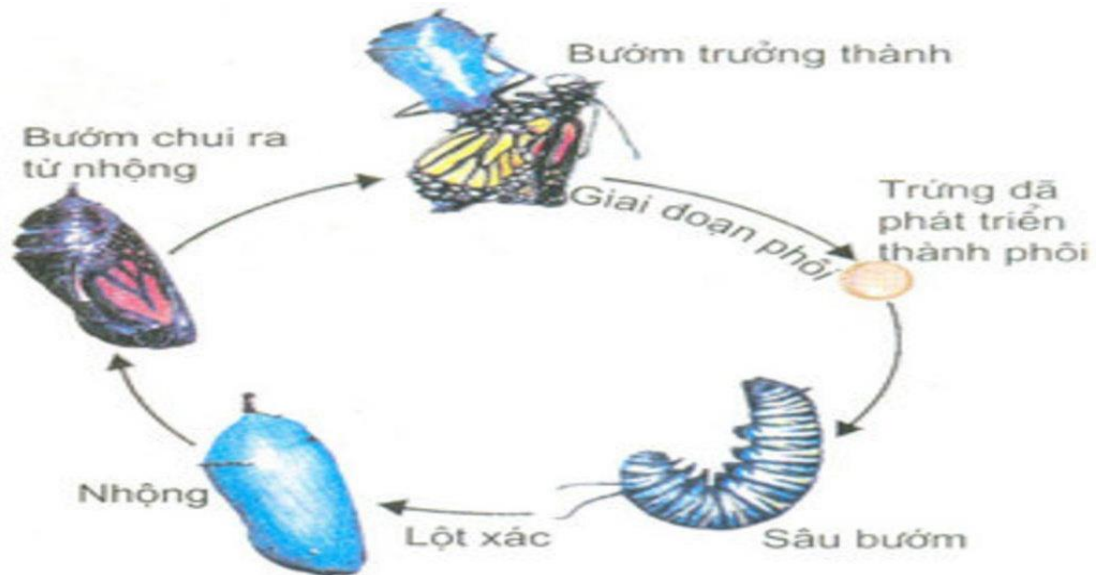
Vòng đời biến thái không hoàn toàn của côn trùng gồm có 3 pha sinh học: Trứng - Ấu trùng - Trưởng thành.



Hình 15: Vòng đời biến thái không hoàn toàn của Châu Chấu

Vòng đời biến thái hoàn toàn của côn trùng gồm 4 pha sinh học: Trứng - Ấu trùng - Nhộng - Bướm.

Ở những côn trùng biến thái hoàn toàn, trứng nở thành dạng ấu trùng, có dạng giống như giun đất, gọi là giai đoạn sâu non, Ấu trùng phát triển và cuối cùng biến thái thành nhộng (giai đoạn bao bọc trong kén) ở một số loài. Ở trạng thái kén, chúng trải qua những thay đổi đáng kể về hình dạng và cuối cùng chui ra khỏi kén như một con trưởng thành hay còn gọi là hóa vụ



Hình 16: Vòng đời biến thái hoàn toàn của Bướm

2.5.4 Tập tính của côn trùng

Nhiều loài côn trùng có các cơ quan cảm giác rất tinh tế. Trong một số trường hợp, các giác quan của chúng nhạy cảm hơn con người rất nhiều. Ví dụ, ong có thể nhìn được trong phổ bức xạ cực tím để tìm kiếm nơi hút mật là những bông hoa có bức xạ này để "dẫn đường" cho ong. Bướm đực có cái "mũi chuyên hóa" là đôi ăng ten (ở bướm ngày ăng ten có chóp tròn ở đầu mút và ở ngoài (bướm đêm) lại có dạng lông vũ hoặc không có đầu mút tròn) có thể ngửi thấy pheromone của bướm cái từ khoảng cách vài km.

Các côn trùng có tập tính xã hội như kiến hay ong, chúng sống cùng nhau trong một tập đoàn lớn và được tổ chức rất tốt. Các cá thể trong tập đoàn tương đối giống nhau về bộ gen (do trinh sản) nên người ta có thể coi cả tập đoàn như một "siêu cơ thể". Đứng đầu một thị tộc côn trùng như vậy là con chúa-con cái duy nhất có khả năng sinh sản, và chỉ đảm nhiệm chức vụ này trong bấy và là mẹ của mọi con côn trùng khác trong thị tộc, bao gồm những con thợ là những con cái không có khả năng sinh sản, thực hiện mọi nhiệm vụ của tổ, từ kiếm thức ăn, vệ sinh tổ và vệ sinh con chúa, chăm sóc ấu trùng... Con chúa điều khiển lũ con của mình bằng pheromon, và cứ vào mỗi mùa sinh sản mới, chúng lại cho ra đời một lứa con chúa là hậu duệ của mình, khi trưởng thành những con này sẽ bay đi để tạo nên một thị tộc riêng, những đàn kiến cánh bay vào nhà

bạn chính là hình ảnh minh họa rõ nét của chúng. Còn những con thỏ thì được sinh ra hằng ngày với tốc độ chóng mặt. Còn những con đực chỉ đóng vai trò sinh sản.

Một tập tính quan trọng của côn trùng là một vài loài và ở một số giai đoạn biến thái chúng có thời kỳ ngủ đông (hibernate) và thời kỳ đình dục (diapause).

2.5.5 Giác quan của côn trùng

Một trong những lý do giúp côn trùng không ngừng tồn tại, tiến hóa và phát triển trong suốt hàng trăm triệu năm qua, thích ứng với mọi môi trường sống trên cạn chính là một hệ thống giác quan cực kì nhanh nhạy và chính xác mà tạo hóa trang bị cho chúng, được sử dụng trong mọi hoạt động di chuyển, tìm kiếm thức ăn, trốn tránh kẻ thù và sinh sản.

Thị giác của côn trùng thuộc hàng tốt nhất trong thế giới động vật. Và chúng lại có tới hai loại mắt: mắt kép và mắt đơn. Mỗi mắt kép của côn trùng được tạo nên bởi hàng trăm, hàng nghìn thấu kính nhỏ (là một tế bào thị giác) có kích thước hiển vi, mỗi thấu kính lại tiếp nhận một hình ảnh giống hệt nhau, điều đó có nghĩa là nếu bạn đứng trước một con ruồi, thì trong mắt nó, hình ảnh của khuôn mặt bạn sẽ được nhân lên hàng nghìn lần để hiển thị trên từng ngàn ấy thấu kính tí hon. Trong khi đó, mỗi mắt đơn chỉ được cấu tạo bởi một thấu kính như vậy, và chỉ có tác dụng cảm nhận sáng tối mà thôi.

Một số côn trùng có cả mắt đơn và mắt kép, trong khi những côn trùng khác chỉ có mắt đơn. Đặc biệt, mắt của côn trùng không chỉ nằm trên đầu. Các nhà khoa học đã thử bịt kín đầu của một con côn trùng, nhưng nó vẫn cảm nhận được vùng có ánh sáng nhờ những tế bào thị giác nằm rải rác trên cơ thể.

Không phải côn trùng nào cũng có thị giác tốt như nhau: Những côn trùng có lối sống săn mồi và ham thích bay lượn vào ban ngày như chuồn chuồn, ruồi, bọ ngựa, ong, bướm và bọ cánh cứng thường có thị giác rất tốt, bằng chứng là đôi mắt của chúng gần như bao trùm một nửa hay toàn bộ cái đầu. Những côn trùng khác ưa tối và hoạt động vào ban đêm (như gián), có cuộc sống chật chội dưới những hốc sâu trong lòng đất (như kiến và mối thì có thị giác kém hơn rất nhiều. Bù lại, con gián có đôi anten dài có vai trò xúc giác (chạm vào các vật thể xung quanh như chiếc gậy dò đường của người mù), vai trò khứu giác giúp chúng tìm ra chiếc bánh ngọt của bạn và có những lông xúc giác cực nhạy nhô ra từ đằng sau bụng có thể cảm nhận mọi rung động nhỏ nhất của không khí và mặt đất xung quanh giúp chúng biến mất ngay khi con người xuất hiện trong bếp. Mối là hậu duệ tiến hóa của gián, phần lớn chúng đều mù, và một số loài kiến, kẻ thù truyền kiếp của chúng cũng vậy. Nhưng chúng có hệ thống khứu

giác hết sức ưu việt và một tập thể trình sản được tổ chức một cách thông minh, giúp cả tập đoàn kiến thống nhất như một cơ thể trong mọi hoạt động sống thường ngày.

2.5.6 Phân loại côn trùng

Phân lớp: Apterygota (Không cánh) (“Côn trùng,” 2024)

- Bộ Archaeognatha (Hàm nguyên thủy)
- Bộ Thysanura (Đuôi tơ, Ba đuôi, Anh vĩ)
- Bộ Monura - (Độc vĩ, Một đuôi) *tuyệt chủng*

Phân lớp: Pterygota (Có cánh)

- Bộ Ephemeroptera (Phù du)
- Bộ Odonata (Chuồn chuồn)
- Bộ Diaphanopteroidea - *tuyệt chủng*
- Bộ Palaeodictyoptera - *tuyệt chủng*
- Bộ Megasecoptera - *tuyệt chủng*
- Bộ Archodonata - *tuyệt chủng*

Siêu bộ: Neoptera (Cánh mới)

- Bộ Blattodea (Gián)
- Isoptera (Bộ Đẳng cánh-Cánh Đều: Mối. Hiện có người xếp mối vào bộ gián-*Blattodea*)
- Mantodea (Bọ ngựa)
- Bộ Dermaptera (Cánh da)
- Bộ Plecoptera (Cánh úp)
- Bộ Orthoptera (Cánh thẳng: Châu chấu, cào cào, muỗim, dế)
- Bộ Phasmatodea (Bọ que)
- Bộ Embioptera (Cánh lợp, bọ chân dệt)
- Bộ Zoraptera (Rận đất)
- Bộ Grylloblattodea
- Bộ Mantophasmatodea (gladiators)

Siêu bộ: Exopterygota (Cánh ngoài)

- Bộ Psocoptera (Rệp sáp, Mọt)

- Bộ Thysanoptera (Cánh viền, Bộ trĩ)
- Bộ Phthiraptera (Rận, chấy)
- Bộ Hemiptera (Cánh nửa)

Siêu bộ: Endopterygota (Cánh trong)

- Raphidioptera (snakeflies)
- Megaloptera (Cánh rộng)
- Neuroptera (Cánh gân: Tảo linh)
- Coleoptera (Cánh cứng: Bộ rùa, Bộ hung)
- Strepsiptera (Cánh vuốt)
- Mecoptera (Cánh dài)
- Siphonaptera (Cánh ống: Bộ chét)
- Diptera (Cánh đôi-Hai cánh: Ruồi, Muỗi)
- Trichoptera (Cánh lông)
- Lepidoptera (Cánh vẩy, cánh phấn: bướm, ngài, nhậy)
- Hymenoptera (Cánh màng: Ong, kiến)
- Miomoptera - *tuyệt chủng*
- Proto Diptera (Hai cánh nguyên thủy) tuyệt chủng

2.5.7 Côn trùng có lợi và côn trùng gây hại

2.5.7.1 Côn trùng gây hại:

Chỉ có 0,1% các loài côn trùng là đi ngược lại lợi ích của con người. Nhiều côn trùng được coi là những con vật có hại với loài người vì chúng truyền bệnh (ruồi, muỗi), phá hại mùa màng như sâu, rầy, bọ xít, phá hủy các công trình (mối), hay làm hỏng các sản phẩm lương thực (mọt). Các nhà côn trùng học đã đưa ra nhiều biện pháp để kiểm soát chúng mà phổ biến nhất là thuốc trừ sâu. Tuy nhiên, ngày nay các phương pháp kiểm soát bằng sinh học (methods of biocontrol) đang ngày càng được dùng phổ biến hơn.

2.5.7.2 Côn trùng có lợi

Mặc dù các côn trùng có hại thường nhận được nhiều sự quan tâm hơn, bên cạnh đó vẫn có nhiều loài có lợi cho môi trường và con người. Một số loài thụ phấn cho các loài thực vật có hoa. Sự giao phấn (pollination) là sự trao đổi (hạt phấn) giữa các thực vật có hoa để sinh sản. Các loài côn trùng khi lấy mật và phấn hoa đã vô tình tiến hành giao phấn. Ngày nay, một loạt các vấn đề về môi trường đã làm giảm các quần thể "nhà giao phấn" (pollinator) này. Số lượng các loài côn trùng được nuôi với mục đích làm vật trung gian quản lý việc thụ phấn cho thực vật đang trong thời kỳ phát triển thịnh vượng.

Một số côn trùng cũng sinh ra những chất rất hữu ích như mật, sáp, tơ. Ong mật đã được con người nuôi từ hàng ngàn năm nay để lấy mật. Tơ tằm đã có ảnh hưởng rất lớn tới lịch sử loài người, các mối quan hệ thương mại được thiết lập trên con đường vận chuyển tơ lụa giữa Trung Quốc và phần còn lại của thế giới. Ấu trùng maggot được sử dụng để chữa trị vết thương, ngăn chặn sự hoại tử do chúng ăn các phần chết thối. Phương pháp điều trị hiện đại này đã được sử dụng ở một vài bệnh viện trên thế giới.

Nhiều nơi trên thế giới, côn trùng được sử dụng làm thức ăn cho con người (entomophagy) trong khi nó lại là đồ kiêng kỵ với vùng khác. Thực ra đây cũng là một nguồn protein trong dinh dưỡng của loài người. Người ta không thể ước tính được có bao nhiêu loài côn trùng đã nằm trong thực đơn của con người nhưng nó đã có mặt trong rất nhiều thức ăn, đặc biệt trong ngũ cốc. Hầu hết chúng ta không nhận ra rằng các luật bảo vệ thực phẩm ở nhiều nước không ngăn cản việc có mặt của côn trùng trong thức ăn.

Nhiều côn trùng, đặc biệt là các loài cánh cứng là những bọ ăn xác thối, chúng ăn các xác động vật chết, các cây bị gãy mục, trả lại môi trường các dạng hữu ích cho các sinh vật khác sử dụng. Ai Cập cổ đại đã sùng bái coi những con bọ cánh cứng như bọ hung là thần linh, bên cạnh nhiều động vật linh thiêng khác của họ như cá sấu, hà mã, cá trê, chim ưng... Điều này bắt nguồn từ một sự quan sát gắn với truyền thuyết: những con bọ hung Ai Cập sử dụng phân động vật làm thức ăn cho những con non của nó. Mà với một số lượng bọ hung đông đúc hoàn toàn sống dựa vào những bãi phân thì đối với chúng, thứ thức ăn bốc mùi này quả thật quý như vàng, và vì thế mà tranh chấp xảy ra. Chúng phải tìm cách lẫn cục phân đi càng nhanh càng tốt khỏi đồng phân và tìm một nơi chôn "kho báu" để giữ cho nó không bị cướp lại bởi những bà mẹ côn trùng khác. Chúng sử dụng hai chân sau để lẫn phân-điều này đồng nghĩa với việc phải lộn ngược thân mình trong tư thế tròng cây chuối, mà như vậy thì không tiện cho việc quan sát đường đi cho lắm. Bởi vậy, những con bọ hung sử dụng hướng di chuyển của Mặt Trời, tức là từ Đông sang Tây làm la bàn định vị, những ông chủ kim tự tháp nhìn thấy các viên phân tròn dịch chuyển theo hướng di chuyển của Mặt Trời, rồi lại biến mất xuống lòng đất (bọ hung chôn phân trước khi đẻ trứng lên đó) đã ví những hình tượng không lấy gì làm vệ sinh lắm ấy với thần Mặt Trời, thần linh tối cao của họ. Và để trả ơn cho công lao dọn vệ sinh của con bọ hung, người Ai Cập đã trao cho chúng cái chức danh "người dẫn đường cho thần Mặt Trời".

Nhiều loài côn trùng là động vật săn mồi, chúng ăn sâu, rầy trong tự nhiên hoặc ký sinh vào trứng, ấu trùng và nhộng của sâu là những thiên địch côn trùng trong tự nhiên, rất có ích lợi trong nông nghiệp nhờ chúng diệt bớt sâu, rầy.

Hầu hết chúng ta đều không ý thức được rằng, lợi ích lớn nhất của côn trùng chính là loài ăn côn trùng (insectivores). Nhiều loài côn trùng như châu chấu có thể sinh sản nhanh đến nỗi mà chúng có thể bao phủ Trái Đất chỉ trong một mùa sinh sản. Tuy nhiên có hàng trăm loài côn trùng khác ăn trứng của châu chấu, một số khác thì ăn cả những con trưởng thành. Vai trò này trong sinh thái thường được cho là của các loài chim, nhưng chính côn trùng, mặc dù không thực sự quyến rũ như những loài lông vũ kia mới chính là những con vật có vai trò quan trọng hơn. Với bất kỳ loài côn trùng có hại nào, như con người thường gọi, thì cũng có một loài ong bắp cày là vật ký sinh hay là thiên địch của chúng và giữ một vai trò quan trọng trong việc kiểm soát các loài có hại đó.

Sự quan tâm của với việc kiểm soát dịch hại bằng thuốc trừ sâu có thể có tác dụng phản lại, thực tế thì chúng ta đã không nhận ra rằng chính côn trùng đã tự kiểm soát lẫn nhau và cả các quần thể có hại. Vì vậy, kiểm soát bằng thuốc độc thậm chí có thể dẫn đến sự bùng phát một loạt dịch hại nào đó.

2.5 NHỮNG NGHIÊN CỨU TƯƠNG TỰ

2.5.1 Trong nước

Ứng dụng “Nhận diện sinh vật gây hại cho lúa” do cục BVTV đã phối hợp với Tổng Công ty Giải pháp Doanh nghiệp Viettel và Sở NNPTNT tỉnh An Giang xây dựng. Ứng dụng hỗ trợ nông dân trong việc nhận diện, chẩn đoán các vấn đề liên quan tới sinh vật gây hại cây trồng cũng như tư vấn sử dụng thuốc bảo vệ thực vật tiết kiệm, an toàn và hiệu quả. Ứng dụng cung cấp cho nông dân chức năng tự động nhận diện loài sinh vật gây hại qua hình ảnh, đồng thời có thể tra cứu thông tin sinh vật gây hại như đặc điểm hình thái, sinh học, và biện pháp phòng trừ. Ở ứng dụng này thì có điểm mạnh ở việc là nó hoạt động ở cả hai nền tảng là Android và IOS. Bên cạnh điểm mạnh thì nó còn hạn chế là việc phát hiện sinh vật gây hại chỉ thực hiện thông qua hình ảnh.

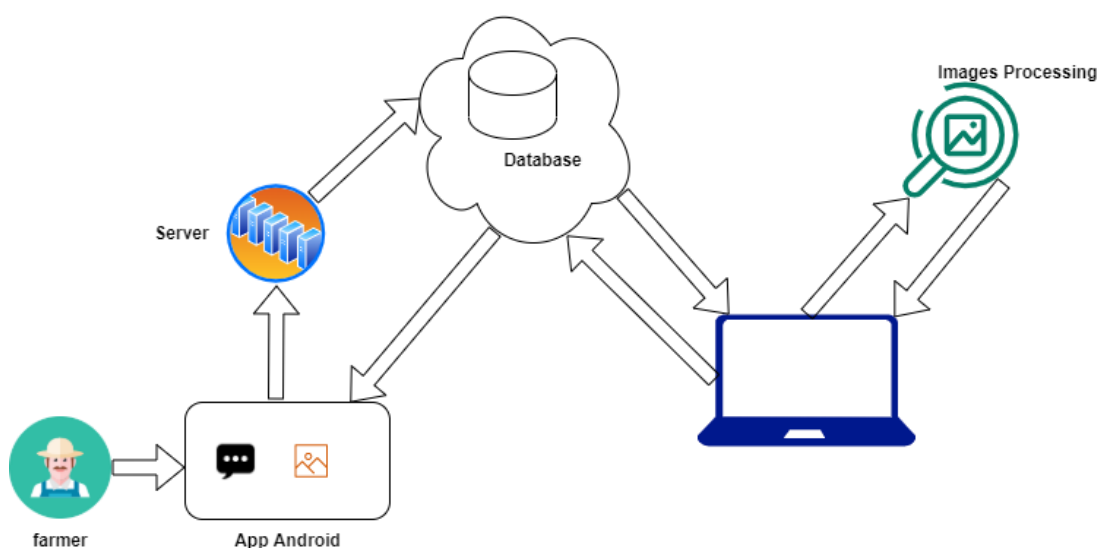
Ngày 8.4.2024, tại TP Quy Nhơn, Công viên sáng tạo TMA Bình Định (thuộc Công ty TNHH Giải pháp phần mềm Tường Minh) tổ chức hội thảo nghiên cứu ứng dụng trí tuệ nhân tạo trong nhận diện và cảnh báo một số sâu bệnh hại lúa trên địa bàn tỉnh (*Hệ Thống Giám Sát Côn Trùng Thông Minh - Cổng Thông Tin Điện Tử Cục Thông Tin, Thống Kê*, n.d.). Đây cũng là đề tài do đơn vị chủ trì; ông Trần Hoàn Anh Nguyên làm chủ nhiệm, thực hiện từ tháng 3.2022 đến 3.2024.

Tại hội thảo, ông Trần Hoàn Anh Nguyên báo cáo tóm tắt kết quả đạt được sau thời gian nghiên cứu. Cụ thể, đã hoàn thành khảo sát, thu thập, đánh giá và xây dựng cơ sở dữ liệu hiện trạng một số loại sâu, bệnh hại trên cây lúa; nghiên cứu các thuật toán xử lý hình ảnh, thị giác máy tính, hệ thống IoT, phần

cứng, mô hình phát hiện bệnh và côn trùng, hệ thống bản đồ thông tin địa lý; xây dựng mô hình nhận diện và cảnh báo; xây dựng thiết bị IoT (3 máy giám sát sâu, rầy); xây dựng phần mềm và phân tích, cải thiện giải pháp, triển khai cài đặt. Các đại biểu tham dự cho rằng, đề tài nghiên cứu có tính hữu ích; là công cụ giúp nông dân có thể tự nhận biết loài sinh vật gây hại trên đồng ruộng, từ đó giúp cơ quan quản lý nhà nước và người dân chủ động áp dụng các biện pháp phòng chống, nhằm bảo vệ sản xuất và đáp ứng các yêu cầu của thị trường nông sản...

2.5.2 Ngoài nước

Hệ thống nhận diện dịch hại thông minh của Sri Lanka (Theertha & Wanniarachchi, 2020). Sri Lanka là một quốc gia đang phát triển nông nghiệp, Nông nghiệp là sinh kế chính ở Sri Lanka. Nông nghiệp ở srilanka chia làm 2 mùa 64% diện tích đất được canh tác trong mùa khô và 35% đất trồng trong mùa mưa. Mặc dù việc trồng trọt ở tình trạng tốt 100% nhưng mọi người không thể thu được năng suất cao nhất. Tác nhân chính gây nên vấn đề trên là sự gây hại của côn trùng. Vì vậy việc cấp thiết là phải xác định được côn trùng gây hại và đó cũng là khó khăn lớn nhất trong nông nghiệp của Sri lanka là người dân không có nhiều kiến thức về việc nhận dạng những loài côn trùng gây hại. Vì vậy hệ thống nhận dạng dịch hại thông minh cho Sri Lanka trong trồng trọt là một bước ngoặt trong nông nghiệp Sri Lanka, bởi vì mỗi nông dân đều có điện thoại di động, nhưng họ chỉ sử dụng điện thoại đó cho mục đích liên lạc. Sri Lanka không có bất kỳ ứng dụng di động Android nào để phân tích xác định côn trùng. Hệ thống tập chung vào nhận dạng ảnh và xử lý. Hệ thống là một ứng dụng phân tán. Máy khách nằm ở thiết bị android.

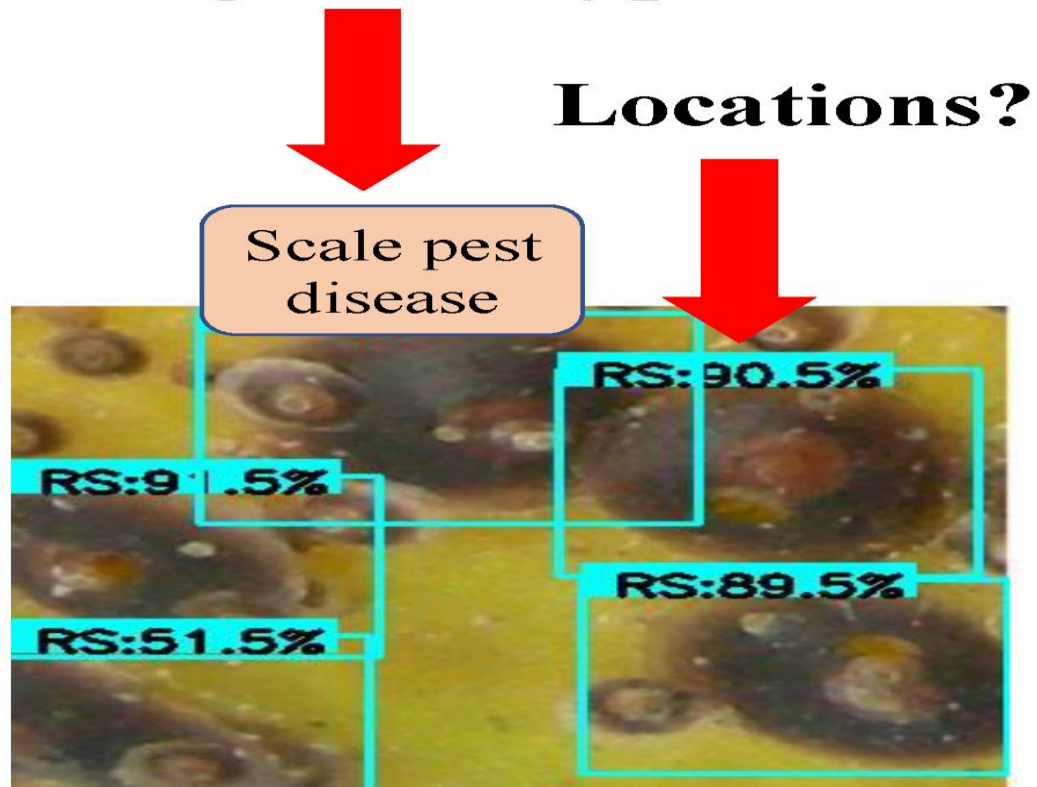


Hình 17: Hệ thống của ứng dụng nhận diện dịch hại của Sri Lanka

Để thực hiện theo quy trình, người nông dân chỉ cần tải côn trùng gây hại của mình lên ứng dụng di động và sau đó ứng dụng khách sẽ gửi hình ảnh vào máy chủ và bên trong máy chủ, hình ảnh sẽ được xử lý và gửi kết quả cho nông dân. Thông tin trả về của côn trùng sẽ là chiều cao, kích thước côn trùng, màu sắc côn trùng, từ đó đưa ra các bài thuốc để tiêu diệt côn trùng đó. Điều đạt được của ứng dụng là đã giải quyết được những vấn đề cần một hệ thống để xác định côn trùng gây hại, để từ đó có những phương pháp phòng chống cho phù hợp thay vì chỉ toàn sử dụng thuốc bảo vệ thực vật như trước kia, đồng thời còn cung cấp tư liệu cho bộ nông nghiệp. Bên cạnh những điều mà hệ thống đạt được thì còn một số hạn chế như người dân nếu muốn phân biệt côn trùng thì phải gửi ảnh hoặc tin nhắn miêu tả về côn trùng lên server và phải chờ kết quả trả về điều này sẽ làm cho việc nhận biết côn trùng trở nên tốn thời gian và chi phí hơn. Ứng dụng dựa trên điện thoại thông minh để phát hiện quy mô dịch hại bằng cách sử dụng phương pháp phát hiện nhiều đối tượng (A Smartphone-Based Application for Scale Pest Detection Using Multiple-Object Detection Methods) (Lee & Hwang, 2022). Mục đích của nghiên cứu này là thiết kế một hệ thống mà nông dân có thể sử dụng như một máy phát hiện dịch hại quy mô để ngăn ngừa sớm sự phá hại của dịch hại, hệ thống được phát triển được kết hợp với thiết bị điện thoại thông minh để giúp nông dân nâng cao hiệu quả của họ, hệ thống sử dụng nền tảng đám mây, đề xuất sử dụng nền tảng Keras để triển khai mô hình phát hiện đối tượng CNN trên thiết bị di động có thể phát hiện vị trí của ba loại sinh vật gây hại quy mô trong hình ảnh, Sử dụng các mô hình phát hiện đối tượng dựa trên học tập sâu để dự đoán các loại và vị trí của các loài gây hại quy mô xuất hiện trong hình ảnh, Sau khi đào tạo các mô hình phát hiện đối tượng, việc lưu các mô hình vào nền tảng đám mây của cơ sở dữ liệu. Bài báo này phát triển một hệ thống nhận dạng dịch hại thông minh dựa trên nền tảng đám mây, sử dụng công nghệ Internet of Things để tải mô hình được đào tạo lên nền tảng lưu trữ đám mây thông qua mạng Wifi hoặc mạng 4G. Để triển khai và suy luận mô hình được đào tạo trong môi trường thực địa, một thiết bị điện thoại thông minh đã được phát triển để kết hợp với nền tảng đám mây, có thể được người dùng sử dụng để xác định dịch hại theo thời gian thực. Hệ thống ứng dụng điện thoại thông minh để phát hiện quy mô dịch hại bằng cách sử dụng phương pháp nhiều đối tượng trên có ưu điểm là nó đảm bảo phát hiện đối tượng thời gian thực nhưng vẫn đảm bảo những yêu cầu đối với một ứng dụng thời gian thực là phát hiện đối tượng đáng tin cậy cao, đảm bảo thời gian dịch vụ phát hiện đối tượng, đảm bảo tốc độ khung hình, và đảm bảo thời gian phát hiện đối tượng (thời gian thực) (*YOLO with Adaptive Frame Control for Real-Time Object Detection Applications / Multimedia Tools and Applications*, n.d.). Bên cạnh ưu điểm thì hệ thống còn một khuyết điểm là mô

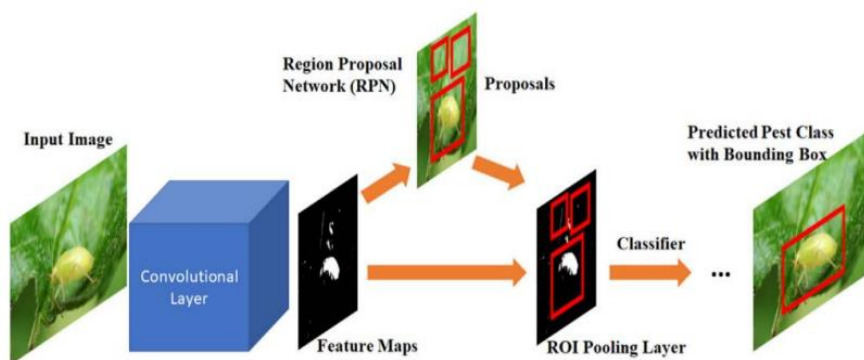
hình khi xây dựng xong thì mô hình sẽ được lưu trên đám mây, khi đó ứng dụng sẽ được kết nối với nền tảng đám mây, mô hình sẽ được truyền đến thiết bị di động để người dùng sử dụng để xác minh dịch hại theo thời gian thực.

Recognition types?



Hình 18: Xác định đúng vị trí của đối tượng.

Hay một nghiên cứu khác là ứng dụng di động mới về nhận dạng dịch hại nông nghiệp bằng cách sử dụng học sâu trong hệ thống điện toán đám mây (*A New Mobile Application of Agricultural Pests Recognition Using Deep Learning in Cloud Computing System - ScienceDirect*, n.d.) sử dụng Faster R-CNN (Ren et al., 2015). Faster R-CNN là một CNN học sâu thống nhất để phát hiện và xác định mục tiêu trong hình ảnh, bao gồm phát triển tính năng, tạo vùng ứng cử viên, phân loại ảnh vùng và tinh chỉnh vị trí. Các lớp chấp sử dụng mô hình InceptionV2 (Szegedy et al., 2016) được sử dụng trong nghiên cứu của tác giả như một chiếc feature cho Faster R-CNN vì ưu điểm của nó để cân bằng giữa tốc độ xử lý và độ chính xác trong hệ thống phát hiện dựa trên học sâu của modern. Mạng lưới InceptionV2 được đào tạo trên tập dữ liệu COCO (Lin et al., 2014) độ phân giải ảnh tối thiểu là 600pixel và tối đa là 1024 pixel, kích thước ảnh là 224 x 224pixel và có thể thay đổi tối đa là [600, 1024].



Hình 19: Quy trình công việc cơ bản của Faster R-CNN để phát hiện và phân loại lớp đối tượng mục tiêu, tức là các loài gây hại cây trồng (trong nghiên cứu của tác giả).

Bảng 4: Hiệu suất của phương pháp Faster R-CNN với các phương pháp khác trong tài liệu.

Method	Dataset	Number of Pest Class	Accuracy(%)
Bio-inspired methods	IP102 dataset	10	92.4%
Deep residual models	Cotton pests	2	98.0%
Transfer learning models	Soybean pests	13	93.8%
Machine learning	IP102 datasets	9	91.5%
techniques	IP102 datasets	24	90.0%
Our proposed Faster R-CNN	IP102 datasets	5	98.9%

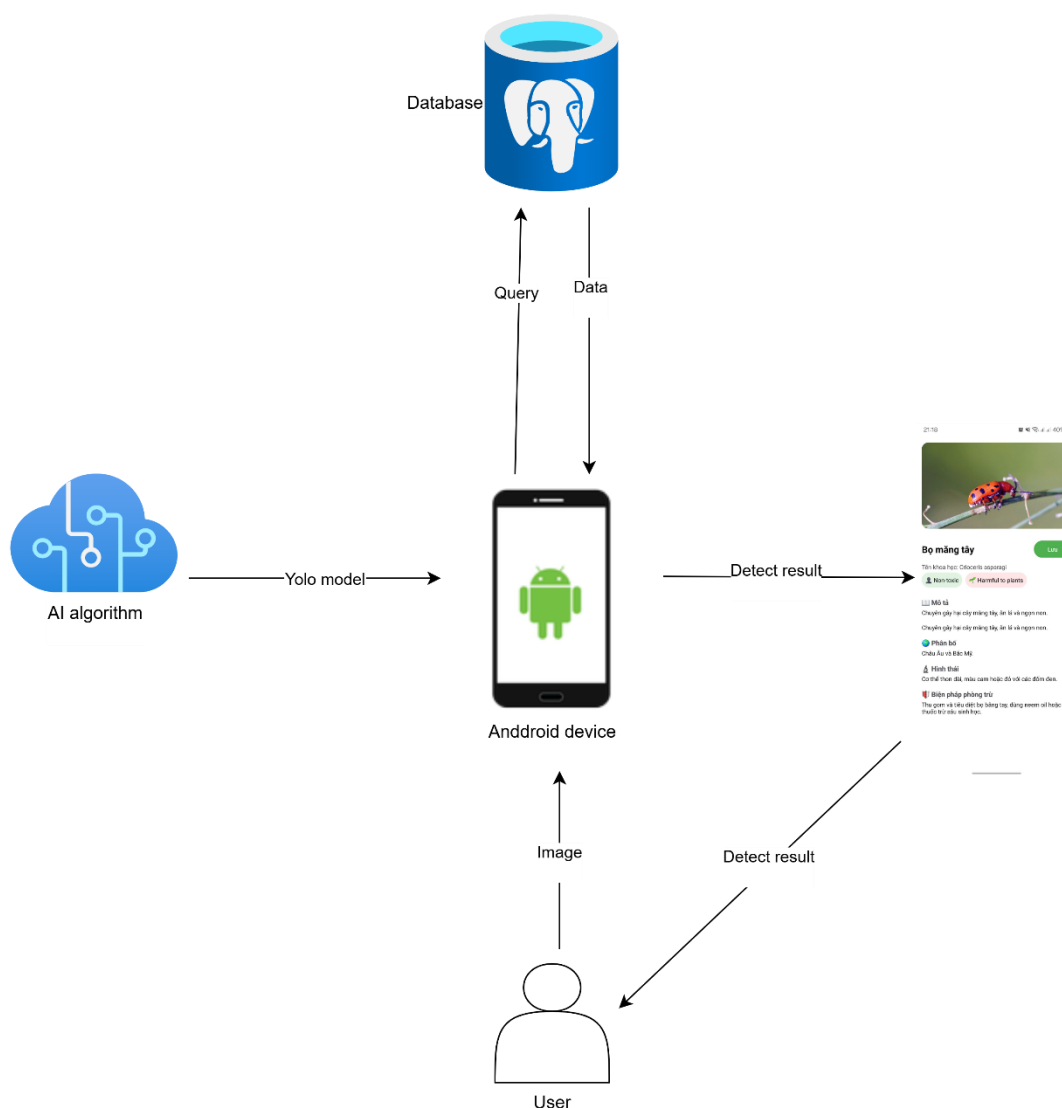
Từ bảng đánh giá trên thì chúng ta có thể thấy được là ứng dụng sử dụng mô hình Faster R-CNN thu về kết quả độ chính xác cao (khoảng 98.9%). Nhưng mô hình ứng dụng vẫn còn hạn chế ở một số điểm như sau thứ nhất thứ nhất hệ thống chỉ mới nhận diện côn trùng thông qua hình ảnh chưa thể nhận diện trực tiếp (nhận diện thời gian thực) vì vậy độ tối ưu của ứng dụng sẽ không được cao, ngoài ra mô hình xây dựng để nhận dạng cũng được lưu trữ trên đám mây, điều này sẽ đòi hỏi người dùng khi muốn dùng ứng dụng thì phải có kết nối với internet, làm cho ứng dụng không thể thích ứng với môi trường nhận diện. Từ những nghiên cứu ở trên tuy các nghiên cứu cho kết quả tương đối tốt nhưng

vẫn còn những hạn chế điều đó làm cho hệ thống chưa trở nên thuận tiện, rút kinh nghiệm từ những hệ thống trước đó, thì ở hệ thống của tôi xây dựng cũng có những cải tiến hơn. Vì hệ thống nhận diện côn trùng thời gian thực được tích hợp trên ứng dụng di động nên nó có những đòi hỏi riêng như là mô hình phải nhẹ, bảo đảm phát hiện đối tượng đáng tin cậy cao, đảm bảo thời gian dịch vụ phát hiện, đảm bảo khả năng thời gian thực. Để thích ứng với những tiêu chí đó thì chúng tôi sử dụng mô hình YOLOv5s, mô hình có kích thước không quá cao phù hợp để tích hợp vào ứng dụng di động, mô hình YOLO nếu xét về độ hiệu quả thì không phải tốt nhất, nhưng tốc độ thì nó là một thuật toán có tốc độ nhanh nhất trong lớp mô hình object detection, nó đạt được tốc độ gần như real time, đồng thời mô hình khi đào tạo xong thì sẽ được tích hợp và lưu vào trong ứng dụng di động, điều này giải quyết được vấn đề là ứng dụng có thể nhận diện được côn trùng khi không có Internet.

CHƯƠNG 3

HỆ THỐNG NHẬN DẠNG CÔN TRÙNG TRÊN THIẾT BỊ DI ĐỘNG

3.1 TỔNG QUAN HỆ THỐNG



Hình 20: Sơ đồ tổng quát hệ thống

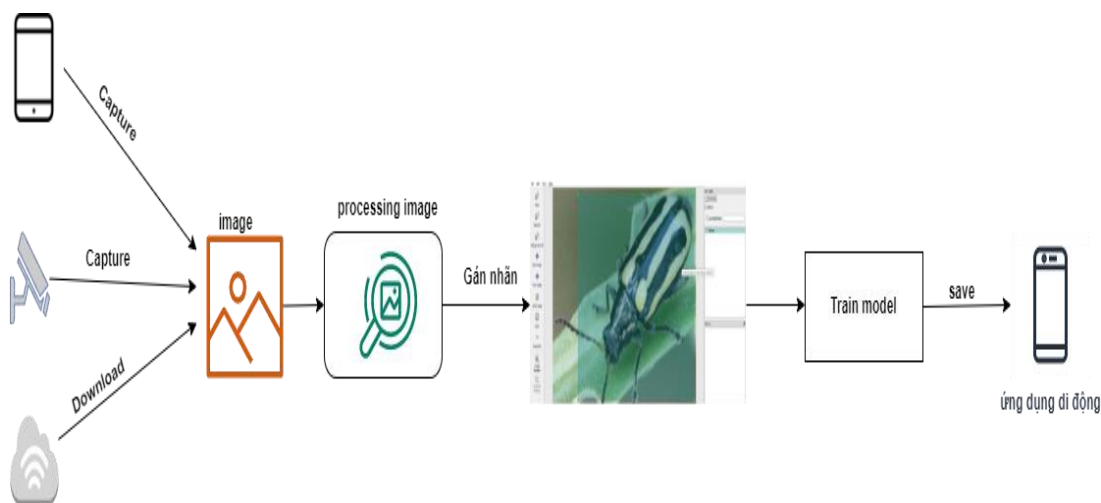
Trước hết, để có được một dữ liệu côn trùng để phục vụ cho việc nhận diện thì người dùng có thể tìm kiếm hình ảnh hoặc tên khoa học của loài côn trùng bạn muốn tìm hiểu trên google, Sử dụng camera điện thoại để ghi lại hình ảnh của côn trùng bạn gặp phải, hoặc là lấy từ những hệ thống bẫy côn trùng. Sau khi có được thông tin côn trùng, người dùng tiến hành sử dụng ứng dụng di động được xây dựng, ứng dụng có sử dụng mô hình học máy YOLOv9s được tích hợp sẵn để phân tích hình ảnh hoặc mô tả bằng văn bản về côn trùng một cách chính xác để tiến hành nhận diện côn trùng đó, khi nhận diện thành công chúng ta sẽ thu được tên của côn trùng và thông tin chi tiết về côn trùng, để có

được thông tin từ côn trùng thì ứng dụng sẽ truy vấn thông tin là tên về loài côn trùng vừa nhận diện xong trên database, sau đó database sẽ xử lý và trả về thông tin cho ứng dụng và hiển thị lên giao diện di động.

Việc ghi nhận côn trùng gây hại đóng vai trò rất quan trọng trong đa dạng sinh học và nông nghiệp, số lượng côn trùng gây hại đóng vai trò như chỉ số phản ánh mức độ đa dạng của hệ sinh thái. dữ liệu này giúp đánh giá tình trạng sức khỏe môi trường, theo dõi biến đổi hệ sinh thái theo thời gian và đánh giá tác động của hoạt động con người.

YOLO là một thuật toán Object detection nên mục tiêu của mô hình YOLO không chỉ là dự báo nhãn cho vật thể như các bài toán classification mà nó còn xác định vị trí của vật thể. Về độ chính xác thì có vẻ YOLO không phải là thuật toán có độ chính xác cao nhất nhưng về tốc độ thì YOLO là thuật toán nhanh nhất trong lớp mô hình Object detection.

Trong hệ thống này, không giống như các ứng dụng trước đây, ứng dụng của chúng tôi phát triển hướng đến việc ứng dụng có thể nhận diện côn trùng gây hại bằng hình ảnh hoặc thời gian thực giúp người dùng dễ dàng phát hiện côn trùng mọi lúc mọi nơi, nâng cao hiệu quả và tính tiện lợi. Hệ thống tích hợp trên thiết bị di động dễ dàng cài đặt và sử dụng trên mọi thiết bị di động thông minh, giúp bạn kiểm soát côn trùng mọi lúc mọi nơi. Vì tôi cần hướng đến là một hệ thống có thể phát hiện đối tượng thời gian thực nên tôi tiến hành tìm hiểu và sử dụng thuật toán YOLOv9s tiên tiến, ứng dụng hoạt động nhanh chóng và chính xác, ngay cả trên thiết bị di động có cấu hình thấp., tuy YOLO không phải là thuật toán tốt nhất nhưng nó là thuật toán nhanh nhất trong các lớp mô hình object detection, nó có thể đạt tốc độ gần như realtime mà độ chính xác không quá giảm so với các model thuộc top đầu.



Hình 21: Sơ đồ tổng quát quá trình huấn luyện mô hình và nơi lưu trữ mô hình

3.2 MÔ TẢ HỆ THỐNG

3.2.1 Danh sách Actor

Actor được hiểu là một thực thể tương tác với hệ thống để thực hiện một nhiệm vụ hoặc mục đích nào đó. Actor có thể là người dùng (user), một hệ thống khác, hoặc thậm chí là một phần của hệ thống.

Bảng 5: Danh sách Actor của hệ thống

STT	Tên Actor	Ý nghĩa
1	Admin	Người dùng có toàn quyền trong ứng dụng.
2	User	Người dùng bị hạn chế một số chức năng trong phần mềm

3.2.2 Danh sách Use case

Use Case là chức năng mà các Actor sẽ sử dụng.

Bảng 6: Danh sách các use case của hệ thống

STT	Tên Use-case	Chức năng
1	Đăng ký tài khoản	Đăng ký tài khoản người dùng.
2	Đăng nhập	Đăng Nhập.
3	Detect	Sử dụng model YOLO v9s được lưu trong máy thông qua camera để detect côn trùng.
4	Xem danh sách côn trùng	Xem danh sách côn trùng
5	Xem danh sách hình ảnh theo tên côn trùng	Xem thêm nhiều hình ảnh côn trùng detect được.
6	Xem thông tin côn trùng detect được	Hiển thị kết quả thông tin của côn trùng detect được từ ứng dụng.

Yêu cầu phi chức năng của hệ thống

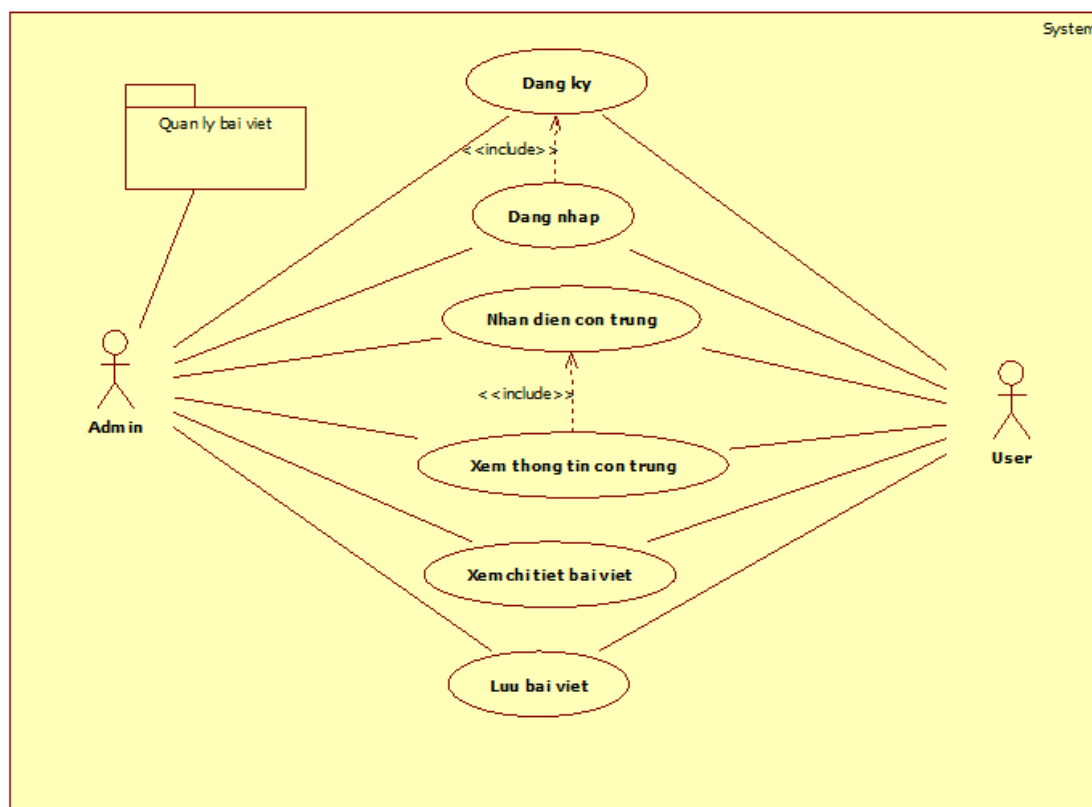
Ngoài các chức năng ở trên thì ứng dụng cũng có những yêu cầu phi chức năng khác như:

Giao diện đẹp, sinh động, thân thiện với người dùng.

Ứng dụng chạy ổn định, không có lỗi.

Tính bảo mật của ứng dụng cao.

3.2.3 Sơ đồ use case



Hình 22: Sơ đồ usecase

3.2.4 Đặc tả chi tiết use case

3.2.4.1 Đặc tả use case đăng nhập

Bảng 7: Đặc tả use case “Đăng Nhập”

Use Case: Đăng Nhập
Tác nhân chính: User, Admin
Mô tả ngắn gọn: - User hoặc Admin đăng nhập vào ứng dụng.
Điều kiện tiên quyết:

<ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường và tài khoản ở trạng thái mở.
Sự kiện kích hoạt: Khi người dùng khởi động phần mềm
Điều kiện thực hiện: Người dùng phải nhập chính xác tài khoản và mật khẩu.
Luồng sự kiện chính: <ol style="list-style-type: none"> 1. Khi khởi động ứng dụng lên người dùng chọn tiếp tục 2. Hệ thống sẽ chuyển qua giao diện đăng nhập, người dùng nhập username và password. 3. Hệ thống kiểm tra dữ liệu đầu vào có hợp lệ không. Nếu không thì sẽ chuyển đến luồng phụ A. 4. Hệ thống sẽ trả về kết quả và chuyển sang giao diện tiếp theo. 5. Kết thúc sự kiện.
Các luồng thay thế: <p>Luồng phụ A: Thông tin dữ liệu đầu vào không hợp lệ.</p> <ol style="list-style-type: none"> 1. Hệ thống báo lỗi cho người dùng, chỉ rõ nội dung không hợp lệ. 2. Hệ thống chuyển đến luồng sự kiện chính – bước 1.

3.2.4.2 Đặc tả use case đăng ký

Bảng 8: Đặc tả use case “Đăng Ký”

Use Case: Đăng Ký
Tác nhân chính: User, Admin
Mô tả ngắn gọn: <ul style="list-style-type: none"> - Người dùng đăng ký tài khoản.
Điều kiện tiên quyết: <ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường.
Sự kiện kích hoạt: Khi người dùng khởi động phần mềm, chọn chức năng đăng ký tài khoản.
Điều kiện thực hiện: Người dùng phải nhập đầy đủ các thông tin đăng ký tài khoản.

<p>Luồng sự kiện chính:</p> <ol style="list-style-type: none"> 1. Người dùng chọn nút “Đăng Ký” 2. Hệ thống sẽ chuyển sang giao diện đăng ký, và người dùng nhập thông tin 3. Hệ thống sẽ kiểm tra thông tin mà người dùng nhập vào. Nếu không đáp ứng yêu cầu thì sẽ chuyển sang luồng phụ A 4. Hệ thống sẽ trả về kết quả và chuyển sang giao diện đăng nhập 5. Kết thúc sự kiện.
<p>Các luồng thay thế:</p> <p>Luồng phụ A: Thông tin dữ liệu đầu vào không hợp lệ.</p> <ol style="list-style-type: none"> 1. Hệ thống báo lỗi cho người dùng, chỉ rõ nội dung không hợp lệ. 2. Hệ thống chuyển đến luồng sự kiện chính – bước 1.

3.2.4.3 Đặc tả use case quản lý users login

Bảng 9: Đặc tả use case “User Login”

Use Case: Đăng Nhập
Tác nhân chính: Admin
<p>Mô tả ngắn gọn:</p> <ul style="list-style-type: none"> - Admin đăng nhập vào ứng dụng.
<p>Điều kiện tiên quyết:</p> <ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường và tài khoản ở trạng thái mở.
Sự kiện kích hoạt: Khi admin khởi động phần mềm
Điều kiện thực hiện: Admin chọn chức năng xem danh sách users LogIn
<p>Luồng sự kiện chính:</p> <ol style="list-style-type: none"> 6. Khi khởi động ứng dụng lên admin chọn tiếp tục 7. Hệ thống sẽ chuyển qua giao diện đăng nhập, Admin click chọn đăng nhập với account admin. 8. Admin chọn chức năng đăng bài viết. 9. Hệ thống lưu thông tin bài viết .

10. Kết thúc sự kiện.

3.2.4.4 Đặc tả use case detect

Bảng 10: Đặc tả use case “Nhận Diện”

Use Case: Detect real time
Tác nhân chính: Admin, User
Mô tả ngắn gọn: <ul style="list-style-type: none">- Người dùng nhận diện côn trùng.
Điều kiện tiên quyết: <ul style="list-style-type: none">- Hệ thống đang ở trạng thái hoạt động bình thường và người dùng đăng nhập thành công vào hệ thống.
Sự kiện kích hoạt: Khi người dùng khởi động phần mềm, đăng nhập thành công vào hệ thống.
Điều kiện thực hiện: Người dùng đưa camera vào côn trùng hoặc hình ảnh chứa côn trùng
Luồng sự kiện chính: <ol style="list-style-type: none">1. Khi đăng nhập thành công thì sẽ chuyển qua giao diện nhận diện.2. Hệ thống tiến hành nhận diện côn trùng thông qua camera. Nếu không detect được thì đến luồng phụ A.3. Hệ thống hiển thị kết quả và thông tin côn trùng.4. Kết thúc sự kiện.
Các luồng thay thế: <p>Luồng phụ A: Hệ thống không nhận dạng ra được côn trùng.</p> <ol style="list-style-type: none">1. Hệ thống thông báo cho người dùng không thể nhận dạng được côn trùng.2. Người dùng chọn vào chức năng detect để chuyển sang luồng sự kiện chính – bước 1.

3.2.4.5 Đặc tả use case detect loài mới

Bảng 11: Đặc tả use case “Nhận Diện Loài Mới”

Use Case: Detect real time

Tác nhân chính: Admin, User
Mô tả ngắn gọn: <ul style="list-style-type: none"> - Người dùng nhận diện côn trùng.
Điều kiện tiên quyết: <ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường và người dùng đăng nhập thành công vào hệ thống.
Sự kiện kích hoạt: Khi người dùng khởi động phần mềm, đăng nhập thành công vào hệ thống.
Điều kiện thực hiện: Người dùng đưa camera vào côn trùng hoặc hình ảnh chứa côn trùng
Luồng sự kiện chính: <ol style="list-style-type: none"> 5. Khi đăng nhập thành công thì sẽ chuyển qua giao diện nhận diện. 6. Hệ thống tiến hành nhận diện côn trùng thông qua camera. Nếu không detect được vì loài côn trùng mới 7. Hệ thống hiển thị kết quả không nhận diện được và cho phép người dùng nhấn nút chuyển sang màn hình mới. 8. Kết thúc sự kiện.

3.2.4.6 Đặc tả use case xem chi tiết côn trùng

Bảng 12: Đặc tả use case “Xem danh sách hình ảnh theo tên côn trùng”

Use Case: Xem danh sách hình ảnh theo tên côn trùng.
Tác nhân chính: User, Admin
Mô tả ngắn gọn: <ul style="list-style-type: none"> - Người dùng xem chi tiết thông tin về 1 loài côn trùng
Điều kiện tiên quyết: <ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường, đăng nhập thành công vào hệ thống.
Sự kiện kích hoạt: Người dùng click vào mục của listview
Điều kiện thực hiện: Người dùng click chọn 1 mục của listview

<p>Luồng sự kiện chính:</p> <ol style="list-style-type: none"> 1. Người dùng chọn chức năng “Xem chi tiết”. 2. Hệ thống nhận sự kiện click, nếu có lỗi thì chuyển sang luồng phụ A. 3. Hệ thống hiển thị chi tiết 1 loài côn trùng. 4. Kết thúc sự kiện.
<p>Các luồng thay thế:</p> <p>Luồng phụ A: Thông báo cho người dùng biết.</p> <ol style="list-style-type: none"> 3. Hệ thống báo lỗi cho người dùng, chỉ rõ nội dung không hợp lệ. 4. Hệ thống chuyển đến luồng sự kiện chính – bước 1.

3.2.4.7 Đặc tả use case đăng bài viết côn trùng

Bảng 13: Đặc tả use case đăng bài viết côn trùng

Use Case: Đăng bài viết
Tác nhân chính: Admin
<p>Mô tả ngắn gọn:</p> <ul style="list-style-type: none"> - Admin đăng bài viết về 1 loài côn trùng
<p>Điều kiện tiên quyết:</p> <ul style="list-style-type: none"> - Hệ thống đang ở trạng thái hoạt động bình thường và người dùng đăng nhập thành công vào hệ thống.
Sự kiện kích hoạt: Khi admin đăng nhập thành công chọn chức năng đăng bài viết
Điều kiện thực hiện: Admin điền đủ thông tin bài viết
<p>Luồng sự kiện chính:</p> <ol style="list-style-type: none"> 1. Admin chọn chức năng “Đăng bài viết”. 2. Người dùng chọn hình ảnh tại thư viện ảnh trên di động. 3. Hệ thống thông báo kết quả cho người dùng. 4. Kết thúc sự kiện.

3.3 THIẾT KẾ CƠ SỞ DỮ LIỆU

3.3.1 Thiết kế chi tiết các bảng dữ liệu

Bảng 14: Bảng thông tin côn trùng

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
id	int	Khóa chính	Mã côn trùng
Tenvn	varchar		Tên việt nam
Tenkh	varchar		Tên khoa học
Ddsinhhoc	varchar		Đặc điểm sinh học
Phanbo	varchar		Phân bố
Hinhthai	varchar		Hình thái
bienphapphongtru	varchar		Biện pháp phòng trừ
anh	varchar		Hình ảnh

Bảng 15: Bảng người dùng

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
ID	int	Khóa chính	Mã người dùng
Email	varchar		Họ và tên
Username	varchar		Tên đăng nhập
Password	varchar		Mật khẩu
ảnh	varchar		Email
vaitro	Varchar		Quyền hạn
Created_at	timestramptz		Khóa

Bảng 16: Bảng bài viết

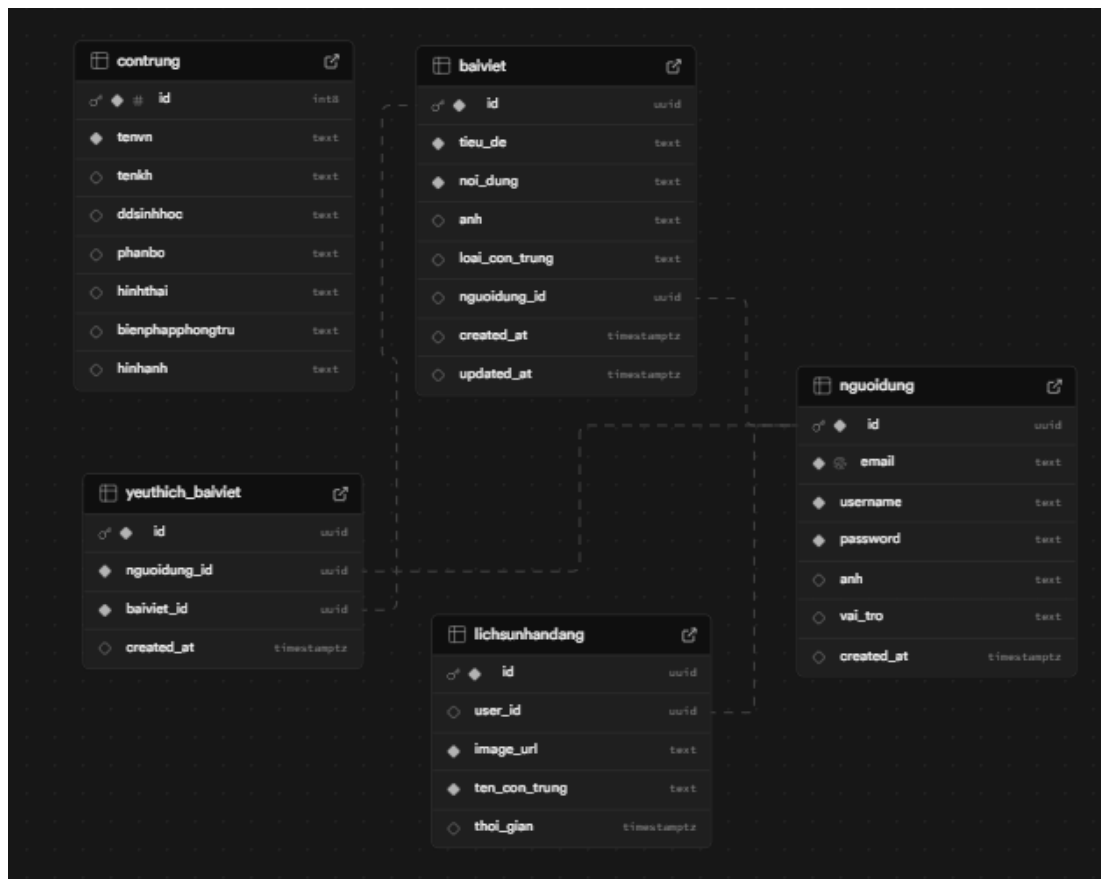
Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
id	int	Khóa chính	Mã bài viết

Tieu_de	varchar		Tiêu đề
Noi_dung	varchar		Nội dung
anh	varchar		Hình ảnh
Loai_con_trung	varchar		Loại côn trùng
Nguoidung_id	varchar		Mã người dùng
Created_at	Timestamptz		Ngày tạo
Update_at	Timestamptz		Ngày cập nhật

Bảng 17: Bảng bài viết yêu thích

Tên trường	Kiểu dữ liệu	Ràng buộc	Mô tả
id	int	Khóa chính	Id
Nguoidung_id	varchar		Mã người dùng
Baiviet_id	varchar		Nội dung
Created_at	Timestamptz		Hình ảnh
Loai_con_trung	varchar		Loại côn trùng
Nguoidung_id	varchar		Mã người dùng
Created_at	Timestamptz		Ngày tạo
Update_at	Timestamptz		Ngày cập nhật

3.3.2 Các lớp đối tượng








Hình 23: Hình ảnh lược đồ quan hệ cơ sở dữ liệu






3.4 THỰC NGHIỆM VÀ BÀN LUẬN

3.4.1 Mô tả tập dữ liệu của 10 lớp

Bảng 18: Côn trùng trong tập dữ liệu 10 lớp

STT	Tên	Ảnh
1	acalymma	

2	alticini	
3	Squash_Bug	
4	asparagus	
5	aulacophora	

6	dermaptera	
7	leptinotarsa	
8	mantodea	
9	Cerotoma_trifurcata	
10	Achatina_fulica	

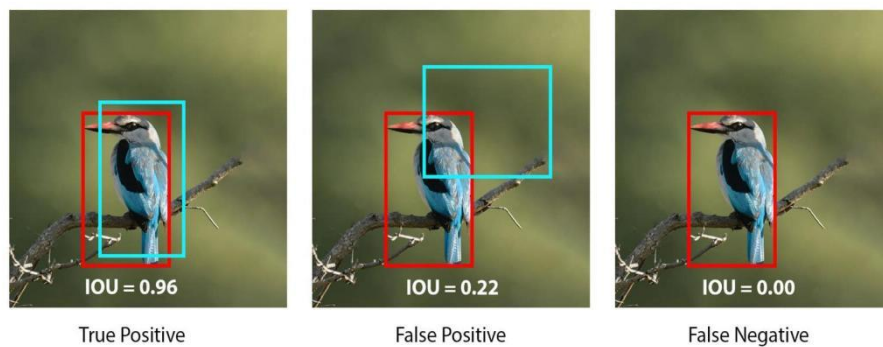
3.4.2 Độ chính xác khi nhận diện

Hiệu suất của các mô hình YOLO được đào tạo được đánh giá bằng cách kiểm tra trên một tập dữ liệu riêng biệt. Quá trình đánh giá sử dụng các chỉ số sau:

True Positive (TP): Số lượng đối tượng được phát hiện chính xác.

False Positive (FP): Số lượng đối tượng bị phát hiện sai.

False Negative (FN): Số lượng đối tượng mục tiêu bị bỏ sót trong quá trình phát hiện.



Hình 24: Độ chính xác khi nhận diện

Dựa trên các chỉ số TP, FP và FN, ta có thể tính toán các thông số sau để đánh giá hiệu quả mô hình:

Độ chính xác là một chỉ số đánh giá được tính toán với các kết quả tích cực đúng và sai. Tỷ lệ phần trăm các đối tượng được phát hiện chính xác so với tổng số đối tượng được phát hiện.

Mối quan hệ giữa độ chính xác và TP và FP có thể được biểu hiện như sau (Le et al., 2024):

$$Precision = \frac{TP}{TP + FP} = \frac{\text{Số lần dự đoán chính xác}}{\text{Tổng số lần dự đoán}}$$

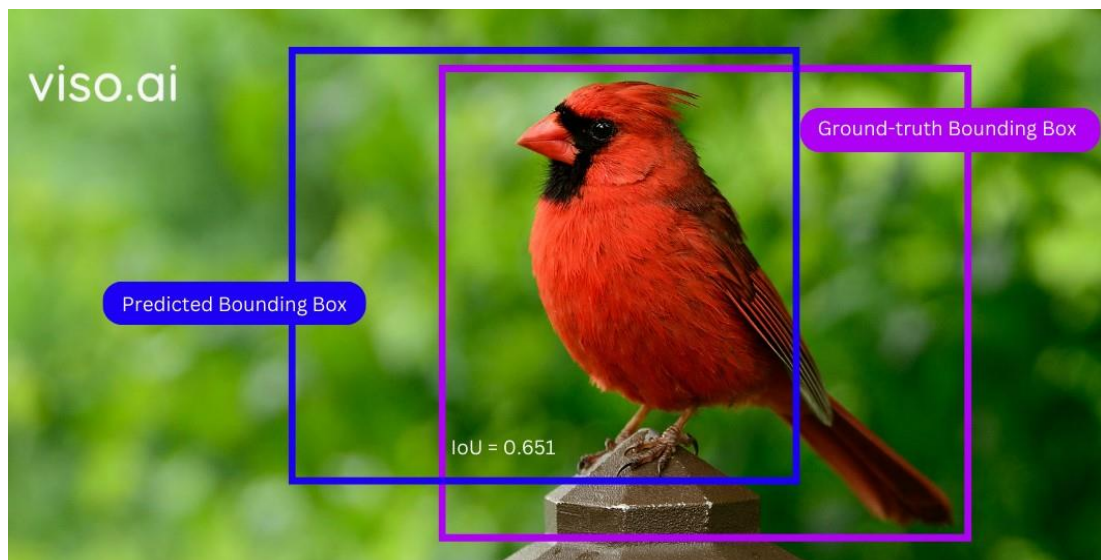
Recall đánh giá khả năng tìm kiếm toàn bộ các ground truth của mô hình (bao nhiêu % positive samples (TP) mà model nhận diện được). Dò bằng cách hiển thị phần trăm đối tượng được phát hiện chính xác trong tổng số các đối tượng thực (Le et al., 2024).

$$recall = \frac{TP}{TP + FN} = \frac{\text{Số lần dự đoán chính xác}}{\text{Số lần nhận dạng đúng có thể có}}$$

Tóm lại :Recall là chỉ số quan trọng để đánh giá khả năng "bao quát" của mô hình YOLO trong việc phát hiện vật thể. Giá trị Recall cao thể hiện mô hình ít bỏ sót đối tượng mục tiêu, mang lại kết quả phát hiện "hoàn chỉnh" hơn.

IOU (Intersection over Union) là chỉ số đánh giá được sử dụng để đo độ chính xác của phát hiện đối tượng trên tập dữ liệu cụ thể. Thường được dùng để đánh giá hiệu năng phát hiện đối tượng (*A Smartphone-Based Application for Scale Pest Detection Using Multiple-Object Detection Methods*, n.d.).

Trong đó, area of overlap là diện tích phần giao nhau giữa predicted bounding box (là đường bao mà chúng ta sử dụng file weight sau khi đào tạo để nhận dạng) với ground-truth bounding box (là đường bao mà chúng ta gán cho vật thể bằng labelImage tool) (*Tìm Hiểu về YOLO Trong Bài Toán Real-Time Object Detection*, n.d.).



Hình 25: Miêu tả area of overlap

$IOU > 0.5$ (True positive: TP): Đối tượng được nhận dạng đúng (Ren et al., 2015).

$IOU < 0.5$ (False positive: FP): Đối tượng nhận dạng sai.

Đối tượng không nhận diện được (False negative: FN).

3.4.3 Chế độ hoạt động của ứng dụng

Ứng dụng nhận diện côn trùng trên thiết bị di động được tôi phát triển để có thể hoạt động ở 2 chế độ là online và offline.

Ở chế độ online thì cơ sở dữ liệu về côn trùng sẽ được lưu ở database khi nhận diện thành công côn trùng thì ứng dụng di động sẽ truy vấn thông tin côn

trùng vừa nhận diện được trên database sau đó trả về thông tin loài côn trùng về ứng dụng để xem chi tiết côn trùng.

Chế độ hoạt động online mang lại ưu điểm là cơ sở dữ liệu được lưu trữ trên máy chủ, giúp giảm tải cho thiết bị di động và tạo điều kiện cho các thiết bị có tài nguyên hạn chế. Tuy nhiên, tốc độ của ứng dụng sẽ phụ thuộc vào tốc độ kết nối mạng.

Trong khi đó, chế độ offline sử dụng một cơ sở dữ liệu địa phương (SQLite) được tích hợp trên thiết bị di động. Dữ liệu côn trùng sẽ được lưu vào một tập tin văn bản trên thiết bị. Chế độ offline đặc biệt hữu ích trong ngành nông nghiệp, nơi mà việc sử dụng mạng có thể bị hạn chế. Ứng dụng có thể nhận diện côn trùng ngay cả khi không có kết nối internet, thích hợp cho việc sử dụng ngoài trời như trên đồng, ruộng hoặc trong rừng. Tuy nhiên, điều này cũng đồng nghĩa rằng các chức năng yêu cầu kết nối mạng của ứng dụng sẽ không hoạt động trong chế độ offline.

3.4.4 Độ chính xác khi xác định côn trùng trên tập dữ liệu hình ảnh

3.4.4.1 Nghiên cứu về tập dữ liệu 10 lớp

Để có được bảng tỷ lệ độ chính xác khi nhận dạng thì tôi tiến hành lấy những bức ảnh lấy những ảnh ngẫu nhiên ngoài tự nhiên của côn trùng có trong tập dữ liệu huấn luyện và tiến hành nhận diện thì cho kết quả chính xác tương đối cao, những hình ảnh của mỗi loài thì có những đặc trưng riêng về màu sắc và độ sáng, trong đó thấp nhất là Mantodea (47.7%).

Bảng 19: Bảng độ chính xác sau khi nhận diện của tập dữ liệu 10 lớp

STT	Tên côn trùng	Độ chính xác khi nhận diện
1	Acalymma vittatum	42.3%
2	Achatina fulica	69.7%
3	Alticini	83.3%
4	Squash bug	65.3%
5	Asparagus beetles	63.4%
6	Aulacophora similis	74.4%
7	Cerotoma trifurcata	86.2%
8	Dermaptera	62.9%

9	Leptinotarsa decemlineata	96.8%
10	Mantodea	47.7%
	Độ chính xác trung bình của 10 lớp	69.5%

Từ bảng trên ta thấy ứng di động của chúng tôi phát triển có thể nhận diện tương đối chính xác đối với các loại côn trùng. Ứng dụng được phát triển dựa trên model YOLOv9s được huấn luyện trên tập dữ liệu 10 lớp, để nhận diện côn trùng so với các phương pháp nhận diện khác thì ở YOLO cho tốc độ nhận diện cao hơn gần như là cho phép nhận diện với thời gian thực

3.4.4.2 Chỉ số FPS trong nhận diện

Chỉ số FPS (Frames Per Second) là một tiêu chí quan trọng phản ánh khả năng xử lý thời gian thực của mô hình nhận diện đối tượng. Trong các ứng dụng như giám sát video, robot, và hệ thống cảnh báo thông minh, FPS càng cao càng giúp hệ thống phản hồi nhanh và chính xác hơn. Dưới đây là bảng so sánh chỉ số FPS của YOLOv9s với các phiên bản cùng cấp chạy trên cùng một môi trường phần cứng Tesla P100:

Bảng 20: So sánh các phiên bản YOLOv9

Phiên bản YOLO	FPS (Tesla P100)	Kích thước mô hình (MB)	Độ chính xác (mAP@0.5)
YOLOv9s	170	~25	~53%
YOLOv9m	120	~60	~57%
YOLOv9l	85	~90	~60%
YOLOv9x	60	~140	~62%

Phân tích:

YOLOv9s đạt tốc độ nhanh nhất với 170 FPS, rất lý tưởng cho các ứng dụng yêu cầu phản hồi thời gian thực như:

Giám sát video trực tiếp

Nhận diện trong robot di động

Thiết bị biên (Edge devices) và IoT

Khi tăng dần kích thước mô hình từ YOLOv9m \rightarrow YOLOv9l \rightarrow YOLOv9x, tốc độ suy luận giảm dần nhưng độ chính xác tăng lên đáng kể. Điều này thể hiện rõ chiến lược đánh đổi (trade-off) giữa tốc độ và hiệu suất.

YOLOv9x tuy có độ chính xác cao nhất (~62%) nhưng chỉ đạt 60 FPS, do độ phức tạp của mạng và số lượng tham số lớn hơn.

Như vậy, YOLOv9s là lựa chọn tối ưu nhất khi cần tốc độ xử lý cao, còn YOLOv9l hoặc YOLOv9x sẽ phù hợp hơn với các tác vụ yêu cầu độ chính xác cao, chấp nhận suy luận chậm hơn.

Kết luận:

Trong dòng YOLOv9, mỗi phiên bản đóng vai trò chiến lược riêng tùy theo yêu cầu thực tế. Tuy nhiên, nếu xét riêng về tốc độ nhận diện (chỉ số FPS), YOLOv9s vượt trội hơn hẳn các phiên bản cùng dòng, và là mô hình lý tưởng để triển khai trên các hệ thống thời gian thực với yêu cầu tốc độ cao và tài nguyên tính toán hạn chế.

3.5 GIAO DIỆN CHƯƠNG TRÌNH

3.5.1 Giao diện màn hình đăng nhập

17:58 49%

Đăng nhập

cuong@gmail.com

.....

☒ Lưu đăng nhập

ĐĂNG NHẬP

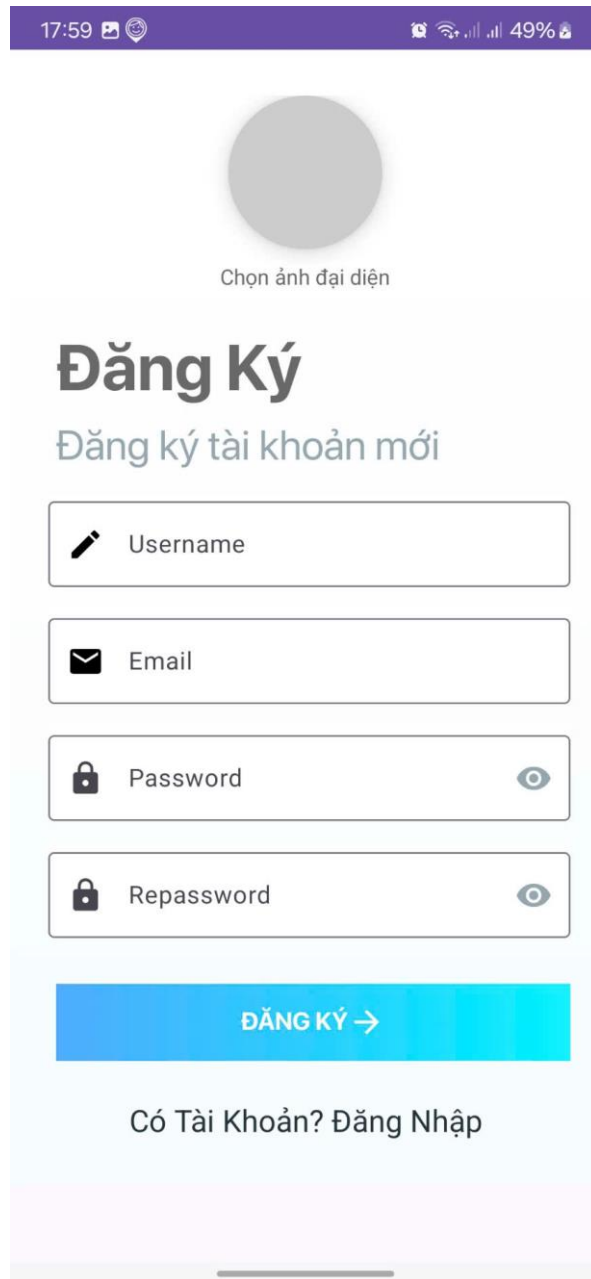
Đăng ký tài khoản

Quên Mật Khẩu?

Đăng nhập

Hình 26: Giao diện đăng nhập

3.5.2 Giao diện màn hình đăng ký




The image shows a mobile application registration screen. At the top, there is a status bar with the time 17:59, signal strength, and 49% battery. Below the status bar is a large gray circle representing a profile picture, with the text "Chọn ảnh đại diện" (Choose profile picture) underneath it. The main heading is "Đăng Ký" (Register) in a large, bold font, followed by the subtitle "Đăng ký tài khoản mới" (Register new account) in a smaller, gray font. There are four input fields: "Username" with a pencil icon, "Email" with an envelope icon, "Password" with a lock icon and a toggle eye icon, and "Repassword" with a lock icon and a toggle eye icon. Below these fields is a large blue button with the text "ĐĂNG KÝ →" (Register →). Under the button, there is a link that says "Có Tài Khoản? Đăng Nhập" (Have an account? Log in). The bottom of the screen shows a white home indicator bar.


17:59 49%



Chọn ảnh đại diện



Đăng Ký

Đăng ký tài khoản mới

 Username

 Email

 Password 

 Repassword 

ĐĂNG KÝ →

Có Tài Khoản? Đăng Nhập

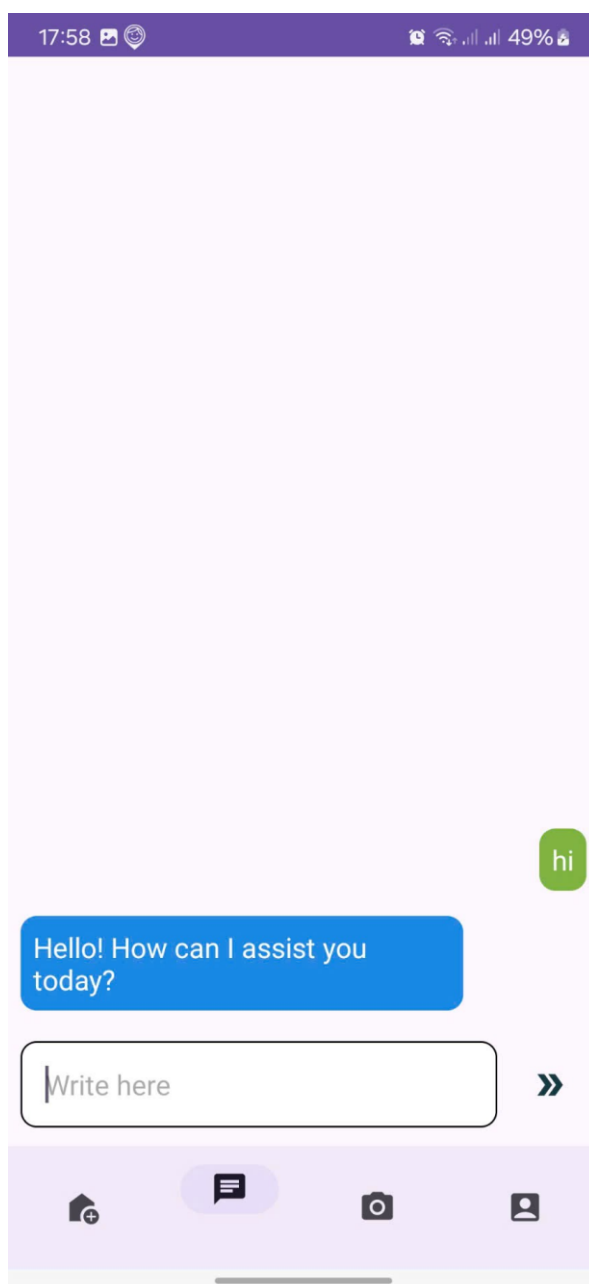
Hình 27: Giao diện đăng ký

3.5.3 Giao diện home



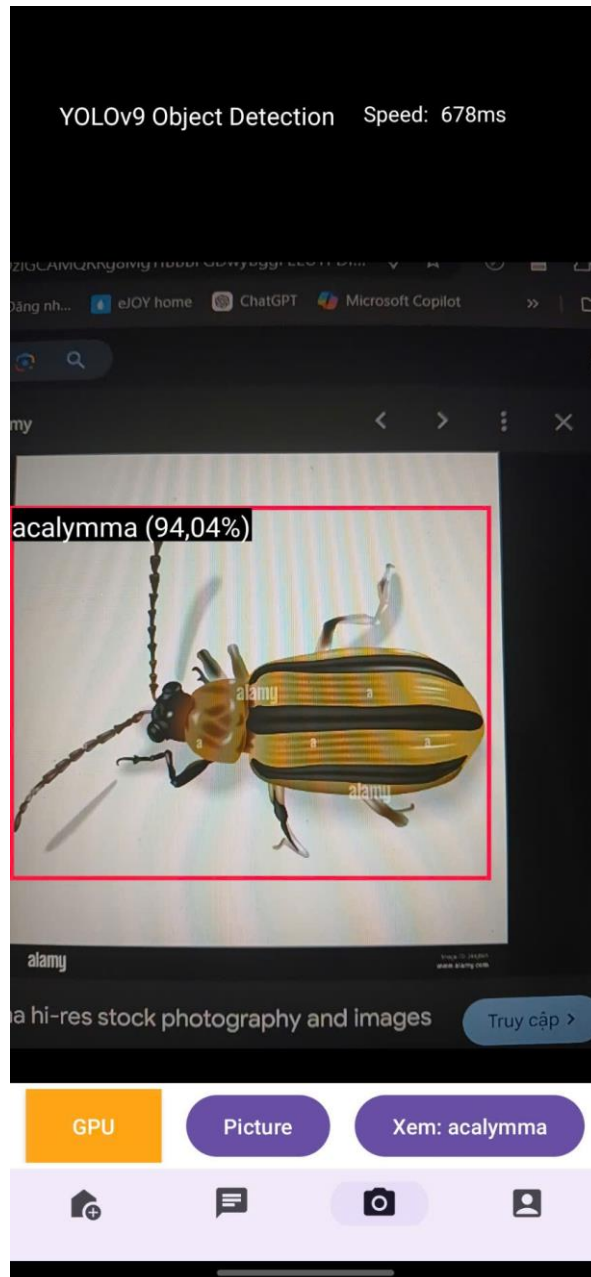
Hình 28: Giao diện home

3.5.4 Giao diện chatbot



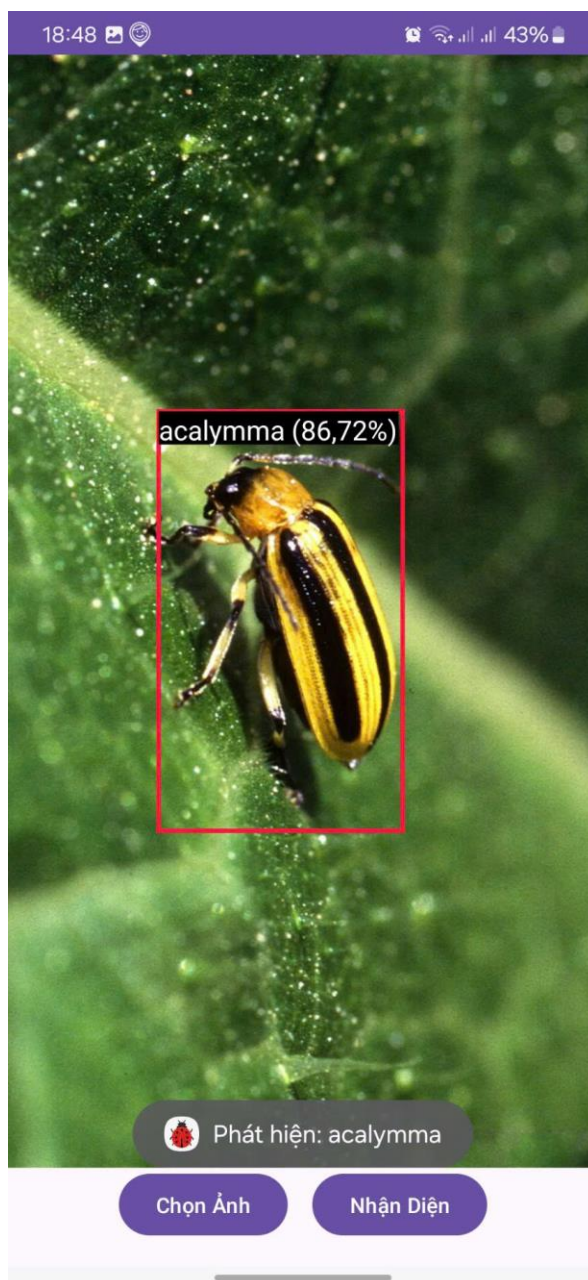
Hình 29: Giao diện chatbot

3.5.5 Giao diện detect côn trùng từ camera



Hình 30: Giao diện nhận dạng camera

3.5.6 Giao diện detect côn trùng từ ảnh trong thư viện



Hình 31: Giao diện nhận dạng bằng ảnh

3.5.7 Giao diện xem thông tin chi tiết côn trùng

18:51

43%



Bọ xít bí

Tên khoa học: *Anasa tristis* (Squash Bug)

Non-toxic

Harmful to plants

Mô tả

Hút nhựa cây bầu bí làm cây héo và suy yếu, có thể gây chết cây.

Hút nhựa cây bầu bí làm cây héo và suy yếu, có thể gây chết cây.

Phân bố

Bắc Mỹ và Trung Mỹ.

Hình thái

Cơ thể dài khoảng 15mm, màu xám nâu, có mùi hôi khi bị đe dọa.

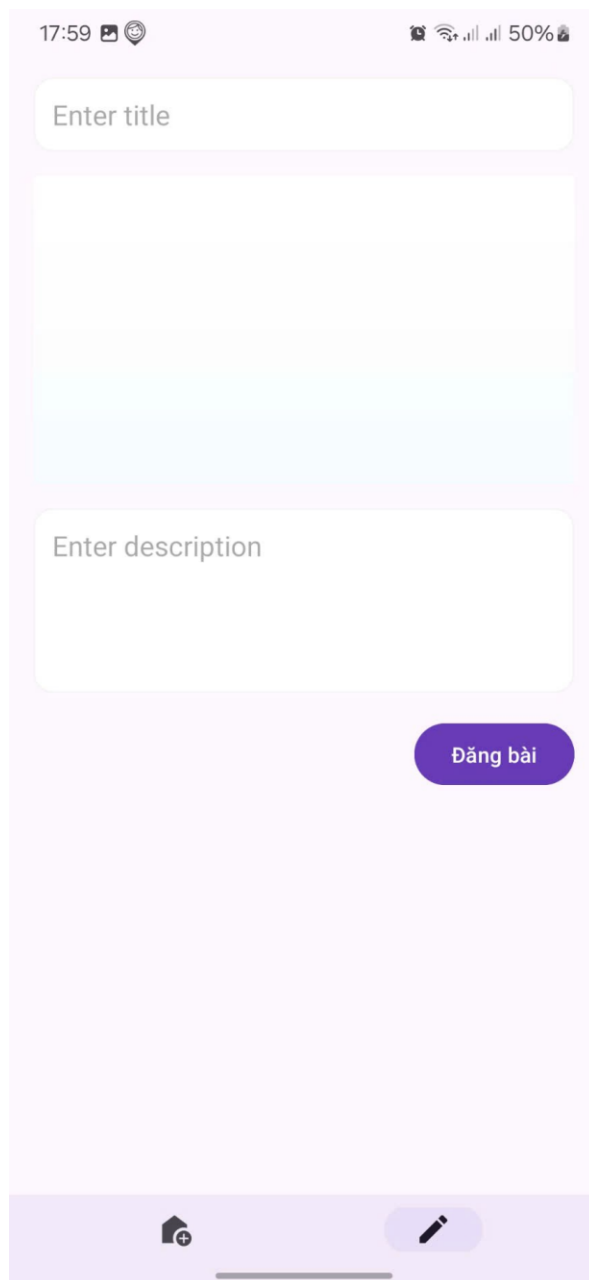
Biện pháp phòng trừ

Loại bỏ trứng, dùng vải phủ cây, sử dụng thiên địch như ong ký sinh *Trichopoda pennipes*.

Bọ xít bí

Hình 32: Giao diện xem chi tiết côn trùng

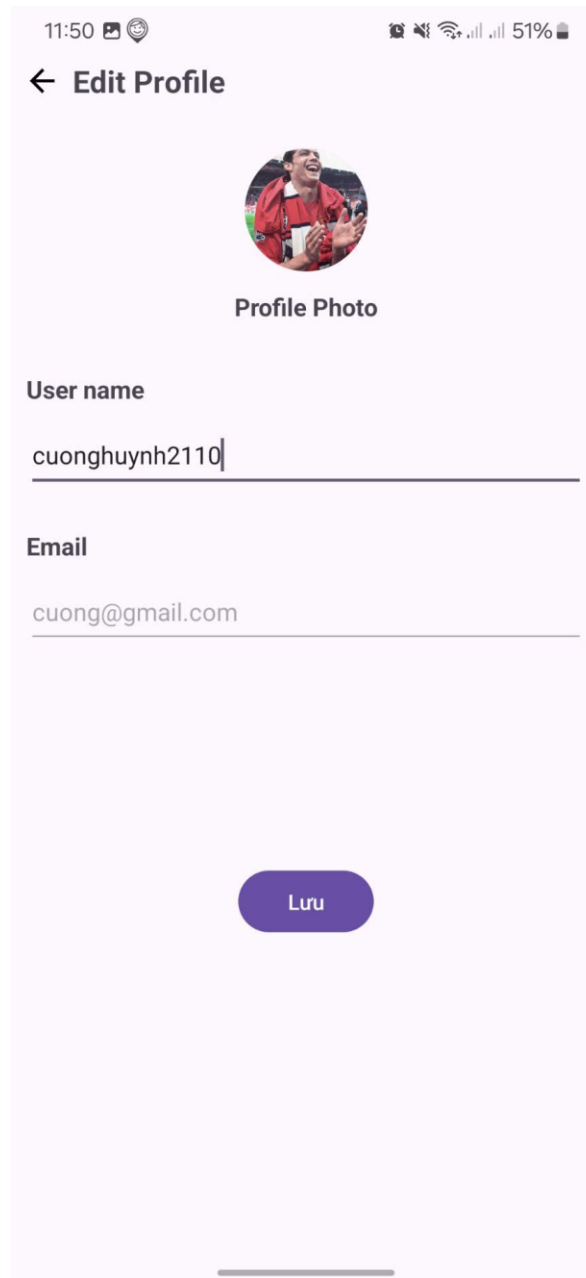
3.5.8 Giao diện Admin đăng bài viết



The image shows a mobile application interface for an admin user to create a new article. The interface is displayed on a light purple background. At the top, there is a status bar showing the time 17:59, signal strength, and 50% battery. Below the status bar, there is a text input field with the placeholder text "Enter title". Underneath the title field is a large text area for the article content, which is divided into three horizontal sections with a light yellow background for the top two and a light blue background for the bottom one. Below the content area is another text input field with the placeholder text "Enter description". To the right of the description field is a purple button with the text "Đăng bài" (Post). At the bottom of the screen, there is a navigation bar with two icons: a home icon with a plus sign and a pencil icon.

Hình 33: Giao diện tạo bài viết admin

3.5.9 Giao diện sửa thông tin người dùng



Hình 34: Giao diện sửa thông tin người dùng

3.6 KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

3.6.1 Kết luận

Ứng dụng di động được xây dựng bằng ngôn ngữ lập trình Kotlin và tích hợp mô hình YOLOv9s mang đến một công cụ hỗ trợ hữu ích cho người nông dân trong việc phát hiện và kiểm soát côn trùng gây hại một cách hiệu quả. Nhờ khả năng nhận diện đối tượng nhanh chóng, chính xác cùng với việc cung cấp các thông tin liên quan đến từng loài côn trùng, ứng dụng giúp người dùng chủ động lựa chọn biện pháp phòng trừ và xử lý phù hợp, góp phần nâng cao hiệu quả trong canh tác nông nghiệp.

Tuy nhiên, do mô hình YOLOv9s trong ứng dụng mới chỉ được huấn luyện trên tập dữ liệu sẵn có gồm 10 loài côn trùng, nên khả năng nhận diện các loài mới còn hạn chế và độ chính xác vẫn chưa đạt mức tối ưu. Ngoài ra, mặc dù YOLOv9s là một phiên bản cải tiến, nhưng vẫn còn tiềm năng để nâng cấp nhằm đạt kết quả nhận diện tốt hơn trong thực tế.

Quá trình thực hiện đề tài đã giúp tôi nắm vững các kiến thức nền tảng về nhận diện đối tượng, bao gồm các khái niệm như mạng nơ-ron tích chập (CNN), thuật toán YOLO, học chuyển giao (Transfer Learning), cũng như kỹ thuật xây dựng mô hình học sâu. Bên cạnh đó, tôi cũng tích lũy được kỹ năng phát triển ứng dụng di động trên nền tảng Android bằng ngôn ngữ Kotlin. Không chỉ vậy, việc nghiên cứu về các loài côn trùng cũng đã mở ra nhiều ý tưởng mới, tạo nền tảng để tôi tiếp tục cải tiến và hoàn thiện ứng dụng nhằm đáp ứng tốt hơn nhu cầu thực tiễn của người nông dân.

3.6.2 Hướng phát triển

Mặc dù ứng dụng nhận dạng côn trùng đã đạt được những kết quả ban đầu tích cực, tuy nhiên để nâng cao hiệu quả sử dụng trong thực tế và đáp ứng tốt hơn nhu cầu đa dạng của người nông dân, đề tài vẫn còn nhiều tiềm năng để tiếp tục phát triển trong các hướng sau:

Mở rộng tập dữ liệu huấn luyện: Việc bổ sung thêm nhiều hình ảnh và chủng loại côn trùng mới từ các vùng miền khác nhau sẽ giúp mô hình nâng cao khả năng nhận diện và mở rộng phạm vi ứng dụng. Đồng thời, việc xây dựng tập dữ liệu chất lượng cao, được gán nhãn chính xác từ chuyên gia nông nghiệp sẽ góp phần cải thiện độ chính xác và độ tin cậy của mô hình.

Tối ưu hóa mô hình nhận dạng: Mặc dù YOLOv9s cho tốc độ xử lý nhanh, nhưng còn hạn chế với các thiết bị có cấu hình thấp vì vậy cần kết hợp các kỹ thuật tối ưu hóa như Pruning, Quantization để cải thiện hiệu suất và giảm dung lượng mô hình, phù hợp với các thiết bị di động có cấu hình thấp.

Phát triển chức năng cộng đồng chia sẻ: Xây dựng nền tảng để người nông dân có thể chia sẻ hình ảnh, thảo luận và nhận phản hồi từ các chuyên gia hoặc cộng đồng người dùng sẽ giúp tạo ra hệ sinh thái hỗ trợ nông nghiệp bền vững. Điều này cũng đồng thời giúp cải thiện dữ liệu huấn luyện từ thực tiễn.

Hỗ trợ đa nền tảng và ngôn ngữ: Bên cạnh hệ điều hành Android, đề tài có thể mở rộng để phát triển phiên bản trên iOS và hỗ trợ đa ngôn ngữ, từ đó tiếp cận được nhiều đối tượng người dùng hơn, kể cả ở nước ngoài.

PHỤ LỤC

HƯỚNG DẪN SỬ DỤNG ỨNG DỤNG

Người dùng có thể cài đặt ứng dụng thông qua việc tải ứng dụng xuống từ CH-Play hoặc thông qua file apk. Khi cài đặt thành công thì người dùng tiến sử dụng như sau:

- Bước 1: Mở ứng dụng lên và cấp quyền truy cập vào điện thoại cho ứng dụng
- Bước 2: Nếu người dùng chưa có tài khoản để sử dụng thì tiến hành đăng ký tài khoản.
- Bước 3: Khi đăng nhập thành công thì người dùng có thể tiến hành các chức năng như nhận diện côn trùng, xem danh sách côn trùng, xem danh sách hình ảnh côn trùng, xem các thông tin chi tiết về côn trùng vừa nhận diện. Ngoài ra người dùng có thể di chuyển dễ dàng giữa các màn hình thông qua thanh điều hướng phía dưới.

TÀI LIỆU THAM KHẢO

- A Gentle Introduction to Transfer Learning for Deep Learning—MachineLearningMastery.com.* (n.d.). Retrieved April 14, 2025, from <https://machinelearningmastery.com/transfer-learning-for-deep-learning/>
- A new mobile application of agricultural pests recognition using deep learning in cloud computing system—ScienceDirect.* (n.d.). Retrieved April 13, 2025, from <https://www.sciencedirect.com/science/article/pii/S1110016821001642>
- A Smartphone-Based Application for Scale Pest Detection Using Multiple-Object Detection Methods.* (n.d.). Retrieved April 14, 2025, from <https://www.mdpi.com/2079-9292/10/4/372>
- Android Là Gì -Tổng Quan Về Hệ Điều Hành Di Động Số 1.* (n.d.). Retrieved April 12, 2025, from <https://itnavi.com.vn/blog/android-he-dieu-hanh-di-dong-dan-dau-tren-the-gioi>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020a). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (No. arXiv:2004.10934). arXiv. <https://doi.org/10.48550/arXiv.2004.10934>
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020b). *YOLOv4: Optimal Speed and Accuracy of Object Detection* (No. arXiv:2004.10934). arXiv. <https://doi.org/10.48550/arXiv.2004.10934>

Brownlee, J. (2017, December 19). A Gentle Introduction to Transfer Learning for Deep Learning. *MachineLearningMastery.Com*.
<https://www.machinelearningmastery.com/transfer-learning-for-deep-learning/>

Brownlee, J. (2019, February 7). How to Improve Performance With Transfer Learning for Deep Learning Neural Networks. *MachineLearningMastery.Com*.
<https://www.machinelearningmastery.com/how-to-improve-performance-with-transfer-learning-for-deep-learning-neural-networks/>

Côn trùng. (2024). In *Wikipedia tiếng Việt*.
https://vi.wikipedia.org/w/index.php?title=C%C3%B4n_tr%C3%B9ng&oldid=71954817#Ph%C3%A2n_lo%E1%BA%A1i

CON TRUNG HAI LUA. (n.d.). Trám BVTV Huyen Vinh Hung. Retrieved April 13, 2025, from <http://trambvtvvinhhung.weebly.com/con-trung-hai-lua.html>

Figure 6. The architecture of YOLOv2. (n.d.). ResearchGate. Retrieved April 15, 2025, from https://www.researchgate.net/figure/The-architecture-of-YOLOv2_fig4_336177198

Giới thiệu—Khoa Công nghệ thông tin—ĐHAG. (n.d.). Retrieved April 12, 2025, from <https://fit.agu.edu.vn/bai-viet/gioi-thieu/gioi-thieu-1.html>

Girshick, R. (2015). *Fast R-CNN* (No. arXiv:1504.08083). arXiv.
<https://doi.org/10.48550/arXiv.1504.08083>

- Hệ thống giám sát côn trùng thông minh—Cổng thông tin điện tử Cục Thông tin, Thống kê.* (n.d.). Retrieved April 13, 2025, from <https://vista.gov.vn/vi/news/print/khoa-hoc-ky-thuat-va-cong-nghe/he-thong-giam-sat-con-trung-thong-minh-7339.html>
- Jocher, G. (2020). *YOLOv5 by Ultralytics* (Version 7.0) [Python]. <https://doi.org/10.5281/zenodo.3908559>
- Khanam, R., & Hussain, M. (2024). *YOLOv11: An Overview of the Key Architectural Enhancements* (No. arXiv:2410.17725). arXiv. <https://doi.org/10.48550/arXiv.2410.17725>
- Khánh, P. Đ. (n.d.). *Khoa học dữ liệu*. Khanh's Blog. Retrieved April 15, 2025, from <https://phamdinhhkhanh.github.io>
- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6), 84–90. <https://doi.org/10.1145/3065386>
- Le, A. D., Pham, D. A., Pham, D. T., & Vo, H. B. (2024). AlertTrap: A study on object detection in remote insects trap monitoring system using on-the-edge deep learning platform. *Journal of Computational and Cognitive Engineering*. <https://doi.org/10.47852/bonviewJCCE42023264>
- Lee, J., & Hwang, K. (2022). YOLO with adaptive frame control for real-time object detection applications. *Multimedia Tools and Applications*, 81(25), 36375–36396. <https://doi.org/10.1007/s11042-021-11480-0>

- Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). *YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications* (No. arXiv:2209.02976). arXiv. <https://doi.org/10.48550/arXiv.2209.02976>
- Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., & Zitnick, C. L. (2014). Microsoft COCO: Common Objects in Context. In D. Fleet, T. Pajdla, B. Schiele, & T. Tuytelaars (Eds.), *Computer Vision – ECCV 2014* (pp. 740–755). Springer International Publishing. https://doi.org/10.1007/978-3-319-10602-1_48
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). *SSD: Single Shot MultiBox Detector* (Vol. 9905, pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2
- Perrin, T. (2024). *Uniform estimates for solutions of nonlinear focusing damped wave equations* (No. arXiv:2403.06541). arXiv. <https://doi.org/10.48550/arXiv.2403.06541>
- PostgreSQL: About.* (n.d.). Retrieved April 13, 2025, from <https://www.postgresql.org/about/>
- PostgreSQL: Documentation.* (n.d.). Retrieved April 13, 2025, from <https://www.postgresql.org/docs/>

- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016b). *You Only Look Once: Unified, Real-Time Object Detection* (No. arXiv:1506.02640). arXiv. <https://doi.org/10.48550/arXiv.1506.02640>
- Redmon, J., & Farhadi, A. (2016). *YOLO9000: Better, Faster, Stronger* (No. arXiv:1612.08242). arXiv. <https://doi.org/10.48550/arXiv.1612.08242>
- Redmon, J., & Farhadi, A. (2018). *YOLOv3: An Incremental Improvement* (No. arXiv:1804.02767). arXiv. <https://doi.org/10.48550/arXiv.1804.02767>
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Advances in Neural Information Processing Systems*, 28. https://proceedings.neurips.cc/paper_files/paper/2015/hash/14bfa6bb14875e45bba028a21ed38046-Abstract.html
- Supabase. (2025, April 11). *Architecture | Supabase Docs*. <https://supabase.com/docs/guides/getting-started/architecture>
- Supabase: Giải pháp Backend-as-a-Service mã nguồn mở*. (n.d.). Retrieved April 13, 2025, from <https://200lab.io/blog/supabase-la-gi/>
- Supabase Tính năng, Ưu điểm, Nhược điểm và Trường hợp sử dụng*. (n.d.). Retrieved April 13, 2025, from https://aipure.ai/vn/products/supabase/features?utm_source=chatgpt.com
- Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR),
2818–2826. <https://doi.org/10.1109/CVPR.2016.308>

Theertha, L. G. C., & Wanniarachchi, W. (2020). *Smart Pest Recognition System for Sri Lankan Crop-Growing*.
<http://ir.kdu.ac.lk/handle/345/2991>

Tìm hiểu về YOLO trong bài toán real-time object detection. (n.d.). Retrieved
April 14, 2025, from <https://viblo.asia/p/tim-hieu-ve-yolo-trong-bai-toan-real-time-object-detection-yMnKMdvr57P>

TopDev, B. biên tập. (2023, May 25). Kotlin là gì? Kotlin và Java khác nhau
như thế nào? *TopDev*. <https://topdev.vn/blog/kotlin-la-gi/>

Ultralytics. (n.d.). *YOLOv8*. Retrieved April 13, 2025, from
<https://docs.ultralytics.com/vi/models/yolov8>

Ultralytics YOLO11—Ultralytics YOLO Tài liệu. (n.d.). Retrieved April 13,
2025, from <https://docs.ultralytics.com/vi/models/yolo11/>

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M. (2022). *YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*
(No. arXiv:2207.02696). arXiv.
<https://doi.org/10.48550/arXiv.2207.02696>

Yaseen, M. (2024). *What is YOLOv9: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector* (No.
arXiv:2409.07813). arXiv. <https://doi.org/10.48550/arXiv.2409.07813>

YOLO with adaptive frame control for real-time object detection applications /
Multimedia Tools and Applications. (n.d.). Retrieved April 13, 2025,
from <https://link.springer.com/article/10.1007/s11042-021-11480-0>