

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**NGHIÊN CỨU YOLOV8, XÂY DỰNG ỨNG DỤNG
ANDROID NHẬN DIỆN ĐỘ CHÍN TRÁI XOÀI**

Người hướng dẫn : TS. NGUYỄN HÀ HUY CƯỜNG
Sinh viên thực hiện: NGUYỄN MẠNH QUÝ
Mã số sinh viên: 2151220236
Sinh viên thực hiện: TRẦN THANH QUÝ HUY
Mã số sinh viên: 2151220268
Lớp:21CT5

Đà Nẵng, 12/2025

TRƯỜNG ĐẠI HỌC KIẾN TRÚC ĐÀ NẴNG
KHOA CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN TỐT NGHIỆP
NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI:

**NGHIÊN CỨU YOLOV8, XÂY DỰNG ỨNG DỤNG
ANDROID NHẬN DIỆN ĐỘ CHÍN TRÁI XOÀI**

Người hướng dẫn : TS. NGUYỄN HÀ HUY CƯỜNG

Sinh viên thực hiện: NGUYỄN MẠNH QUÝ

Mã số sinh viên: 2151220236

Sinh viên thực hiện: TRẦN THANH QUÝ HUY

Mã số sinh viên: 2151220268

Lớp: 21CT5

Đà Nẵng, 12/2025

TÓM TẮT

Tên đề tài: Nghiên cứu YOLOv8, xây dựng ứng dụng Android nhận diện độ chín trái xoài.

Sinh viên thực hiện:

- | | | |
|-----------------------|------------------|------------|
| 1. Nguyễn Mạng Quý | MSSV: 2151220236 | Lớp: 21CT5 |
| 2. Trần Thanh Quý Huy | MSSV: 2151220268 | Lớp: 21CT5 |

Đề tài này tập trung vào việc nghiên cứu và xây dựng ứng dụng trên thiết bị di động (Android) có khả năng nhận diện trái xoài theo thời gian thực hoặc qua hình ảnh chụp, hỗ trợ người dùng trong việc xác định và lưu trữ thông tin nông sản.

Về mặt kỹ thuật, đề tài sử dụng mô hình học sâu YOLOv8 (You Only Look Once version 8) nhờ ưu điểm vượt trội về tốc độ và độ chính xác trong bài toán phát hiện đối tượng (Object Detection). Quy trình xây dựng mô hình bắt đầu bằng việc thu thập và gán nhãn dữ liệu trên nền tảng Roboflow, sau đó tiến hành huấn luyện (training) mô hình trên môi trường Google Colab để tận dụng sức mạnh tính toán GPU.

Sau khi huấn luyện, mô hình được tích hợp vào ứng dụng di động được phát triển trên môi trường Android Studio. Để hoàn thiện tính năng của một ứng dụng thực tế, hệ thống tích hợp Firebase để quản lý xác thực tài khoản người dùng và sử dụng Cloudinary làm giải pháp lưu trữ hình ảnh đám mây, giúp tối ưu hóa dung lượng thiết bị và dễ dàng truy xuất dữ liệu. Kết quả thử nghiệm cho thấy ứng dụng hoạt động ổn định, nhận diện tốt trái xoài trong các điều kiện ánh sáng khác nhau và quy trình lưu trữ dữ liệu diễn ra liền mạch.

LỜI NÓI ĐẦU

Đồ án tốt nghiệp này không chỉ là một bài kiểm tra cuối khóa, mà là cột mốc đánh dấu sự trưởng thành của em sau bốn năm ngồi trên giảng đường đại học. Khi chọn đề tài “Xây dựng ứng dụng Android nhận diện trái xoài sử dụng YOLOv8”, thú thật là em đã rất hào hứng nhưng cũng đầy lo lắng. Hào hứng vì được chạm tay vào những công nghệ AI mới nhất, nhưng lo lắng vì khoảng cách từ lý thuyết đến một ứng dụng chạy mượt mà trên điện thoại là cả một chặng đường dài đầy chông gai.

Để hoàn thành đồ án này, bên cạnh sự nỗ lực của bản thân, em đã nhận được rất nhiều sự quan tâm, giúp đỡ quý báu. Trước hết, em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến thầy Nguyễn Hà Huy Cường. Thầy đã tận tình hướng dẫn, định hướng nghiên cứu và chỉ bảo những kiến thức chuyên môn quý báu cho em trong suốt quá trình thực hiện đề tài. Những lời khuyên của Thầy không chỉ giúp em giải quyết các bài toán kỹ thuật hóc búa về YOLOv8 hay Android Studio, mà còn giúp em rèn luyện tư duy làm việc khoa học.

Em cũng xin trân trọng cảm ơn quý Thầy, Cô giáo trong Khoa Công nghệ thông tin, Trường Đại học Kiến trúc Đà Nẵng đã truyền đạt kiến thức nền tảng vững chắc trong những năm học vừa qua. Đây chính là hành trang quan trọng để em có thể tiếp cận và làm chủ các công nghệ mới trong đồ án này.

Cuối cùng, em xin cảm ơn gia đình và bạn bè đã luôn động viên, khích lệ tinh thần và hỗ trợ em trong những lúc gặp khó khăn.

Mặc dù đã rất cố gắng để hoàn thiện đồ án một cách tốt nhất, nhưng do giới hạn về mặt thời gian và kiến thức, đề tài khó tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp, nhận xét từ quý Thầy/Cô để đề tài được hoàn thiện và phát triển hơn nữa trong tương lai.

Em xin chân thành cảm ơn!

CAM ĐOAN

Em xin cam đoan đồ án này là công trình nghiên cứu của nhóm em, dưới sự hướng dẫn của Thầy Nguyễn Hà Huy Cường.

Các nội dung nghiên cứu, kết quả xây dựng ứng dụng nhận diện trái xoài nêu trong Đồ án Tốt nghiệp là trung thực, không phải là sao chép toàn văn của bất kỳ công trình nào khác. Các nguồn trích dẫn có chú thích rõ ràng, có tính kế thừa và phát triển từ các tài liệu đã được liệt kê ở mục Tài liệu tham khảo.

Em xin chân thành chịu trách nhiệm về lời cam đoan của mình.

Sinh viên thực hiện

{Chữ ký, họ và tên sinh viên}

MỤC LỤC

TÓM TẮT	4
LỜI NÓI ĐẦU.....	i
CAM ĐOAN.....	ii
MỤC LỤC	iii
DANH SÁCH CÁC BẢNG, HÌNH VẼ	v
DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT	vii
MỞ ĐẦU	1
1.1 Mục đích thực hiện đề tài:	1
1.2 Mục tiêu đề tài:	1
1.3 Phạm vi và đối tượng nghiên cứu:	1
1.4 Phương pháp nghiên cứu :	2
1.5 Cấu trúc của đồ án :	2
CHƯƠNG 1: TỔNG QUAN LÝ THUYẾT	4
1.1 Tổng quan về bài toán nhận diện đối tượng.....	4
1.1.1 Khái niệm Deep Learning và CNN	4
1.1.2 Giới thiệu về bài toán Object Detection	4
1.2 Mô hình YOLOv8 (You Only Look Once):.....	5
1.2.1 Lịch sử phát triển các phiên bản YOLO.....	5
1.2.3 Lý do lựa chọn YOLOv8:	8
1.2.5 Kỹ thuật tăng cường dữ liệu (Data Augmentation) trong YOLOv8	10
1.2.6 Các chỉ số đánh giá hiệu năng mô hình	12
1.2.7 Cơ chế tối ưu hóa cho thiết bị di động (TFLite Export)	14
1.3 Nền tảng phát triển ứng dụng	15
1.3.1 Hệ điều hành Android và Android Studio	15
1.3.2 Ngôn ngữ lập trình (Java/Kotlin)	17
1.4 Các công cụ và dịch vụ tích hợp	18
1.4.1 Roboflow: Quản lý và tiền xử lý dữ liệu ảnh.....	18
CHƯƠNG 2: CÀI ĐẶT VÀ TRIỂN KHAI	26
2.1 Quá trình huấn luyện mô hình	26
2.1.1 Bộ dữ liệu trái xoài	26
2.1.2 Gán nhãn với Roboflow	27

2.1.3 Chia tập dữ liệu.....	29
2.1.4 Tiền xử lý dữ liệu hình ảnh	29
2.1.5 Cấu hình file huấn luyện	31
2.1.6 Thực hiện huấn luyện trên Google Colab	31
2.1.7 Theo dõi và đánh giá kết quả huấn luyện	32
2.1.8 Kiểm thử và đánh giá mô hình	33
2.2 Tích hợp mô hình vào Android Studio	34
2.2.1 Xuất model sang TFLite với metadata	34
2.2.2 Tích hợp TFLite vào Android Studio	35
2.3 Các dịch vụ đám mây	36
2.3.1 Firebase	36
2.3.2 Cloudinary	37
CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ.....	39
3.1 Demo sản phẩm	39
3.1.1 Đăng nhập, đăng kí.....	39
3.1.2 Chụp và nhận diện sản phẩm	39
3.1.3 Nhận diện hình ảnh từ điện thoại	40
3.1.4 Lịch sử nhận diện	41
3.1.5 Xóa lịch sử	41
3.1.6 Xuất file.....	42
3.1.7 Giao diện admin.....	42
3.1.8 Đổi mật khẩu	43
3.2 Bảng số liệu	44
3.3 Ưu nhược điểm hệ thống.....	44
3.3.1 Ưu điểm.....	44
3.3.2 Nhược điểm.....	44
KẾT LUẬN	46
1. Kết luận chung	46
2. Các kết quả đạt được	46
3. Hạn chế của đề tài	47
4. Đề xuất và Kiến nghị.....	48
TÀI LIỆU THAM KHẢO	51

DANH SÁCH CÁC BẢNG, HÌNH VẼ

BẢNG 1.1 Bảng so sánh các thông số của các phiên bản YOLOv8	10
BẢNG 3.1 Bảng số liệu nhận diện	44
.....	
HÌNH 1.1 Cấu trúc tổng quan của YOLOv8	7
HÌNH 1.2 Dữ liệu ảnh tải lên Roboflow.....	18
HÌNH 1.3 Gán nhãn đối tượng trên Roboflow	19
HÌNH 1.4 Tăng cường và chia tập dữ liệu.....	20
HÌNH 1.5 Xuất dữ liệu để huấn luyện môi trường Google Colab	20
HÌNH 2.1 Giao diện gán nhãn hình ảnh với Roboflow	29
HÌNH 2.2 Biểu đồ chia dữ liệu Train/Val/Test trên Roboflow	29
HÌNH 2.3 Tiền xử lý dữ liệu hình ảnh.....	30
HÌNH 2.4 Cấu trúc file huấn luyện	31
HÌNH 2.5 Thực hiện huấn luyện trên Google Colab.....	32
HÌNH 2.6 Biểu đồ loss và mAP được tạo trong quá trình huấn luyện YOLOv8	33
HÌNH 2.7 Số liệu (Precision, Recall, mAP) trong quá trình huấn luyện YOLOv8...	33
HÌNH 2.8 Xuất model sang TFLite với metadata	35
HÌNH 2.9: Thêm TFLite vào Android Studio.....	36
HÌNH 2.10 Quản lý người dùng.....	37
HÌNH 2.11 Quản lý hình ảnh	38
HÌNH 3.1 Giao diện đăng nhập, đăng kí	39
HÌNH 3.2 Giao diện nhận diện đối tượng.....	39
HÌNH 3.3 Giao diện chọn ảnh.....	40
HÌNH 3.4 Nhận diện hình ảnh được chọn	40
HÌNH 3.5 Giao diện lịch sử	41
HÌNH 3.6 Xóa lịch sử	41
HÌNH 3.7 Xuất file	42
HÌNH 3.8 Giao diện Admin	42

HÌNH 3.9 Xóa người dùng..... 43

HÌNH 3.10 Giao diện đổi mật khẩu..... 43

.....

DANH SÁCH CÁC KÝ HIỆU, CHỮ VIẾT TẮT

STT	Chữ viết tắt	Tiếng Anh	Tiếng Việt / Giải thích
1	AI	Artificial Intelligence	Trí tuệ nhân tạo
2	ANN	Artificial Neural Network	Mạng nơ-ron nhân tạo
3	API	Application Programming Interface	Giao diện lập trình ứng dụng
4	CNN	Convolutional Neural Network	Mạng nơ-ron tích chập
5	FPS	Frames Per Second	Số khung hình trên giây (Tốc độ xử lý)
6	GPU	Graphics Processing Unit	Đơn vị xử lý đồ họa
7	IDE	Integrated Development Environment	Môi trường phát triển tích hợp (Ví dụ: Android Studio)
8	IoU	Intersection over Union	Chỉ số đo lường độ trùng khớp giữa khung dự đoán và thực tế
9	mAP	mean Average Precision	Độ chính xác trung bình (Chỉ số đánh giá mô hình)
10	SDK	Software Development Kit	Bộ công cụ phát triển phần mềm
11	TFLite	TensorFlow Lite	Định dạng mô hình học sâu tối ưu cho thiết bị di động
12	URL	Uniform Resource Locator	Đường dẫn tham chiếu tài nguyên trên Internet
13	YOLO	You Only Look Once	Mô hình mạng nơ-ron phát hiện đối tượng thời gian thực

MỞ ĐẦU

1.1 Mục đích thực hiện đề tài:

Xuất phát từ nhu cầu thực tiễn của nền nông nghiệp thông minh, mục đích cốt lõi của đề tài là nghiên cứu giải pháp đưa trí tuệ nhân tạo (AI) lên thiết bị di động, giúp người dùng có thể tiếp cận công nghệ nhận diện hình ảnh một cách dễ dàng nhất.

Cụ thể, đề tài hướng đến việc:

Hỗ trợ phân loại nông sản: Tạo ra công cụ giúp nhận diện trái xoài nhanh chóng, thay thế phương pháp quan sát bằng mắt thường vốn phụ thuộc nhiều vào cảm tính.

Làm chủ công nghệ mới: Vận dụng các kiến thức tiên tiến về Thị giác máy tính (Computer Vision) để giải quyết bài toán thực tế.

Tối ưu hóa trải nghiệm người dùng: Kết hợp khả năng xử lý offline (nhận diện trên máy) và lưu trữ online (Cloudinary) để tạo ra một ứng dụng vừa nhanh, vừa nhẹ.

1.2 Mục tiêu đề tài:

Để đạt được các mục đích trên, đề tài đặt ra các mục tiêu cụ thể cần hoàn thành như sau:

Về dữ liệu: Xây dựng bộ dữ liệu hình ảnh trái xoài đa dạng (về góc chụp, ánh sáng) và thực hiện gán nhãn chuẩn xác trên nền tảng Roboflow.

Về mô hình AI: Huấn luyện thành công mô hình YOLOv8 trên môi trường Google Colab, sau đó chuyển đổi (convert) mô hình sang định dạng phù hợp (TFLite) để hoạt động được trên thiết bị di động.

Về ứng dụng: Xây dựng hoàn thiện ứng dụng trên Android Studio với các chức năng: Nhận diện trái xoài qua ảnh. Đăng ký, đăng nhập bảo mật qua Firebase, tải lên và quản lý lịch sử hình ảnh nhận diện qua Cloudinary.

Về đánh giá: Ứng dụng hoạt động ổn định, mô hình đạt độ chính xác chấp nhận được và tốc độ phản hồi nhanh.

1.3 Phạm vi và đối tượng nghiên cứu:

Đối tượng nghiên cứu:

Đặc điểm thị giác của trái xoài: Nghiên cứu các đặc trưng hình ảnh giúp phân biệt trạng thái của quả xoài như: màu sắc vỏ (xanh/vàng), kết cấu bề mặt và các dấu hiệu bất thường (đốm đen, vết thối) để phục vụ cho bài toán nhận diện chất lượng.

Mô hình học sâu YOLOv8: Tập trung tìm hiểu kiến trúc mạng nơ-ron của YOLOv8, cơ chế trích xuất đặc trưng và các tham số ảnh hưởng đến hiệu suất nhận diện trên các thiết bị tài nguyên thấp (mobile).

Hệ sinh thái phát triển ứng dụng di động: Nghiên cứu kỹ thuật tích hợp mô hình AI vào Android (sử dụng TFLite), cơ chế xác thực người dùng của Firebase Authentication và giao thức truyền tải hình ảnh qua API của Cloudinary.

Phạm vi nghiên cứu:

Phạm vi nội dung: Đề tài tập trung vào bài toán nhận diện (Detection) trái xoài, không đi sâu vào việc phân loại chi tiết từng giống xoài (Classification) hay đánh giá độ hư hỏng (trừ khi bộ dữ liệu của bạn có label này).

Phạm vi kỹ thuật: Ứng dụng chạy trên hệ điều hành Android. Mô hình được huấn luyện giới hạn trong tài nguyên miễn phí của Google Colab.

Phạm vi dữ liệu: Dữ liệu hình ảnh được thu thập từ nguồn thực tế và Internet, tập trung vào đối tượng là trái xoài Việt Nam.

1.4 Phương pháp nghiên cứu :

Phương pháp nghiên cứu lý thuyết:

Tìm hiểu nguyên lý hoạt động của mạng nơ-ron và kiến trúc YOLOv8. Nghiên cứu tài liệu về phát triển ứng dụng Android, tích hợp SDK của Firebase và Cloudinary API.

Phương pháp thực nghiệm:

Thu thập và xử lý dữ liệu: Sử dụng Roboflow để tiền xử lý (resize, augment). Huấn luyện và kiểm thử: Train model trên Google Colab, theo dõi các chỉ số Loss, mAP để tinh chỉnh tham số. Triển khai và đánh giá: Đóng gói ứng dụng, cài đặt lên thiết bị thật để kiểm tra độ trễ (latency) và độ chính xác thực tế.

1.5 Cấu trúc của đồ án :

Báo cáo đồ án được trình bày gồm 3 chương :

Chương 1: Tổng quan cơ sở lý thuyết: Trình bày các khái niệm về Deep Learning, bài toán Object Detection, giới thiệu tổng quan về YOLOv8 và nền tảng Android. Giới thiệu chi tiết về các công cụ hỗ trợ: Roboflow (quản lý dữ liệu), Google Colab (môi trường huấn luyện), Firebase (xác thực) và Cloudinary (lưu trữ).

Chương 2: Cài đặt và triển khai: Trình bày quá trình huấn luyện mô hình, các bước lập trình tích hợp mô hình vào Android và kết nối các dịch vụ đám mây.

Chương 3: Kết quả thực nghiệm và đánh giá: Demo giao diện ứng dụng, bảng số liệu đánh giá độ chính xác của mô hình và ưu nhược điểm của hệ thống.

CHƯƠNG 1: TỔNG QUAN LÝ THUYẾT

1.1 Tổng quan về bài toán nhận diện đối tượng

Để giải quyết bài toán nhận diện trái xoài trong đề án này, nền tảng lý thuyết cốt lõi dựa trên Học sâu (Deep Learning) và bài toán Phát hiện đối tượng (Object Detection).

1.1.1 Khái niệm Deep Learning và CNN

Học sâu (Deep Learning): Học sâu là một nhánh chuyên sâu của Học máy (Machine Learning), lấy cảm hứng từ cấu trúc và chức năng của bộ não con người, được gọi là mạng nơ-ron nhân tạo (Artificial Neural Networks - ANN). Khác với các thuật toán học máy truyền thống cần sự can thiệp của con người để trích xuất đặc trưng (feature extraction), các mô hình học sâu có khả năng tự động học các đặc trưng từ dữ liệu thô thông qua nhiều tầng ẩn (hidden layers).

Mỗi tầng trong mạng nơ-ron học sâu sẽ biến đổi dữ liệu đầu vào thành các biểu diễn trừu tượng hơn. Ví dụ, trong bài toán nhận diện ảnh, các tầng đầu tiên có thể nhận diện các cạnh, góc; các tầng tiếp theo nhận diện hình dáng (tròn, bầu dục); và các tầng cuối cùng nhận diện toàn bộ đối tượng (trái xoài).

Còn về Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN): Trong lĩnh vực xử lý ảnh, CNN được xem là kiến trúc mạng hiệu quả nhất và là nền tảng của hầu hết các hệ thống nhận diện hiện đại (bao gồm cả YOLO).

1.1.2 Giới thiệu về bài toán Object Detection

Phát hiện đối tượng (Object Detection) là một bài toán cơ bản nhưng phức tạp trong thị giác máy tính. Nếu như bài toán Phân loại ảnh (Image Classification) chỉ trả lời câu hỏi "Trong ảnh có cái gì?" (ví dụ: ảnh này là quả xoài), thì bài toán phát hiện đối tượng phải giải quyết nhiệm vụ phân loại (Classification): Xác định danh tính của đối tượng (Xoài xanh, Xoài chín, hay Xoài hỏng).

Đầu ra của bài toán: Với mỗi đối tượng được phát hiện trong ảnh, mô hình sẽ trả về các thông tin sau: Nhãn lớp (Class Label): Tên của đối tượng được dự đoán (Ví dụ: "Ripe Mango"). Độ tin cậy (Confidence Score): Một giá trị xác suất (thường từ 0 đến 1) thể hiện mức độ chắc chắn của mô hình về dự đoán đó.

Ứng dụng trong đề án: Trong phạm vi đề án này, bài toán Object Detection được lựa chọn thay vì Classification đơn thuần bởi tính ứng dụng thực tế cao hơn. Một bức ảnh chụp thực tế thường chứa nhiều trái xoài với các trạng thái khác nhau (vừa có quả xanh, vừa có quả hỏng) nằm lẫn lộn. Object Detection cho phép hệ thống đếm số lượng, định vị từng quả và đánh giá chất lượng riêng biệt cho

từng đối tượng trong cùng một khung hình, điều mà bài toán phân loại đơn thuần không thể thực hiện được.

1.2 Mô hình YOLOv8 (You Only Look Once):

1.2.1 Lịch sử phát triển các phiên bản YOLO

Thuật toán YOLO (You Only Look Once) đại diện cho một bước ngoặt lịch sử trong lĩnh vực thị giác máy tính, được giới thiệu lần đầu tiên vào năm 2015 bởi Joseph Redmon. Trước khi YOLO xuất hiện, các hệ thống nhận diện đối tượng chủ yếu dựa trên các phương pháp "Two-stage" (hai giai đoạn) như R-CNN, vốn thực hiện tách biệt việc đề xuất vùng chứa đối tượng và phân loại chúng. YOLO đã phá vỡ rào cản này bằng cách đề xuất một hướng tiếp cận "One-stage" (một giai đoạn) đột phá: coi việc phát hiện đối tượng như một bài toán hồi quy đơn lẻ.

Giai đoạn khởi đầu (YOLOv1 - YOLOv3): * YOLOv1 (2015): Là nền tảng cho tư duy "nhìn một lần", dự đoán trực tiếp các khung bao (bounding boxes) và xác suất lớp từ toàn bộ hình ảnh trong một lần tính toán duy nhất. Tuy nhiên, phiên bản này còn hạn chế trong việc nhận diện các vật thể nhỏ nằm sát nhau.

YOLOv2 (2016): Còn được gọi là YOLO9000, giới thiệu các kỹ thuật quan trọng như Batch Normalization và sử dụng Anchor Boxes để cải thiện khả năng định vị đối tượng.

YOLOv3 (2018): Sử dụng mạng Darknet-53 mạnh mẽ hơn và kỹ thuật dự đoán đa quy mô (multi-scale predictions), giúp cải thiện đáng kể độ chính xác đối với các vật thể nhỏ.

Giai đoạn bùng nổ hiệu năng (YOLOv4 - YOLOv7):

Đây là giai đoạn mà quyền quản lý mã nguồn bắt đầu chuyển giao cho các nhóm nghiên cứu khác nhau.

YOLOv4 và YOLOv5: Mang đến sự tối ưu vượt trội về tốc độ huấn luyện và khả năng triển khai. Đặc biệt, YOLOv5 do Ultralytics phát triển đã trở thành tiêu chuẩn công nghiệp nhờ tính dễ dùng và thư viện hỗ trợ tận tình.

YOLOv6 và YOLOv7: Tiếp tục đẩy giới hạn về tốc độ xử lý (FPS) và độ chính xác (mAP) lên những tầm cao mới thông qua các cải tiến về kiến trúc mạng như E-ELAN.

Kỷ nguyên hiện đại - YOLOv8 (Tháng 1/2023): * Được chính thức ra mắt bởi Ultralytics – đơn vị đứng sau thành công vang dội của YOLOv5.

YOLOv8 không chỉ là một sự nâng cấp về thông số, mà là một bước nhảy vọt về kiến trúc. Nó kế thừa hoàn hảo những ưu điểm từ các

thể hệ trước, đồng thời tích hợp những kỹ thuật tiên tiến nhất như kiến trúc Anchor-Free (không dùng khung neo) và các module C2f cải tiến.

Đây hiện được xem là phiên bản tiên tiến nhất (State-of-the-Art), được thiết kế như một nền tảng đa năng, hỗ trợ từ phát hiện đối tượng, phân đoạn hình ảnh (segmentation) cho đến ước lượng tư thế (pose estimation).

Việc lựa chọn YOLOv8 cho bài toán nhận diện độ chín của trái xoài trong đề án này không chỉ dựa trên danh tiếng của dòng họ YOLO, mà còn dựa trên khả năng tối ưu hóa thực tế của nó đối với các thiết bị di động có tài nguyên giới hạn. Sự kế thừa và phát triển qua gần một thập kỷ đã biến YOLOv8 thành một công cụ mạnh mẽ, cân bằng hoàn hảo giữa tốc độ xử lý thời gian thực và độ chính xác cần thiết cho nông nghiệp thông minh.

1.2.2 Kiến trúc mạng và cải tiến của YOLOv8

YOLOv8 mang đến những thay đổi đáng kể về mặt kiến trúc so với các phiên bản tiền nhiệm:

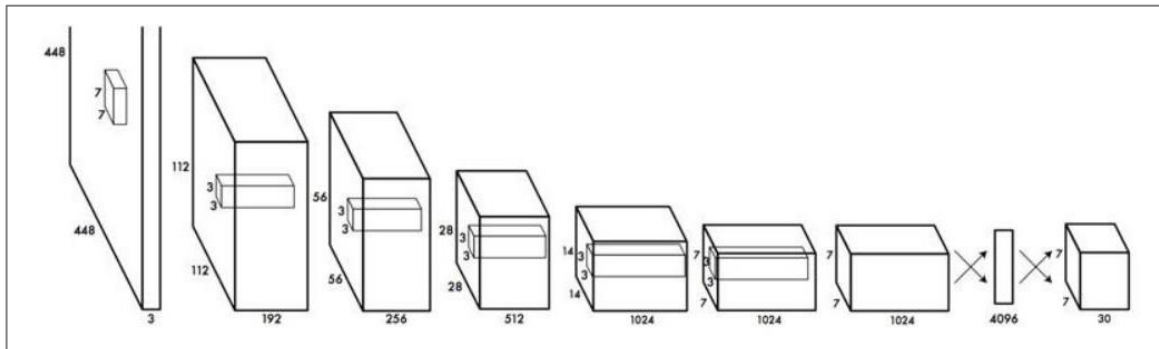
Anchor-Free Detection (Phát hiện không dùng neo): Đây là cải tiến lớn nhất. Thay vì dựa vào các khung neo (anchor boxes) được định nghĩa trước (dễ gây khó khăn khi đối tượng có hình dạng bất thường), YOLOv8 dự đoán trực tiếp tâm của đối tượng. Điều này giúp mô hình linh hoạt hơn và giảm bớt số lượng siêu tham số cần tinh chỉnh.

Module C2f mới: Phần Backbone (xương sống) của mạng thay thế khối C3 cũ bằng khối C2f. C2f được thiết kế để cải thiện luồng gradient, giúp mô hình học được các đặc trưng phong phú hơn nhưng vẫn đảm bảo trọng lượng nhẹ.

Decoupled Head (Đầu ra tách biệt): YOLOv8 tách riêng hai nhánh nhiệm vụ: một nhánh xử lý việc phân loại (Classification) và một nhánh xử lý việc định vị khung hình (Regression). Việc tách biệt này giúp tăng độ chính xác hội tụ của mô hình.

Hàm mất mát (Loss Function) mới: Sử dụng VFL Loss cho phân loại và DFL Loss + CIOU Loss cho định vị, giúp mô hình "phạt" nặng hơn các dự đoán sai lệch vị trí.

Cấu trúc tổng quan của YOLOv8



Hình 1.1 Cấu trúc tổng quan của YOLOv8

YOLOv8 bao gồm ba thành phần chính trong kiến trúc:

- **Backbone:** Chịu trách nhiệm trích xuất đặc trưng từ ảnh đầu vào. Trong YOLOv8, backbone sử dụng kiến trúc CSPDarknet hoặc một số biến thể nhẹ hơn, giúp tăng khả năng học đặc trưng mà không tăng nhiều tham số.
- **Neck:** Là cầu nối giữa backbone và head, dùng để tổng hợp các đặc trưng ở nhiều cấp độ khác nhau. Neck thường sử dụng FPN (Feature Pyramid Network) hoặc PANet nhằm tăng khả năng nhận diện đối tượng ở nhiều kích thước.
- **Head:** Đầu ra của mô hình, nơi diễn ra quá trình dự đoán các bounding box, class (nhãn đối tượng), confidence score (độ tin cậy) tương ứng.

Ngoài ra YOLOv8 còn sử dụng các kỹ thuật như:

- **Anchor-free detection:** Không cần định nghĩa trước các anchor box, giúp đơn giản hoá quá trình huấn luyện.
- **Auto-learning bounding box:** Cải thiện độ chính xác trong việc dự đoán vị trí và kích thước vật thể.
- **Loss Function cải tiến:** Tối ưu hàm mất mát giúp mô hình hội tụ nhanh và chính xác hơn.

Ưu điểm nổi bật của YOLOv8:

- **Tốc độ xử lý nhanh:** YOLOv8 có thể chạy real-time trên cả CPU và GPU, thích hợp cho ứng dụng yêu cầu phản hồi thời gian thực như camera giám sát hay thiết bị nhúng.
- **Độ chính xác cao:** Được tinh chỉnh và cải tiến thuật toán, YOLOv8 đạt điểm mAP (mean Average Precision) rất cao trên nhiều bộ dữ liệu benchmark.
- **Dễ sử dụng và triển khai:** Thư viện Ultralytics hỗ trợ cài đặt nhanh qua pip, có sẵn có model pre-trained, tài liệu rõ ràng và cú pháp lập trình

ngắn gọn.

- Tuỳ biến tốt: Cho phép dễ dàng fine-tune trên dataset riêng hỗ trợ export mô hình sang nhiều định dạng (ONNX, TorchScript, CoreML...) để triển khai linh hoạt trên nhiều nền tảng.

1.2.3 Lý do lựa chọn YOLOv8:

Việc triển khai các mô hình Trí tuệ nhân tạo (AI) trên các thiết bị di động, hay còn gọi là Edge AI, luôn là một thách thức lớn đối với các nhà phát triển do những giới hạn khắt khe về tài nguyên phần cứng như dung lượng pin, hiệu suất CPU, khả năng tính toán của GPU và bộ nhớ RAM. Sau quá trình nghiên cứu và so sánh thực nghiệm, nhóm thực hiện đã quyết định lựa chọn YOLOv8 làm kiến trúc cốt lõi cho hệ thống vì những lý do chiến lược sau đây:

Sự tối ưu vượt trội của biến thể Nano (YOLOv8n) YOLOv8 không chỉ là một mô hình đơn lẻ mà là một hệ sinh thái với nhiều kích thước khác nhau để phù hợp với mọi loại thiết bị. Trong đó, biến thể YOLOv8n (Nano) là lựa chọn tối ưu nhất cho đề án này nhờ các đặc điểm:

Kích thước siêu nhỏ: Đây là biến thể có số lượng tham số ít nhất và trọng lượng file mô hình nhẹ nhất trong "gia đình" YOLOv8. Điều này cho phép ứng dụng khởi chạy nhanh chóng và không chiếm dụng quá nhiều bộ nhớ lưu trữ của điện thoại.

Độ chính xác ấn tượng: Mặc dù được lược bỏ đáng kể về số lượng tầng mạng để tối ưu tốc độ, YOLOv8n vẫn duy trì được độ chính xác rất tốt với khoảng 37.3 mAP trên tập dữ liệu COCO. Đối với bài toán nhận diện trạng thái trái xoài (Xanh, Chín, Hồng), mức độ chính xác này là hoàn toàn đáp ứng được yêu cầu thực tế.

Khả năng xuất bản và tính tương thích cao (Exportability) Một trong những rào cản lớn nhất khi đưa AI lên di động là việc chuyển đổi định dạng mô hình từ môi trường huấn luyện (Python/PyTorch) sang môi trường thực thi (Android/Java/Kotlin). YOLOv8 đã giải quyết triệt để vấn đề này:

Hỗ trợ đa định dạng: Framework của Ultralytics hỗ trợ chuyển đổi mượt mà sang các định dạng tối ưu cho di động như TFLite (TensorFlow Lite) hay ONNX.

Tích hợp dễ dàng: Việc xuất mô hình kèm theo Metadata giúp quá trình tích hợp vào Android Studio trở nên đơn giản, giảm thiểu các lỗi logic khi xử lý đầu vào và đầu ra của mạng nơ-ron.

Sự cân bằng hoàn hảo giữa Tốc độ và Độ chính xác Trong bài toán nhận diện nông sản, tốc độ phản hồi là yếu tố tiên quyết để tạo ra trải nghiệm người dùng tốt.

Dẫn đầu về thời gian thực (Real-time): So với các đối thủ cùng phân khúc như SSD MobileNet, YOLOv8 hiện đang dẫn đầu về khả năng xử lý hình ảnh với độ trễ cực thấp (low latency).

Tối ưu hóa luồng dữ liệu: Nhờ kiến trúc C2f và cơ chế Anchor-free, mô hình có thể thực hiện suy luận (inference) ngay lập tức khi người dùng chụp ảnh, giúp việc phân loại chất lượng trái xoài diễn ra gần như tức thì.

Cộng đồng hỗ trợ và Hệ sinh thái công cụ mạnh mẽ Việc sử dụng YOLOv8 giúp nhóm tận dụng được sự hỗ trợ từ thư viện Ultralytics với tài liệu hướng dẫn chi tiết, các mô hình pre-trained phong phú và cộng đồng nhà phát triển đông đảo. Điều này giúp việc xử lý các lỗi phát sinh trong quá trình huấn luyện và triển khai trở nên hiệu quả hơn, đảm bảo tiến độ hoàn thành đồ án.

Tóm lại, YOLOv8 không chỉ là một công cụ nhận diện mạnh mẽ mà còn là giải pháp kỹ thuật toàn diện, giải quyết được sự mâu thuẫn giữa yêu cầu về độ chính xác cao và sự hạn chế về phần cứng của thiết bị di động.

1.2.4 Các biến thể của mô hình YOLOv8

Một trong những ưu điểm vượt trội giúp YOLOv8 trở thành tiêu chuẩn mới trong lĩnh vực thị giác máy tính là khả năng mở rộng (scalability) linh hoạt. Thay vì chỉ cung cấp một mô hình duy nhất, Ultralytics đã thiết kế một hệ sinh thái bao gồm 5 biến thể với kích thước và độ phức tạp khác nhau. Sự phân cấp này cho phép nhà phát triển lựa chọn mô hình tối ưu nhất tùy thuộc vào tài nguyên phần cứng và yêu cầu về độ chính xác của bài toán cụ thể.

Các biến thể của YOLOv8 được ký hiệu dựa trên kích thước từ nhỏ đến lớn: Nano (n), Small (s), Medium (m), Large (l), và XLarge (x).

YOLOv8n (Nano): Đây là biến thể nhỏ nhất và nhanh nhất trong hệ sinh thái YOLOv8. Với số lượng tham số cực thấp, mô hình này được tối ưu hóa hoàn toàn cho các thiết bị di động và thiết bị nhúng có tài nguyên hạn chế về CPU, RAM và dung lượng pin. Mặc dù có kích thước khiêm tốn, YOLOv8n vẫn đạt được độ chính xác 37.3 mAP trên tập dữ liệu COCO, đủ mạnh mẽ cho các bài toán nhận diện nông sản trong thực tế.

YOLOv8s (Small): Phiên bản này cân bằng giữa tốc độ và độ chính xác. Nó phù hợp cho các ứng dụng yêu cầu độ chính xác cao hơn Nano nhưng vẫn cần duy trì tốc độ xử lý nhanh trên các phần cứng tầm trung.

YOLOv8m (Medium): Biến thể này có số lượng tham số lớn hơn, mang lại khả năng trích xuất đặc trưng phức tạp tốt hơn. Nó thường được triển khai trên các hệ thống có hỗ trợ GPU rời để đảm bảo tốc độ phản hồi.

YOLOv8l (Large) và YOLOv8x (XLarge): Đây là hai biến thể cao cấp nhất, tập trung tối đa vào độ chính xác. Với hàng triệu tham số và cấu trúc mạng dày đặc, chúng thường được sử dụng trong các nhiệm vụ nghiên cứu hoặc các hệ thống giám sát server-side, nơi tài nguyên tính toán không phải là rào cản.

Dưới đây là bảng so sánh các thông số kỹ thuật cơ bản giúp làm nổi bật sự khác biệt giữa các biến thể:

Bảng 1.1 Bảng so sánh các thông số kỹ thuật của các phiên bản YOLOv8

Biến thể	Kích thước (pixels)	mAP val (50-95)	Tốc độ (CPU ONNX - ms)	Số tham số (M)	FLOPs (B)
YOLOv8n	640	37.3	80.4	3.2	8.7
YOLOv8s	640	44.9	128.4	11.2	28.6
YOLOv8m	640	50.2	234.7	25.9	78.9
YOLOv8l	640	52.9	375.2	43.7	165.2
YOLOv8x	640	53.9	479.1	68.2	257.8

Xuất phát từ mục tiêu triển khai AI trên thiết bị di động (Edge AI), đề án này ưu tiên yếu tố tốc độ xử lý và sự mượt mà trong trải nghiệm người dùng. Việc lựa chọn YOLOv8n mang lại những lợi ích thiết thực:

Tối ưu hóa độ trễ (Latency): Giúp ứng dụng có khả năng nhận diện trái xoài gần như tức thì khi người dùng chụp ảnh.

Tiết kiệm năng lượng: Giảm thiểu gánh nặng tính toán cho vi xử lý của điện thoại, giúp thiết bị không bị nóng và tiêu tốn pin quá nhanh khi sử dụng liên tục.

Dung lượng nhẹ: Tập mô hình sau khi chuyển đổi sang định dạng TFLite có kích thước nhỏ, dễ dàng tích hợp vào bộ cài đặt của ứng dụng Android Studio.

1.2.5 Kỹ thuật tăng cường dữ liệu (Data Augmentation) trong YOLOv8

Kỹ thuật tăng cường dữ liệu (Data Augmentation) đóng vai trò then chốt trong việc nâng cao hiệu năng của các mô hình học sâu nói chung và YOLOv8 nói riêng. Đây là quá trình can thiệp vào tập dữ liệu gốc để tạo ra các biến thể hình ảnh mới mà không làm thay đổi nhãn của đối tượng, giúp mô hình học được các đặc trưng mang tính tổng quát hơn và giảm thiểu rủi ro sai lệch khi áp dụng vào thực tế.

Trong phạm vi đề án này, các kỹ thuật tăng cường dữ liệu được thực hiện tập trung thông qua nền tảng Roboflow nhằm giải quyết các thách thức của bài toán nhận diện trái xoài trong môi trường tự nhiên.

Mục đích của việc tăng cường dữ liệu:

Tránh hiện tượng học vẹt (Overfitting): Việc chỉ huấn luyện trên một số lượng ảnh cố định có thể khiến mô hình ghi nhớ máy móc các chi tiết nhiễu thay vì học các đặc trưng thực sự của trái xoài. Tăng cường dữ liệu giúp làm giàu bộ dữ liệu, bắt buộc mô hình phải học cách nhận diện đối tượng dựa trên các đặc điểm cốt lõi.

Tăng tính tổng quát hóa: Giúp mô hình có khả năng nhận diện tốt trái xoài trong nhiều điều kiện khác nhau như góc chụp, khoảng cách và môi trường ánh sáng thay đổi.

Các kỹ thuật tăng cường cụ thể được áp dụng: Dựa trên đặc điểm thị giác của trái xoài và yêu cầu thực tế tại vườn, đề án đã áp dụng các kỹ thuật sau:

Xoay và lật hình ảnh (Rotate & Flip): Kỹ thuật này tạo ra các hình ảnh trái xoài ở nhiều hướng nhìn khác nhau. Điều này cực kỳ quan trọng vì trong thực tế, người dùng có thể chụp trái xoài từ bất kỳ góc độ nào hoặc trái xoài có thể nằm ở các vị trí ngẫu nhiên trên cành cây.

Điều chỉnh độ sáng (Brightness Adjustment): Bằng cách thay đổi cường độ sáng của ảnh, mô hình được rèn luyện để nhận diện trái xoài ổn định ngay cả trong điều kiện ánh sáng mặt trời rực rỡ, dưới bóng râm hoặc trong môi trường thiếu sáng.

Tự động điều chỉnh độ tương phản (Auto-Adjust Contrast): Đề án sử dụng kỹ thuật Adaptive Equalization để làm rõ các chi tiết bề mặt. Việc này giúp mô hình nhận diện chính xác hơn các dấu hiệu bất thường như đốm đen, vết thối trên lớp vỏ của những trái xoài hỏng (unhealthy).

Thay đổi kích thước và định dạng (Resize): Toàn bộ ảnh được đưa về chuẩn đầu vào 640x640 của YOLOv8. Quá trình này đảm bảo tính nhất quán về kích thước đối tượng, giúp mô hình học được cách trích xuất đặc trưng hiệu quả nhất.

Hiệu quả mang lại đối với bài toán nhận diện: Việc áp dụng đồng bộ các kỹ thuật tăng cường dữ liệu đã mang lại những kết quả thực nghiệm ấn tượng cho đề án:

Mô hình đạt độ chính xác trung bình (mAP50) lên tới 93.3%.

Khả năng nhận diện lớp "Xoài hồng" đạt độ chính xác xấp xỉ 98%, cho thấy mô hình đã học được rất tốt các chi tiết bệnh lý phức tạp trên vỏ quả nhờ vào việc tối ưu hóa hình ảnh.

Ứng dụng hoạt động ổn định và có khả năng phản hồi nhanh chóng khi người dùng chụp ảnh thực tế.

1.2.6 Các chỉ số đánh giá hiệu năng mô hình

Trong các bài toán phát hiện đối tượng (Object Detection), việc đánh giá hiệu năng của mô hình không chỉ dựa trên cảm quan mà cần thông qua các chỉ số đo lường định lượng cụ thể. Các chỉ số này giúp xác định khả năng phân loại đúng nhãn lớp và khả năng định vị chính xác khung bao của mô hình. Trong đề án này, hiệu năng của YOLOv8 được đánh giá dựa trên các tiêu chí cốt lõi sau:

Precision (Độ chính xác)

Precision đại diện cho khả năng dự đoán đúng của mô hình trong số tất cả các trường hợp mà mô hình đã gắn nhãn là đối tượng. Chỉ số này trả lời cho câu hỏi: "Trong số các trái xoài mà mô hình nhận diện được, có bao nhiêu phần trăm là thực sự đúng?".

Công thức tính:

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

Trong đó:

TP (True Positive): Số lượng đối tượng được mô hình dự đoán đúng (ví dụ: mô hình báo là xoài chín và thực tế là xoài chín).

FP (False Positive): Số lượng đối tượng bị dự đoán sai (ví dụ: mô hình báo là xoài hồng nhưng thực tế là xoài xanh).

Chỉ số Precision cao cho thấy mô hình rất "cẩn trọng" và có độ tin cậy lớn trong mỗi dự đoán đưa ra

Recall (Độ bao phủ / Độ triệu hồi)

Recall đo lường khả năng tìm thấy toàn bộ các đối tượng thực tế có trong hình ảnh. Chỉ số này trả lời cho câu hỏi: "Trong số tất cả các trái xoài thực sự tồn tại trong ảnh, mô hình đã phát hiện được bao nhiêu quả?".

Công thức tính:

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

Trong đó:

FN (False Negative): Số lượng đối tượng mà mô hình đã bỏ sót (ví dụ: thực tế có xoài hỏng nhưng mô hình không phát hiện được).

Recall cao cho thấy mô hình có khả năng bắt được hầu hết các đối tượng cần tìm, giảm thiểu tình trạng bỏ sót nông sản trong quá trình kiểm định.

mAP (mean Average Precision)

mAP là chỉ số quan trọng và toàn diện nhất để đánh giá các mô hình Object Detection như YOLOv8. Nó là giá trị trung bình của độ chính xác trung bình (Average Precision - AP) trên tất cả các lớp đối tượng.

mAP@0.5: Là chỉ số mAP được tính toán khi ngưỡng giao nhau (IoU - Intersection over Union) giữa khung bao dự đoán và khung bao thực tế đạt từ 0.5 trở lên. Chỉ số này đánh giá khả năng nhận diện đối tượng ở mức độ chấp nhận được.

mAP@0.5:0.95: Là chỉ số trung bình của mAP tính trên các ngưỡng IoU biến thiên từ 0.5 đến 0.95 với bước nhảy 0.05. Đây là chỉ số khắt khe nhất, yêu cầu mô hình không chỉ nhận diện đúng nhãn mà còn phải định vị khung bao cực kỳ chuẩn xác.

Chỉ số giao nhau IoU (Intersection over Union)

IoU là một khái niệm hỗ trợ nhưng không thể thiếu để tính toán Precision và Recall.

Công thức tính:

$$IoU = \frac{Area(B_{pred} \cap B_{gt})}{Area(B_{pred} \cup B_{gt})}$$

Ý nghĩa thực tiễn đối với đề án

Việc theo dõi chặt chẽ các chỉ số này trong quá trình huấn luyện giúp nhóm thực hiện có thể:

Đánh giá khách quan năng lực của mô hình YOLOv8 trên tập dữ liệu trái xoài Việt Nam.

Cân bằng giữa việc tránh dự đoán nhầm (tăng Precision) và tránh bỏ sót (tăng Recall).

Đảm bảo mô hình sau khi tích hợp vào Android Studio có thể hoạt động ổn định với chỉ số mAP cao, đáp ứng được tiêu chuẩn của một ứng dụng hỗ trợ nông nghiệp thông minh

1.2.7 Cơ chế tối ưu hóa cho thiết bị di động (TFLite Export)

Việc triển khai các mô hình học sâu trên thiết bị di động luôn đối mặt với những thách thức lớn về giới hạn phần cứng như dung lượng pin, hiệu năng CPU và bộ nhớ RAM hạn chế. Để giải quyết bài toán này, quy trình tối ưu hóa mô hình từ định dạng huấn luyện gốc (PyTorch) sang định dạng thực thi trên di động (TensorFlow Lite) là một bước bắt buộc và tối quan trọng trong đề án.

Khái niệm về TensorFlow Lite (TFLite)

TensorFlow Lite là một framework máy học mã nguồn mở của Google, được thiết kế chuyên biệt để chạy các mô hình trên các thiết bị có tài nguyên thấp như điện thoại thông minh hoặc các thiết bị nhúng. TFLite không chỉ giúp giảm kích thước tệp mô hình mà còn tối ưu hóa các phép toán toán học để tận dụng tối đa sức mạnh của các bộ xử lý di động.

Quy trình xuất mô hình và tích hợp Metadata

Quá trình chuyển đổi mô hình trong đề án được thực hiện thông qua thư viện Ultralytics trên môi trường Google Colab.

Chuyển đổi định dạng: Mô hình sau khi huấn luyện đạt độ chính xác cao nhất (file best.pt) được chuyển đổi sang định dạng .tflite.

Tích hợp Metadata: Điểm đặc biệt trong quy trình này là việc nhúng Metadata (siêu dữ liệu) trực tiếp vào file mô hình.

Metadata chứa thông tin về tên các nhãn lớp (ripe-mango, unripe-mango, unhealthy) và các tham số cấu hình đầu vào/đầu ra. Điều này giúp thư viện TFLite trong Android Studio có thể tự động hiểu cấu trúc mô hình mà không cần lập trình viên phải định nghĩa lại thủ công bằng mã nguồn, giúp giảm thiểu sai sót kỹ thuật.

Các cơ chế tối ưu hóa cốt lõi

Tối ưu hóa kích thước: Tập mô hình sau khi xuất được giảm thiểu đáng kể về dung lượng, giúp ứng dụng Android nhẹ hơn và dễ dàng tải xuống.

Giảm độ trễ (Latency): TFLite tối ưu hóa luồng dữ liệu để mô hình có khả năng phản hồi ngay lập tức khi người dùng chụp ảnh trái xoài, mang lại trải nghiệm mượt mà.

Hỗ trợ tăng tốc phần cứng: Định dạng TFLite cho phép ứng dụng tận dụng các bộ tăng tốc phần cứng như GPU Delegate để đạt tốc độ khung hình cao (FPS) trong các phiên bản phát triển tương lai.

Hiệu quả thực tế trong đồ án

Nhờ vào cơ chế tối ưu hóa này, ứng dụng đã đạt được những kết quả thực nghiệm khả quan:

Xử lý Offline: Mô hình có thể chạy trực tiếp trên thiết bị mà không cần kết nối Internet, đảm bảo tốc độ và tính riêng tư.

Tiết kiệm tài nguyên: Giảm bớt gánh nặng tính toán cho thiết bị, giúp hạn chế tình trạng nóng máy và tiêu hao pin nhanh khi sử dụng liên tục.

Tích hợp thuận tiện: File TFLite được đưa trực tiếp vào thư mục assets của Android Studio, cho phép nạp mô hình và thực hiện suy luận (inference) một cách nhanh chóng.

Tóm lại, cơ chế TFLite Export không chỉ đơn thuần là một bước chuyển đổi định dạng, mà là một quy trình kỹ thuật giúp thu hẹp khoảng cách giữa một mô hình AI nặng nề trên máy chủ và một ứng dụng di động tinh gọn, hiệu quả trên tay người nông dân.

1.3 Nền tảng phát triển ứng dụng

1.3.1 Hệ điều hành Android và Android Studio

Hệ điều hành Android là một nền tảng phần mềm mã nguồn mở dựa trên nhân Linux (Linux kernel), được thiết kế chủ yếu cho các thiết bị di động có màn hình cảm ứng như điện thoại thông minh và máy tính bảng. Kể từ khi ra mắt,

Android đã nhanh chóng vươn lên trở thành hệ điều hành di động phổ biến nhất, nắm giữ thị phần lớn nhất thế giới hiện nay.

Sự thành công của Android trong nghiên cứu và phát triển công nghệ bắt nguồn từ các đặc điểm sau:

Mã nguồn mở và tính tùy biến: Với bản chất là mã nguồn mở, Android cho phép các nhà phát triển can thiệp sâu vào hệ thống, tối ưu hóa các lớp phần cứng và phần mềm để đáp ứng các yêu cầu khắt khe của ứng dụng.

Hệ sinh thái đa dạng: Khả năng tương thích với nhiều cấu hình phần cứng khác nhau giúp Android trở thành môi trường lý tưởng để thử nghiệm các công nghệ mới, đặc biệt là các ứng dụng tích hợp trí tuệ nhân tạo (AI) và thị giác máy tính.

Hỗ trợ tăng tốc phần cứng: Android cung cấp các bộ API như Neural Networks API (NNAPI), cho phép các mô hình học sâu (Deep Learning) có thể tận dụng sức mạnh xử lý của GPU hoặc các bộ xử lý thần kinh chuyên dụng (NPU) trên thiết bị di động, giúp tối ưu hóa tốc độ nhận diện đối tượng.

Môi trường phát triển Android Studio:

Android Studio là Môi trường phát triển tích hợp (IDE) chính thức và mạnh mẽ nhất do Google cung cấp, được xây dựng dựa trên nền tảng IntelliJ IDEA nhằm phục vụ riêng cho việc phát triển ứng dụng Android. Trong phạm vi đồ án nghiên cứu về nhận diện trái xoài, Android Studio đóng vai trò là "trung tâm điều khiển", nơi tích hợp tất cả các thành phần từ giao diện người dùng, logic xử lý đến mô hình AI.

Các công cụ và tính năng thiết yếu mà Android Studio cung cấp bao gồm:

Hệ thống xây dựng Gradle: Gradle là một công cụ tự động hóa việc xây dựng (build) ứng dụng linh hoạt, giúp quản lý các thư viện phụ thuộc (dependencies) một cách hiệu quả. Trong dự án này, Gradle đóng vai trò quan trọng trong việc tích hợp các thư viện Machine Learning như tensorflow-lite và các SDK từ bên thứ ba như Firebase hay Cloudinary.

Trình mô phỏng (Android Emulator): Cho phép kiểm thử ứng dụng trên nhiều cấu hình thiết bị và phiên bản hệ điều hành khác nhau ngay trên máy tính mà không cần thiết bị thật, giúp tiết kiệm thời gian và tối ưu hóa giao diện.

Công cụ Debug và Profiler: Android Studio cung cấp các công cụ theo dõi hiệu năng hệ thống chuyên sâu. Điều này cực kỳ quan trọng khi triển khai mô hình YOLOv8 dưới dạng TFLite, vì nhà phát triển có thể theo dõi lượng RAM tiêu thụ, mức sử dụng CPU/GPU và độ trễ (latency)

khi mô hình thực hiện suy luận, từ đó đưa ra các giải pháp tối ưu hóa phần mềm.

Hỗ trợ ngôn ngữ Kotlin: Android Studio ưu tiên hỗ trợ Kotlin – ngôn ngữ lập trình hiện đại, an toàn và ngắn gọn, giúp việc viết code xử lý hình ảnh và tích hợp máy học trở nên hiệu quả và ít lỗi hơn.

1.3.2 Ngôn ngữ lập trình (Java/Kotlin)

Trong quá trình phát triển ứng dụng di động cho đề tài nhận diện trái xoài, nhóm thực hiện đã lựa chọn Kotlin làm ngôn ngữ lập trình chính. Đây là một ngôn ngữ lập trình hiện đại, tĩnh (statically typed), chạy trên máy ảo Java (JVM) và đã được Google chính thức công nhận là ngôn ngữ ưu tiên hàng đầu (Kotlin-first) cho phát triển ứng dụng Android Native kể từ năm 2019.

Việc lựa chọn Kotlin mang lại những lợi thế vượt trội, giúp tối ưu hóa quá trình xây dựng hệ thống nhận diện dựa trên AI:

Tính hiện đại và ngắn gọn: Kotlin loại bỏ các cú pháp rườm rà của Java, giúp mã nguồn trở nên sạch sẽ, dễ đọc và dễ bảo trì hơn. Điều này đặc biệt quan trọng trong các tác vụ xử lý ảnh và điều phối luồng dữ liệu từ camera đến mô hình học sâu, nơi logic lập trình cần sự chuẩn xác và minh bạch.

An toàn với lỗi Null (Null Safety): Một trong những nguyên nhân phổ biến gây lỗi ứng dụng Android là NullPointerException. Kotlin giải quyết vấn đề này bằng hệ thống kiểu dữ liệu an toàn, giúp ngăn chặn lỗi ngay từ bước biên dịch. Trong đồ án, tính năng này đảm bảo ứng dụng hoạt động ổn định khi xử lý các đối tượng hình ảnh hoặc phản hồi từ server mà không bị dừng đột ngột.

Hỗ trợ mạnh mẽ các thư viện máy học (Machine Learning): Kotlin cung cấp khả năng tích hợp và xây dựng hoàn hảo với thư viện `org.tensorflow:tensorflow-lite`, cho phép nạp và thực thi mô hình YOLOv8 một cách mượt mà trên thiết bị di động. Nhờ vào khả năng xử lý bất đồng bộ (Coroutines), ứng dụng có thể thực hiện suy luận AI (inference) ở luồng nền, giúp giao diện người dùng không bị giật lag khi nhận diện trái xoài.

Tương thích tối ưu với hệ sinh thái Google và bên thứ ba: Kotlin có khả năng tương tác 100% với Java, giúp dễ dàng tích hợp các bộ công cụ phát triển phần mềm (SDK) hiện đại như Firebase Authentication để quản lý người dùng và Cloudinary API để lưu trữ hình ảnh trên đám mây.

Các thư viện này đều cung cấp tài liệu và hỗ trợ tốt nhất cho Kotlin, giúp nhóm rút ngắn thời gian triển khai logic backend.

Cộng đồng hỗ trợ và hệ sinh thái phong phú: Với vị thế là ngôn ngữ nền tảng của Android, Kotlin sở hữu một cộng đồng nhà phát triển khổng lồ trên toàn cầu. Điều này giúp nhóm thực hiện dễ dàng tiếp cận các giải pháp cho các vấn đề kỹ thuật phát sinh, từ việc tối ưu hóa hiệu năng mô hình TFLite đến việc xử lý các lỗi logic phức tạp trong quá trình phát triển ứng dụng thực tế.

Việc sử dụng Kotlin không chỉ đảm bảo ứng dụng đạt được hiệu suất cao nhất mà còn giúp việc tích hợp các công nghệ phức tạp như YOLOv8 trở nên hiệu quả, tạo nền tảng vững chắc cho một ứng dụng nhận diện nông sản thông minh và ổn định.

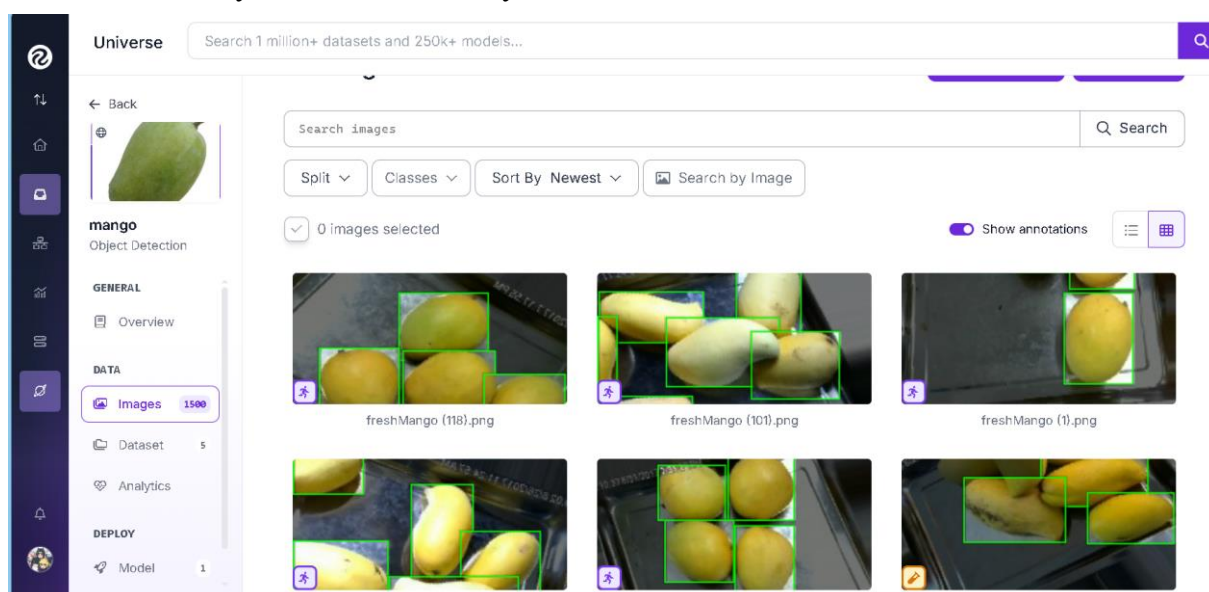
1.4 Các công cụ và dịch vụ tích hợp

1.4.1 Roboflow: Quản lý và tiền xử lý dữ liệu ảnh

Roboflow là một nền tảng toàn diện hỗ trợ quy trình MLOps (Machine Learning Operations), được thiết kế chuyên biệt để quản lý vòng đời của các dự án thị giác máy tính. Trong đồ án này, Roboflow đóng vai trò là "cầu nối" quan trọng giữa dữ liệu thô và mô hình huấn luyện, giúp tối ưu hóa công tác chuẩn bị dữ liệu cho YOLOv8.

Quy trình xây dựng bộ dữ liệu nhận diện trái xoài trên Roboflow được thực hiện theo các bước lý thuyết và thực tiễn sau:

Thu thập và Tải lên (Upload): Toàn bộ hình ảnh trái xoài từ các nguồn thực tế (vườn chụp) và Internet được tập trung về một kho lưu trữ duy nhất trên đám mây.



Hình 1.2: Dữ liệu ảnh tải lên Roboflow

Gán nhãn đối tượng (Annotation):

Sử dụng công cụ vẽ khung bao (Bounding box) trực quan để xác định vị trí của từng trái xoài trong ảnh.

Gán nhãn cho 3 lớp đối tượng cụ thể: ripe-mango (xoài chín), unripe-mango (xoài xanh), và unhealthy (xoài hỏng).

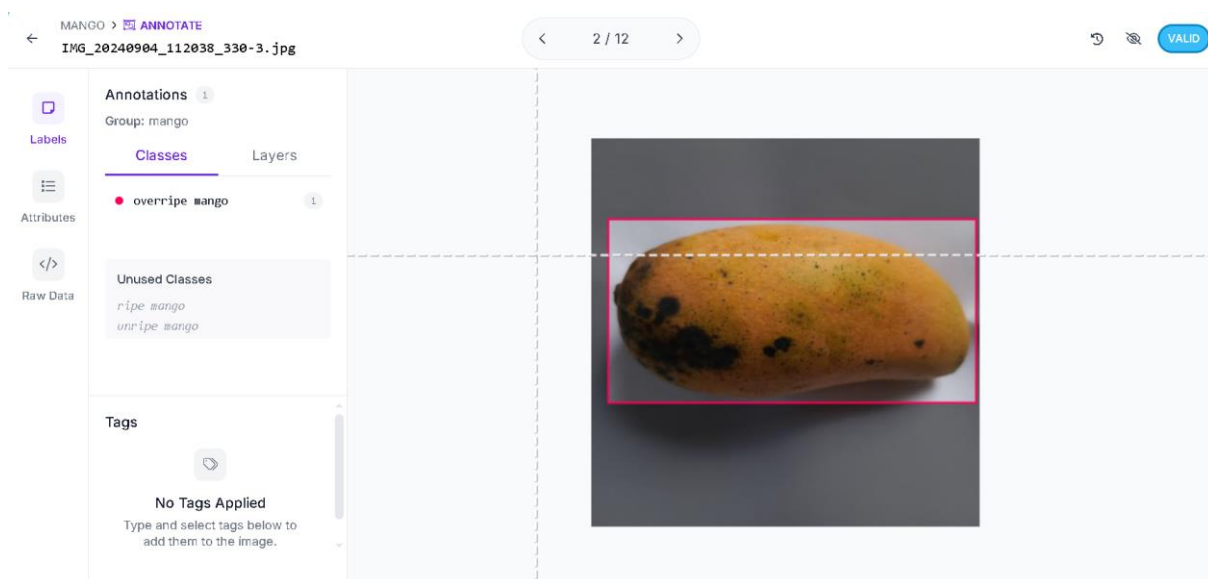
Quá trình này được thực hiện thủ công để đảm bảo độ chính xác cao nhất cho tập dữ liệu huấn luyện.

Tiền xử lý dữ liệu (Preprocessing):

Auto-Orient: Tự động điều chỉnh hướng ảnh để đảm bảo tính nhất quán.

Resize: Thay đổi kích thước toàn bộ ảnh về chuẩn đầu vào 640 x 640 của YOLOv8 nhằm tối ưu hóa tốc độ tính toán.

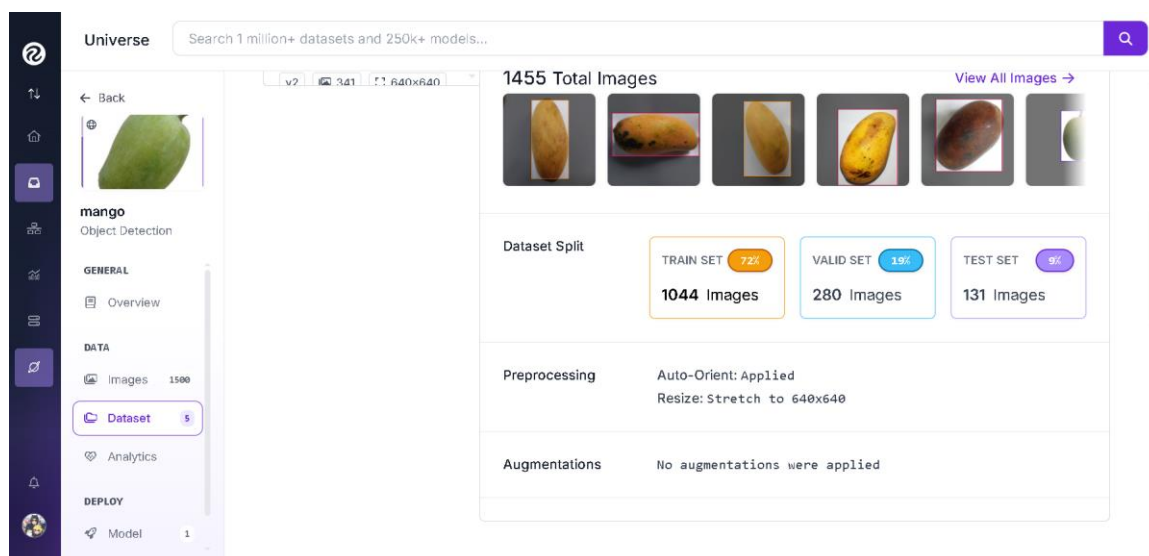
Contrast Adjustment: Sử dụng Adaptive Equalization để điều chỉnh độ tương phản, giúp các đặc trưng của trái xoài nổi bật hơn.



Hình 1.3: Gán nhãn đối tượng trên Roboflow

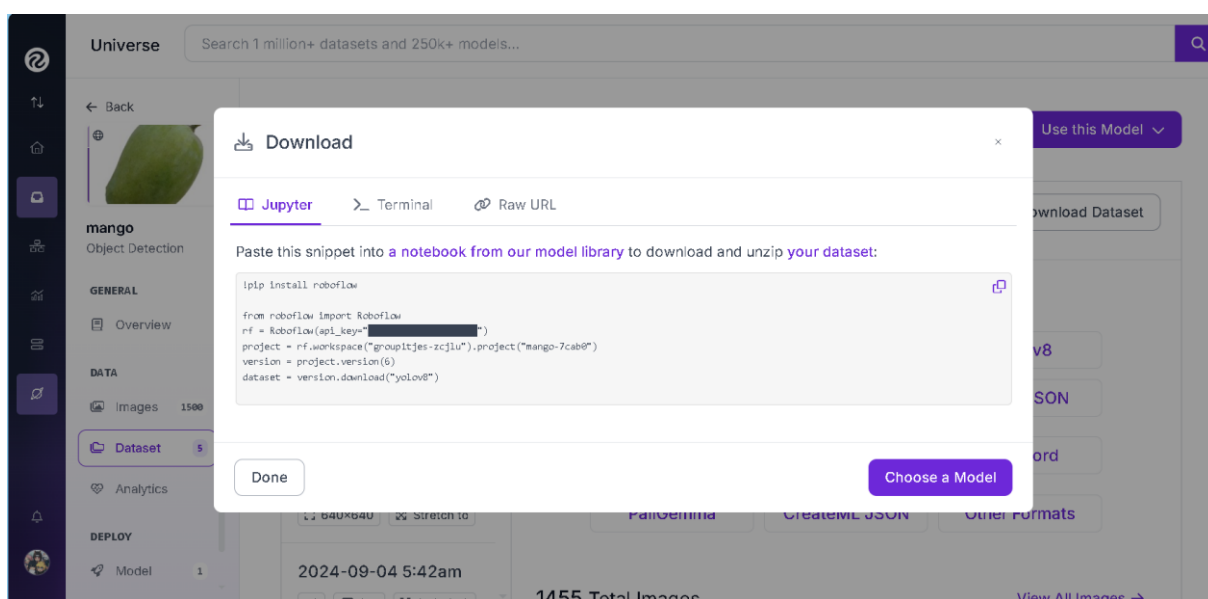
Tăng cường dữ liệu (Augmentation): Tạo ra các biến thể mới từ dữ liệu gốc thông qua các phép biến đổi như xoay, lật, hay thay đổi độ sáng. Bước này giúp làm giàu bộ dữ liệu, giúp mô hình học được các đặc trưng đa dạng và tránh hiện tượng học vẹt (overfitting).

Chia tập dữ liệu (Dataset Splitting): Hệ thống tự động phân chia dữ liệu theo tỷ lệ tối ưu: 72% cho tập huấn luyện (Train), 19% cho tập kiểm thử (Validation) và 9% cho tập đánh giá cuối cùng (Test).



Hình 1.1: Tăng cường và chia tập dữ liệu

Xuất dữ liệu (Export): Chuyển đổi bộ dữ liệu sang định dạng YOLOv8 để sẵn sàng đưa vào huấn luyện trên môi trường Google Colab.



Hình 1.5: Xuất dữ liệu để huấn luyện môi trường Google Colab

Ưu điểm của Roboflow

Quản lý tập trung: Giúp lưu trữ và tổ chức tập dữ liệu hình ảnh trái xoài một cách khoa học, dễ dàng truy xuất.

Công cụ gán nhãn thông minh: Giao diện trực quan giúp việc vẽ bounding box diễn ra nhanh chóng, hỗ trợ quản lý nhiều nhãn lớp phức tạp.

Tích hợp sẵn các kỹ thuật Augmentation: Giúp tăng quy mô bộ dữ liệu mà không cần thu thập thêm ảnh mới, từ đó nâng cao độ chính xác của mô hình.

Hỗ trợ đa định dạng: Khả năng xuất dữ liệu tương thích hoàn toàn với Ultralytics YOLOv8 và nhiều framework AI khác.

Tự động hóa quy trình chia dữ liệu: Việc chia ngẫu nhiên tập Train/Val/Test giúp tăng khả năng tổng quát hóa cho mô hình.

Nhược điểm của Roboflow

Phụ thuộc vào kết nối Internet: Vì là nền tảng Cloud-based, mọi thao tác từ tải ảnh đến gán nhãn đều yêu cầu kết nối mạng ổn định.

Giới hạn về sự đa dạng chủng loại: Mô hình hoạt động tốt nhất trên dữ liệu có sẵn, nhưng nếu gặp các giống xoài quá lạ hoặc hình dáng khác biệt so với tập huấn luyện, hệ thống có thể nhận diện nhầm lẫn (ví dụ: nhầm xoài xanh với đu đủ xanh).

Rào cản về chi phí: Các tính năng nâng cao hoặc dung lượng lưu trữ lớn thường bị giới hạn trong các gói trả phí.

Yêu cầu thao tác thủ công: Mặc dù hỗ trợ nhiều công cụ, nhưng để có nhãn chuẩn xác nhất cho bài toán chất lượng nông sản, người dùng vẫn phải bỏ ra nhiều thời gian để gán nhãn thủ công.

1.4.2 Google Colab: Môi trường huấn luyện mô hình đám mây

Trong kỷ nguyên của Trí tuệ nhân tạo, việc huấn luyện các mô hình học sâu (Deep Learning) như YOLOv8 đòi hỏi năng lực tính toán cực kỳ lớn, đặc biệt là các đơn vị xử lý đồ họa (GPU) chuyên dụng. Do các máy tính cá nhân thông thường thường bị hạn chế về cấu hình phần cứng, đề án đã lựa chọn Google Colab (Google Colaboratory) làm nền tảng thực hiện quá trình huấn luyện mô hình.

Google Colab là một dịch vụ lưu trữ Notebook dựa trên điện toán đám mây do Google cung cấp. Nền tảng này cho phép người dùng viết và thực thi mã Python trực tiếp trên trình duyệt web mà không cần thực hiện các bước cài đặt phức tạp. Điểm mấu chốt khiến Colab trở nên lý tưởng cho các dự án AI là việc cung cấp quyền truy cập miễn phí vào các GPU hiệu năng cao, điển hình là dòng Nvidia Tesla T4. Các GPU này giúp tăng tốc độ tính toán ma trận trong mạng nơ-ron lên gấp hàng chục lần so với CPU truyền thống, từ đó rút ngắn đáng kể thời gian huấn luyện mô hình nhận diện trái xoài.

Trong phạm vi đề tài, Google Colab đóng vai trò là "nhà máy xử lý" chính cho các tác vụ nặng nhất:

Thiết lập môi trường: Colab hỗ trợ sẵn các thư viện khoa học dữ liệu và học máy hàng đầu như PyTorch, TensorFlow và đặc biệt là thư viện Ultralytics – framework chính để triển khai YOLOv8.

Huấn luyện mô hình (Training): Thực hiện quá trình học các đặc trưng hình ảnh của trái xoài (xoài xanh, chín, hồng) từ bộ dữ liệu đã được gán nhãn.

Đánh giá và Tinh chỉnh (Validation): Sau mỗi chu kỳ huấn luyện (epoch), hệ thống tự động đánh giá mô hình trên tập Validation để theo dõi các chỉ số quan trọng như Loss (hàm mất mát) và mAP (độ chính xác trung bình).

Xuất mô hình: Sau khi đạt được độ chính xác mong muốn, Colab cho phép chuyển đổi và tải xuống mô hình dưới định dạng .tflite để tích hợp vào ứng dụng Android Studio.

Ưu điểm của Google Colab

Miễn phí tài nguyên mạnh mẽ: Cung cấp quyền truy cập GPU và RAM lớn mà không tốn chi phí đầu tư phần cứng.

Tính linh hoạt và cộng tác: Vì hoạt động trên nền web, nhóm thực hiện có thể dễ dàng chia sẻ file Notebook, làm việc nhóm và lưu trữ trực tiếp mã nguồn trên Google Drive.

Môi trường chuẩn hóa: Loại bỏ hoàn toàn các lỗi xung đột phiên bản thư viện thường gặp khi cài đặt Python trên máy tính cục bộ.

Khả năng tương thích tốt: Kết nối mượt mà với Roboflow để tải tập dữ liệu về môi trường huấn luyện chỉ qua một dòng lệnh API.

Hạn chế

Phụ thuộc vào Internet: Cần kết nối mạng ổn định và liên tục trong suốt quá trình huấn luyện dài hơi.

Giới hạn thời gian phiên (Session): Phiên làm việc miễn phí thường bị ngắt sau 12 giờ hoặc khi không có tương tác, đòi hỏi người dùng phải có chiến thuật lưu trữ các checkpoint (điểm kiểm soát) thường xuyên.

Tài nguyên không cố định: Loại GPU được cấp (K80, P100 hay T4) phụ thuộc vào sự điều phối ngẫu nhiên của Google tại thời điểm truy cập.

1.4.3 Firebase Authentication: Giải pháp xác thực người dùng

Firebase là một nền tảng phát triển ứng dụng di động và web mạnh mẽ dựa trên kiến trúc Backend-as-a-Service (BaaS) do Google cung cấp. Trong đó

án này, nhóm đã tích hợp module Firebase Authentication để giải quyết bài toán quản lý danh tính người dùng một cách an toàn và hiệu quả.

Thay vì phải tự xây dựng và vận hành một hệ thống server backend phức tạp để lưu trữ mật khẩu và quản lý phiên đăng nhập (session), Firebase Authentication cung cấp một giải pháp "trọn gói" với các vai trò then chốt:

Xử lý quy trình đăng ký và đăng nhập: Cung cấp các bộ API sẵn có để thực hiện việc xác thực tài khoản thông qua Email và Mật khẩu, giúp rút ngắn đáng kể thời gian phát triển ứng dụng.

Đảm bảo an toàn thông tin: Firebase sử dụng các tiêu chuẩn bảo mật hàng đầu của Google để lưu trữ và mã hóa thông tin tài khoản, giúp bảo vệ dữ liệu người dùng khỏi các nguy cơ tấn công mạng mà không cần nhóm phải can thiệp sâu vào kỹ thuật bảo mật máy chủ.

Quản lý phiên làm việc (Session Management): Sau khi đăng nhập thành công, hệ thống sẽ tự động duy trì trạng thái đăng nhập trên thiết bị. Điều này cho phép ứng dụng xác định chính xác danh tính của từng người dùng khi họ thực hiện các thao tác như tải ảnh lên hoặc xem lại lịch sử nhận diện.

Quy trình xác thực trong ứng dụng được thực hiện như sau:

Gửi yêu cầu: Người dùng nhập thông tin (Email/Password) trên giao diện ứng dụng.

Xác thực phía Server: Firebase Authentication tiếp nhận và đối soát thông tin với cơ sở dữ liệu trên Cloud của Google.

Phản hồi mã Token: Nếu thông tin hợp lệ, Firebase sẽ trả về một mã định danh (Unique ID - UID). Mã UID này là chìa khóa để ứng dụng liên kết dữ liệu lịch sử nhận diện của người dùng đó với kho lưu trữ hình ảnh trên Cloudinary.

Ưu điểm nổi bật

Tính sẵn sàng cao: Là dịch vụ đám mây nên hệ thống xác thực luôn hoạt động ổn định, có khả năng mở rộng tốt khi số lượng người dùng tăng lên.

Tích hợp dễ dàng: Hỗ trợ thư viện (SDK) tối ưu cho ngôn ngữ Kotlin và Android Studio, cho phép triển khai logic xác thực chỉ với vài dòng mã nguồn.

Hệ sinh thái đồng bộ: Việc sử dụng dịch vụ của Google giúp ứng dụng dễ dàng mở rộng thêm các tính năng khác trong tương lai như lưu

trữ cơ sở dữ liệu (Firestore) hay phân tích hành vi người dùng (Google Analytics).

Hạn chế

Phụ thuộc vào bên thứ ba: Mọi tính năng xác thực đều phụ thuộc hoàn toàn vào dịch vụ của Google; nếu máy chủ Firebase gặp sự cố, tính năng đăng nhập của ứng dụng sẽ bị ảnh hưởng.

Yêu cầu kết nối Internet: Người dùng bắt buộc phải có mạng để thực hiện các thao tác liên quan đến tài khoản.

1.4.4 Cloudinary: Dịch vụ lưu trữ và quản lý hình ảnh đám mây

Để tối ưu hóa dung lượng lưu trữ trên điện thoại và cho phép truy xuất dữ liệu linh hoạt, đồ án đã tích hợp Cloudinary như một giải pháp quản trị phương tiện truyền thông trên nền tảng đám mây (Cloud-based image management). Đây là một nền tảng đóng vai trò quan trọng trong việc xây dựng tính năng lưu trữ lịch sử nhận diện chuyên nghiệp cho người dùng.

Trong quy trình hoạt động của ứng dụng, Cloudinary không chỉ đơn thuần là một kho lưu trữ mà còn là thành phần tối ưu hóa dữ liệu đầu ra:

Tối ưu hóa bộ nhớ thiết bị: Thay vì lưu trữ hàng loạt ảnh độ phân giải cao trực tiếp trong bộ nhớ máy (gây nặng máy và tốn tài nguyên), ứng dụng chỉ thực hiện đẩy ảnh lên đám mây.

Quản lý lịch sử nhận diện: Ứng dụng sử dụng Cloudinary API để tải các hình ảnh trái xoài lên máy chủ ngay sau khi quá trình nhận diện hoàn tất.

Cung cấp truy cập đa nền tảng: Cloudinary trả về một đường dẫn (URL) duy nhất cho mỗi ảnh. URL này được lưu vào cơ sở dữ liệu, giúp ứng dụng có thể hiển thị lại hình ảnh trong phần lịch sử một cách nhanh chóng ở bất kỳ thời điểm nào.

Quy trình tích hợp Cloudinary trong đồ án được thực hiện theo các bước sau:

Chụp/Chọn ảnh: Người dùng thực hiện chụp ảnh trái xoài hoặc chọn từ thư viện.

Nhận diện và Tải lên: Sau khi mô hình YOLOv8 hoàn tất việc phân loại, ảnh sẽ được nén và đẩy lên Cloudinary thông qua SDK tích hợp trong ứng dụng Android.

Lưu trữ liên kết: Ứng dụng nhận URL từ Cloudinary và lưu trữ URL này cùng với các thông tin nhãn (Xanh, Chín, Hỏng) và thời gian nhận diện vào hệ thống quản lý.

Hiện thị: Khi người dùng truy cập mục "Lịch sử", ứng dụng chỉ cần tải ảnh thông qua URL thay vì đọc tệp tin cục bộ, giúp giao diện hoạt động mượt mà hơn.

Ưu điểm nổi bật

Tốc độ phản hồi nhanh: Nhờ hệ thống mạng phân phối nội dung (CDN) toàn cầu, việc tải ảnh từ URL của Cloudinary có độ trễ cực thấp.

Dễ dàng quản lý: Giao diện quản trị của Cloudinary cho phép nhóm thực hiện theo dõi, thống kê và quản lý toàn bộ tệp tin hình ảnh đã được người dùng tải lên một cách trực quan.

Khả năng mở rộng: Dịch vụ này cho phép mở rộng cơ sở dữ liệu hình ảnh trong tương lai, phục vụ cho việc tái huấn luyện (re-train) mô hình để nâng cao độ chính xác.

Hạn chế

Phụ thuộc vào băng thông: Tốc độ tải và hiển thị ảnh phụ thuộc hoàn toàn vào kết nối Internet của thiết bị người dùng.

Giới hạn gói miễn phí: Giống như các dịch vụ SaaS khác, Cloudinary giới hạn dung lượng lưu trữ và số lượng yêu cầu (request) hàng tháng đối với tài khoản miễn phí.

CHƯƠNG 2: CÀI ĐẶT VÀ TRIỂN KHAI

2.1 Quá trình huấn luyện mô hình

2.1.1 Bộ dữ liệu trái xoài

Trong các bài toán về Thị giác máy tính (Computer Vision), chất lượng và độ đa dạng của bộ dữ liệu (Dataset) đóng vai trò quyết định đến hiệu suất và khả năng tổng quát hóa của mô hình học sâu. Vì vậy, việc xây dựng bộ dữ liệu trái xoài đã được nhóm thực hiện một cách tỉ mỉ, đảm bảo mô hình có thể học được các đặc trưng phong phú từ thực tế.

Nguồn thu thập và quy mô dữ liệu Bộ dữ liệu được tổng hợp từ hai nguồn chính nhằm đảm bảo tính khách quan và thực tiễn:

Dữ liệu thực tế: Một phần lớn hình ảnh được sinh viên trực tiếp thu thập tại các vườn xoài. Những hình ảnh này phản ánh đúng bối cảnh mà ứng dụng sẽ được sử dụng, bao gồm các yếu tố nhiễu như lá cây, cành xoài và các góc chụp không chuẩn mực.

Dữ liệu Internet: Nhóm đã chọn lọc các hình ảnh công khai trên mạng để bổ sung các chủng loại xoài khác nhau và các tình trạng bệnh lý hiếm gặp mà dữ liệu thực tế chưa bao phủ hết.

Tổng số lượng: Sau quá trình sàng lọc, bộ dữ liệu cuối cùng đạt quy mô 1.500 hình ảnh.

Đặc điểm chi tiết của bộ dữ liệu Để mô hình YOLOv8 đạt được độ chính xác cao trong các điều kiện môi trường khác nhau, bộ dữ liệu đã được xây dựng dựa trên các tiêu chí đa dạng hóa:

Độ phong phú về bối cảnh: Hình ảnh bao quát nhiều điều kiện ánh sáng khác nhau, từ ánh sáng mặt trời rực rỡ đến các vùng bóng râm hoặc điều kiện thiếu sáng.

Đa dạng về góc độ và khoảng cách: Các đối tượng trái xoài được chụp ở nhiều góc độ (nhìn ngang, từ trên xuống) và khoảng cách (cận cảnh hoặc chụp xa).

Mật độ đối tượng: Mỗi hình ảnh chứa trung bình từ 1 đến 4 trái xoài với các trạng thái phát triển khác nhau, tạo độ phức tạp cần thiết để

mô hình rèn luyện khả năng phát hiện đồng thời nhiều đối tượng trong một khung hình.

Phân loại các lớp đối tượng (Classes) Dựa trên mục tiêu của đề tài, dữ liệu được chia thành 3 lớp chính để gán nhãn:

Ripe-mango (Xoài chín): Các quả có màu vàng đặc trưng, bề mặt nhẵn.

Unripe-mango (Xoài xanh): Các quả đang trong quá trình phát triển, có màu xanh lục.

Unhealthy (Xoài hỏng): Các quả có dấu hiệu bất thường như đốm đen, vết thối hoặc bị sâu bệnh tấn công.

Phân chia tập dữ liệu huấn luyện Để đánh giá mô hình một cách công bằng nhất, bộ dữ liệu 1.089 ảnh được Roboflow tự động phân tách ngẫu nhiên theo tỷ lệ chuẩn của MLOps:

Tập huấn luyện (Train set): Chiếm 88% (tương đương 954 ảnh), được mô hình sử dụng trực tiếp để học các đặc trưng.

Tập kiểm thử (Validation set): Chiếm 8% (tương đương 90 ảnh), dùng để điều chỉnh các siêu tham số trong quá trình huấn luyện nhằm tránh hiện tượng overfitting.

Tập đánh giá (Test set): Chiếm 4% (tương đương 45 ảnh), là tập dữ liệu hoàn toàn mới để đánh giá khách quan năng lực nhận diện của mô hình sau khi hoàn tất huấn luyện.

Việc chuẩn bị kỹ lưỡng bộ dữ liệu đa dạng này là tiền đề vững chắc để mô hình YOLOv8 có thể nhận diện chính xác chất lượng trái xoài trong thực tế, ngay cả trong những điều kiện môi trường khắc nghiệt.

2.1.2 Gán nhãn với Roboflow

Sau khi hoàn tất giai đoạn thu thập và sàng lọc dữ liệu thô, toàn bộ số lượng hình ảnh đã được tải lên nền tảng Roboflow để bắt đầu quy trình gán nhãn (Annotation). Đây là bước then chốt trong quy trình xây dựng mô hình học sâu, vì các nhãn dán này đóng vai trò là "tri thức" giúp mô hình học cách phân biệt các trạng thái khác nhau của trái xoài.

Quy trình thực hiện chi tiết: Giao diện làm việc của Roboflow cung cấp bộ công cụ trực quan, cho phép nhóm thực hiện vẽ các khung bao (bounding boxes) bao quanh từng đối tượng trái xoài xuất hiện trong mỗi bức ảnh. Mỗi

khung bao được vẽ sát với biên giới hạn của quả xoài để giúp mô hình học được đặc trưng về hình dáng và kích thước một cách chuẩn xác nhất. Với mỗi đối tượng đã được khoanh vùng, một nhãn lớp tương ứng sẽ được chỉ định ngay lập tức.

Nhóm thực hiện đã quyết định phương pháp gán nhãn hoàn toàn thủ công. Mặc dù quá trình này tiêu tốn nhiều thời gian và công sức, nhưng nó đảm bảo tính chính xác cao nhất cho bộ dữ liệu, đặc biệt là trong việc phân loại các trạng thái chuyển tiếp giữa xoài xanh và xoài chín hoặc nhận diện các vết hồng nhỏ trên bề mặt.

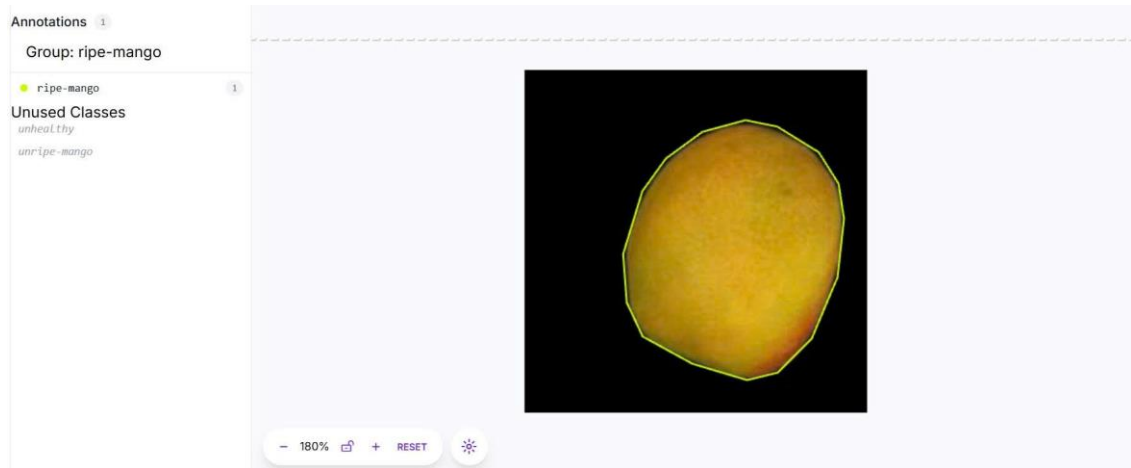
Hệ thống các nhãn lớp (Classes): Để phục vụ cho bài toán nhận diện độ chín và chất lượng, dữ liệu trong đề tài được chia thành 3 lớp nhãn đặc trưng:

ripe-mango (Xoài chín): Được gán cho các quả xoài đã đạt độ chín, biểu hiện qua màu sắc vỏ chuyển sang vàng.

unripe-mango (Xoài xanh): Dành cho các quả xoài còn xanh, chưa đạt độ chín thu hoạch.

overripe-mango (Xoài hồng): Nhãn này được áp dụng cho những trái xoài có dấu hiệu bệnh lý, xuất hiện đốm đen, vết thối hoặc các hư hại vật lý khác trên vỏ.

Kết quả và Xuất dữ liệu: Hình ảnh 2.1 minh họa giao diện thực tế khi nhóm thực hiện gán nhãn cho một quả xoài chín trên nền tảng Roboflow. Sau khi hoàn tất việc gán nhãn cho toàn bộ kho dữ liệu, Roboflow hỗ trợ tính năng xuất (export) dữ liệu sang định dạng YOLO. Tập tin xuất ra bao gồm các tệp hình ảnh và các tệp văn bản tương ứng chứa tọa độ khung bao cùng mã hóa nhãn lớp, hoàn toàn tương thích để đưa vào huấn luyện trên framework Ultralytics YOLOv8. Việc chuẩn hóa dữ liệu ngay từ bước này giúp quy trình huấn luyện trên Google Colab sau đó diễn ra một cách liền mạch và hiệu quả.



Hình 2.1: Giao diện gán nhãn hình ảnh với Roboflow

2.1.3 Chia tập dữ liệu

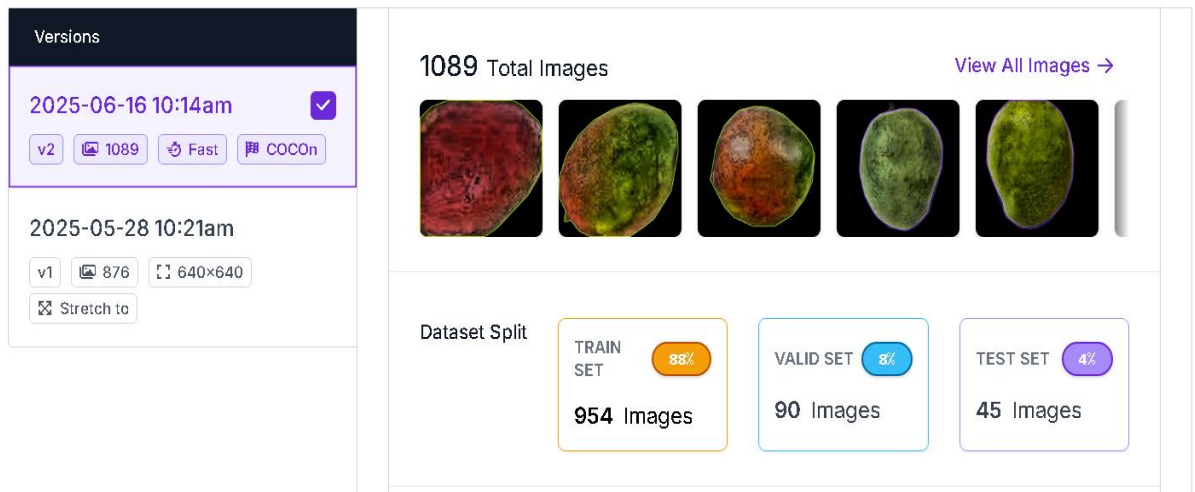
Sau khi gán nhãn, Roboflow cung cấp công cụ chia tập tự động theo tỉ lệ:

Train: 88% (dùng để huấn luyện)

Validation: 8% (dùng để theo dõi mô hình trong quá trình huấn luyện)

Test: 4% (đánh giá cuối cùng sau khi huấn luyện hoàn tất)

Việc chia dữ liệu ngẫu nhiên giúp tăng khả năng tổng quát cho mô hình, hạn chế hiện tượng overfitting.



Hình 2.2: Biểu đồ chia dữ liệu Train/Val/Test trên Roboflow

2.1.4 Tiền xử lý dữ liệu hình ảnh

Tiền xử lý dữ liệu (Data Preprocessing) là một bước bắt buộc nhằm chuẩn hóa các hình ảnh thô trước khi đưa vào huấn luyện². Việc này giúp loại bỏ các yếu tố nhiễu, đồng nhất định dạng đầu vào và giúp mô hình YOLOv8 tập trung

tốt hơn vào các đặc trưng quan trọng của trái xoài. Dựa trên cấu hình thực tế trên nền tảng Roboflow, nhóm đã thực hiện các kỹ thuật tiền xử lý sau⁴:

Tự động định hướng (Auto-Orient): Các hình ảnh được thu thập từ nhiều thiết bị khác nhau (điện thoại, máy ảnh) thường chứa siêu dữ liệu EXIF về góc quay của camera. Bước Auto-Orient giúp loại bỏ các dữ liệu này và tự động xoay ảnh về đúng chiều chuẩn. Điều này đảm bảo tính nhất quán về không gian cho mô hình, tránh trường hợp mô hình bị nhầm lẫn khi nhận diện cùng một đối tượng nhưng ở các hướng quay khác nhau⁷.

Thay đổi kích thước (Resize: Stretch to 640x640): Mô hình YOLOv8 yêu cầu kích thước hình ảnh đầu vào cố định để thực hiện các phép tính toán ma trận trong mạng nơ-ron.

Toàn bộ tập dữ liệu đã được thay đổi kích thước về chuẩn 640 x 640 pixels.

Nhóm sử dụng phương pháp Stretch (kéo giãn) để lấp đầy khung hình. Việc đưa ảnh về cùng một kích thước giúp tối ưu hóa tốc độ huấn luyện trên GPU của Google Colab và đảm bảo tính tương thích khi triển khai trên thiết bị di động thông qua TFLite.

Điều chỉnh độ tương phản (Auto-Adjust Contrast: Adaptive Equalization): Đây là một kỹ thuật quan trọng để cải thiện chất lượng hình ảnh trong các điều kiện ánh sáng không lý tưởng.

Adaptive Equalization giúp cân bằng độ sáng và tăng cường độ tương phản tại các vùng cục bộ của bức ảnh¹³.

Kỹ thuật này giúp làm nổi bật các chi tiết nhỏ như vết đốm trên vỏ xoài hỏng (unhealthy) hoặc sự khác biệt về sắc thái giữa xoài xanh và xoài chín, từ đó giúp mô hình đạt độ chính xác cao hơn khi làm việc với ảnh chụp thực tế có độ sáng phức tạp.

Preprocessing	Auto-Orient: Applied
	Resize: Stretch to 640x640
	Auto-Adjust Contrast: Using Adaptive Equalization

Hình 2.3: Tiền xử lý dữ liệu hình ảnh

2.1.5 Cấu hình file huấn luyện

Sau khi tập dữ liệu đã được tiền xử lý và gán nhãn hoàn tất trên Roboflow, bước tiếp theo là thiết lập kết nối để đưa dữ liệu này vào môi trường huấn luyện. Quy trình này được thực hiện thông qua thư viện Python của Roboflow để đảm bảo tính đồng bộ và chính xác về phiên bản dữ liệu.

Cấu trúc mã nguồn cấu hình bao gồm các thành phần chính sau:

Xác thực API: Sử dụng mã api_key cá nhân để kết nối an toàn với tài khoản Roboflow, cho phép truy xuất các dự án riêng tư.

Định vị dự án: Chỉ định chính xác workspace và project (trong trường hợp này là mango-7cab0) để đảm bảo tải đúng tập dữ liệu trái xoài.

Quản lý phiên bản: Lựa chọn phiên bản cụ thể của tập dữ liệu (phiên bản số 6) để duy trì tính nhất quán, tránh nhầm lẫn với các lần gán nhãn thử nghiệm trước đó.

Định dạng xuất: Dữ liệu được tải về dưới định dạng yolov8, tự động tạo ra tệp data.yaml. Đây là tệp tin quan trọng nhất, chứa các thông tin về đường dẫn tập Train/Val/Test và danh sách các nhãn lớp (ripe-mango, unripe-mango, unhealthy) để mô hình có thể hiểu được cấu trúc dữ liệu.



```
from roboflow import Roboflow
rf = Roboflow(api_key="ovxn4d1gqP2WILj7LvWN")
project = rf.workspace("group1tjes-zcjl").project("mango-7cab0")
version = project.version(6)
dataset = version.download("yolov8")
|
```

Hình 2.4: Cấu trúc file huấn luyện

2.1.6 Thực hiện huấn luyện trên Google Colab

Với môi trường phần cứng mạnh mẽ là GPU Tesla T4, quá trình huấn luyện mô hình YOLOv8 được tiến hành thông qua dòng lệnh thực thi của framework Ultralytics. Đây là giai đoạn quan trọng nhất, nơi mô hình bắt đầu học các đặc trưng hình ảnh của trái xoài từ tập dữ liệu đã chuẩn bị.

Các tham số cấu hình chính trong quá trình huấn luyện bao gồm:

Tham số data: Chỉ định đường dẫn đến tệp data.yaml vừa tải về (/content/mango-6/data.yaml). Tệp này đóng vai trò là "bản đồ" dẫn đường cho mô hình tìm đến dữ liệu hình ảnh.

Tham số epochs: Được thiết lập là 100. Điều này có nghĩa là toàn bộ tập dữ liệu huấn luyện sẽ được đưa qua mạng nơ-ron 100 lần. Việc lựa chọn 100 epoch giúp mô hình có đủ thời gian để hội tụ, giảm thiểu hàm mất mát (loss) xuống mức thấp nhất mà không gây ra hiện tượng quá khớp (overfitting).

Tham số name: Đặt tên cho phiên huấn luyện là smkt_scanner. Mọi kết quả về trọng số mô hình (best.pt, last.pt), các biểu đồ đánh giá (mAP, Precision, Recall) sẽ được lưu trữ tập trung trong thư mục này để phục vụ cho việc kiểm thử và xuất bản mô hình sang định dạng TFLite sau này.

Quá trình huấn luyện trên Colab diễn ra hoàn toàn tự động, hệ thống sẽ liên tục cập nhật các chỉ số hiệu năng sau mỗi epoch, giúp nhóm thực hiện dễ dàng theo dõi tiến trình và đưa ra các điều chỉnh kịp thời.

```
 model.train(  
    data = "/content/mango-6/data.yaml",  
    epochs = 100,  
    name = "smkt_scanner"  
)
```

Hình 2.5: Thực hiện huấn luyện trên Google Colab

2.1.7 Theo dõi và đánh giá kết quả huấn luyện

Trong quá trình huấn luyện mô hình YOLOv8 trên Google Colab, hệ thống tự động sinh ra các biểu đồ trực quan để người thực hiện có thể giám sát sự thay đổi của các chỉ số hiệu năng qua từng chu kỳ (epoch). Việc theo dõi này giúp xác định liệu mô hình có đang học hiệu quả hay gặp phải hiện tượng quá khớp (overfitting).

Phân tích các hàm mất mát (Loss Functions) Dựa trên Hình 2.6, ta có thể đánh giá sự hội tụ của mô hình qua ba loại Loss chính:

Box Loss: Đo lường sai số giữa khung bao dự đoán và khung bao thực tế. Đường biểu diễn có xu hướng giảm mạnh và ổn định, cho thấy khả năng định vị vị trí trái xoài ngày càng chính xác.

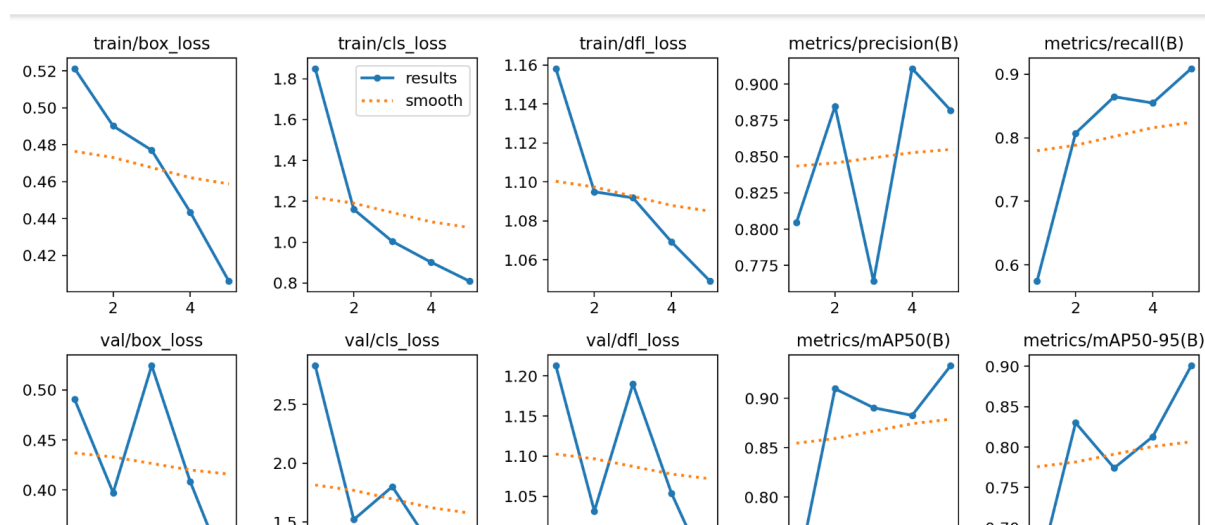
Class Loss (Cls Loss): Đo lường sai số trong việc phân loại các nhãn (Xanh, Chín, Hồng). Sự sụt giảm của chỉ số này chứng minh mô hình đang phân biệt tốt đặc trưng của từng loại xoài.

DFL Loss: Hỗ trợ tinh chỉnh độ chính xác của các cạnh khung bao. Cả ba chỉ số Loss trên tập huấn luyện (train) và tập kiểm định (val) đều giảm dần theo thời gian, khẳng định mô hình đang hội tụ tốt.

Phân tích các chỉ số đánh giá độ chính xác Các biểu đồ Metrics cung cấp cái nhìn tổng thể về năng lực thực tế của mô hình:

Precision & Recall: Cả hai chỉ số này đều tăng dần và đạt mức cao sau 100 epoch. Điều này có nghĩa là mô hình không chỉ dự đoán đúng các đối tượng mà còn nhận diện được hầu hết các trái xoài có trong ảnh.

mAP50 & mAP50-95: Đây là các chỉ số quan trọng nhất để đánh giá một mô hình Object Detection. Đường mAP đạt độ dốc lớn ở các epoch đầu và duy trì ở mức cao ở giai đoạn cuối, cho thấy mô hình đạt được sự cân bằng tối ưu giữa tốc độ và độ chính xác.



Hình 2.6: Biểu đồ loss và mAP được tạo trong quá trình huấn luyện YOLOv8

2.1.8 Kiểm thử và đánh giá mô hình

Độ chính xác (Precision, Recall, mAP)

Model summary (fused): 72 layers, 3,006,233 parameters, 0 gradients, 8.1 GFLOPs

Class	Images	Instances	Box(P)	R	mAP50	mAP50-95): 100%	3/3 [00:24:00:00, 8.32s/it]
all	90	94	0.882	0.909	0.933	0.901	
ripe-mango	26	30	0.867	0.866	0.919	0.861	
unhealthy	35	35	0.901	1	0.982	0.975	
unripe-mango	28	29	0.879	0.862	0.898	0.866	

Speed: 4.7ms preprocess, 252.6ms inference, 0.0ms loss, 0.0ms postprocess per image

Hình 2.7: Số liệu (Precision, Recall, mAP) trong quá trình huấn luyện YOLOv8

Sau khi huấn luyện mô hình YOLOv8 để phân loại trái xoài, kết quả thu được cho thấy hiệu suất của mô hình ở mức khá tốt, thể hiện qua ba chỉ số chính:

- Precision (Độ chính xác): Mô hình đạt 0.882 đối với toàn bộ dữ liệu, cho thấy phần lớn các dự đoán là đúng. Trong đó, lớp “unhealthy” có precision cao nhất (0.901), phản ánh khả năng dự đoán chính xác các trái xoài bị hư. Tuy nhiên, lớp “ripe-mango” có precision thấp hơn (0.667), cho thấy còn xảy ra hiện tượng dự đoán nhầm ở lớp này.
- Recall (Độ bao phủ): Chỉ số recall toàn tập là 0.909, cho thấy mô hình đã phát hiện được phần lớn các đối tượng thực sự có trong ảnh. Đặc biệt, lớp “unhealthy” đạt recall tuyệt đối 1.0, cho thấy mô hình không bỏ sót trường hợp nào. Các lớp còn lại cũng có recall cao trên 0.86.
- mAP (mean Average Precision): Chỉ số $mAP@0.5$ đạt 0.933, và $mAP@0.5:0.95$ đạt 0.901, cho thấy mô hình có khả năng phân loại tốt và ổn định ở nhiều mức độ giao nhau giữa nhãn dự đoán và nhãn thật. Các lớp đều có mAP cao (>0.86), phản ánh khả năng tổng thể của mô hình trong bài toán phân loại trái xoài là hiệu quả.

2.2 Tích hợp mô hình vào Android Studio

Sau khi hoàn tất quá trình huấn luyện và thu được mô hình có độ chính xác cao dưới dạng trọng số PyTorch (.pt), bước tiếp theo là đưa mô hình này vào môi trường phát triển Android Studio. Đây là giai đoạn chuyển giao công nghệ từ nghiên cứu thuật toán sang phát triển sản phẩm phần mềm, yêu cầu mô hình phải được tối ưu hóa để có thể chạy mượt mà trên tài nguyên hạn chế của điện thoại di động.

2.2.1 Xuất model sang TFLite với metadata

Mô hình YOLOv8 gốc thường có kích thước lớn và yêu cầu tài nguyên tính toán cao, không phù hợp để chạy trực tiếp trên các thiết bị Edge (thiết bị đầu cuối). Do đó, nhóm thực hiện đã chuyển đổi mô hình sang định dạng TensorFlow Lite (TFLite) – một định dạng được Google thiết kế riêng để tối ưu hóa hiệu suất và dung lượng cho các ứng dụng di động.

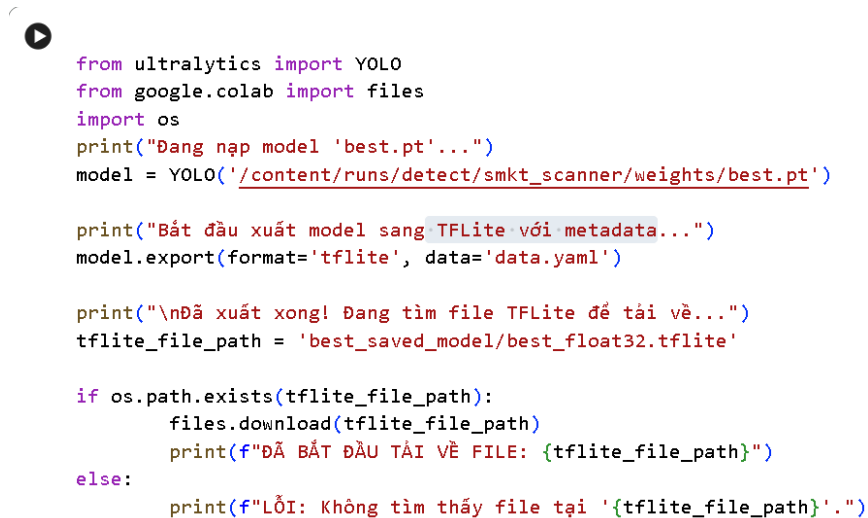
Quá trình chuyển đổi được thực hiện trực tiếp trên Google Colab bằng thư viện Ultralytics với các bước kỹ thuật sau:

Nạp mô hình tốt nhất: Sử dụng tệp trọng số best.pt – kết quả của phiên huấn luyện có độ chính xác cao nhất.

Xuất mô hình (Export): Để xuất mô hình ta thực hiện lệnh `model.export(format='tflite', data='data.yaml')`. Quá trình này không chỉ

thay đổi định dạng tệp mà còn tích hợp thêm Metadata (siêu dữ liệu). Metadata đóng vai trò cực kỳ quan trọng, nó chứa các thông tin về cấu trúc đầu vào/đầu ra, các lớp đối tượng (classes) và các tham số chuẩn hóa, giúp thư viện TensorFlow Lite trong Android Studio có thể hiểu và thực thi mô hình mà không cần cấu hình thủ công phức tạp.

Kết quả: Sau khi chuyển đổi thành công, nhóm thu được tệp `best_float32.tflite` sẵn sàng để tải về và tích hợp.



```
from ultralytics import YOLO
from google.colab import files
import os
print("Đang nạp model 'best.pt'...")
model = YOLO('/content/runs/detect/smkt_scanner/weights/best.pt')

print("Bắt đầu xuất model sang TFLite với metadata...")
model.export(format='tflite', data='data.yaml')

print("\nĐã xuất xong! Đang tìm file TFLite để tải về...")
tflite_file_path = 'best_saved_model/best_float32.tflite'

if os.path.exists(tflite_file_path):
    files.download(tflite_file_path)
    print(f"ĐÃ BẮT ĐẦU TẢI VỀ FILE: {tflite_file_path}")
else:
    print(f"LỖI: Không tìm thấy file tại '{tflite_file_path}'.")
```

Hình 2.8: Xuất model sang TFLite với metadata

2.2.2 Tích hợp TFLite vào Android Studio

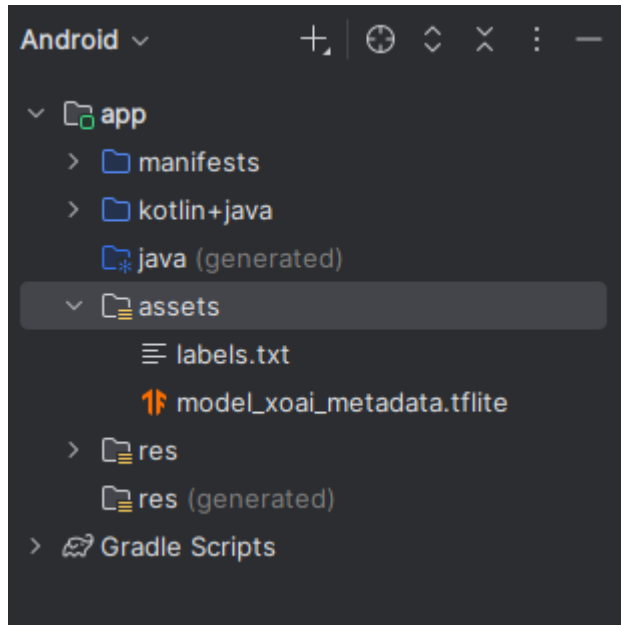
Sau khi có được tệp mô hình đã tối ưu, quy trình tích hợp vào mã nguồn ứng dụng Android được thực hiện qua các bước tổ chức thư mục chuẩn hóa:

Khởi tạo thư mục Assets: Trong cấu trúc dự án Android Studio, nhóm tạo (hoặc sử dụng) thư mục assets nằm trong đường dẫn `app/src/main/assets`. Đây là nơi lưu trữ các tài nguyên tĩnh mà ứng dụng sẽ truy cập trực tiếp trong thời gian thực thi.

Đưa mô hình vào ứng dụng: Tệp mô hình `.tflite` cùng với tệp danh sách nhãn `labels.txt` (chứa tên 3 lớp: `ripe-mango`, `unripe-mango`, `unhealthy`) được sao chép vào thư mục assets này.

Cấu hình thư viện: Trong tệp `build.gradle`, nhóm bổ sung các phụ thuộc (dependencies) cần thiết như `org.tensorflow:tensorflow-lite` để kích hoạt trình thông dịch (Interpreter) của TensorFlow Lite, cho phép ứng dụng nạp mô hình từ bộ nhớ và thực hiện suy luận hình ảnh trực tiếp trên camera.

Việc tích hợp này cho phép ứng dụng thực hiện tác vụ nhận diện hoàn toàn Offline (không cần internet), giúp tăng tốc độ phản hồi và bảo vệ quyền riêng tư dữ liệu của người dùng.



Hình 2.9: Thêm TFLite vào Android Studio

2.3 Các dịch vụ đám mây

Để ứng dụng không chỉ dừng lại ở việc nhận diện đơn thuần mà còn trở thành một hệ thống quản lý nông sản hoàn chỉnh, đồ án đã tích hợp các dịch vụ điện toán đám mây hiện đại. Việc sử dụng các dịch vụ này giúp hệ thống vận hành ổn định, bảo mật và giảm tải tài nguyên cho thiết bị di động.

2.3.1 Firebase

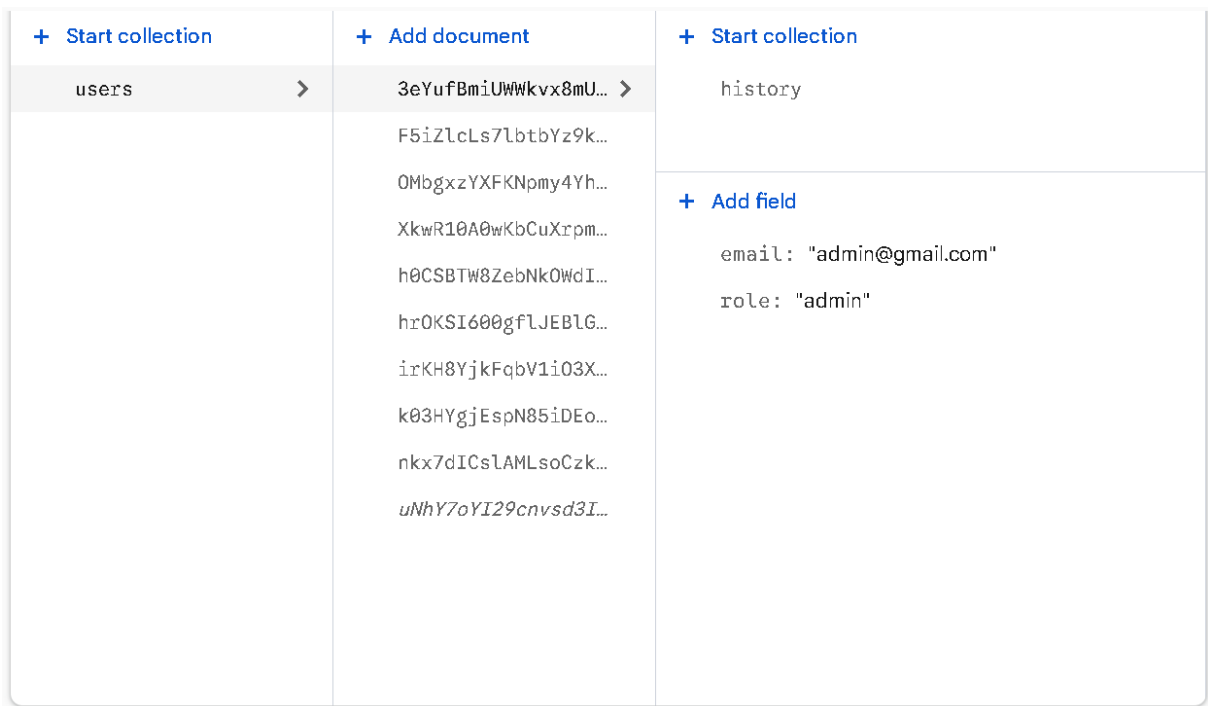
Firebase được sử dụng làm nền tảng quản trị backend chính cho ứng dụng, tập trung vào hai nhiệm vụ cốt lõi: xác thực danh tính và quản lý quyền truy cập.

Xác thực người dùng (Firebase Authentication): Hệ thống cung cấp cơ chế đăng ký và đăng nhập bảo mật qua Email. Mọi thông tin tài khoản đều được mã hóa và quản lý bởi Google, đảm bảo tính an toàn dữ liệu mà không cần xây dựng máy chủ riêng phức tạp.

Quản lý cơ sở dữ liệu (Cloud Firestore): Firebase cung cấp cơ sở dữ liệu NoSQL để lưu trữ thông tin hồ sơ người dùng và các tham chiếu lịch sử. Như minh họa tại Hình 2.10, hệ thống quản lý danh sách người dùng thông qua các ID duy nhất, cho phép phân quyền (như vai trò "admin" hoặc "user") để kiểm soát các tính năng tương ứng trên ứng dụng.

Duy trì phiên đăng nhập: Firebase hỗ trợ quản lý trạng thái đăng nhập, giúp người dùng không phải nhập lại thông tin nhiều lần và cho phép

ứng dụng xác định chính xác chủ sở hữu của các dữ liệu nhận diện được tải lên.



Hình 2.10: Quản lý người dùng

2.3.2 Cloudinary

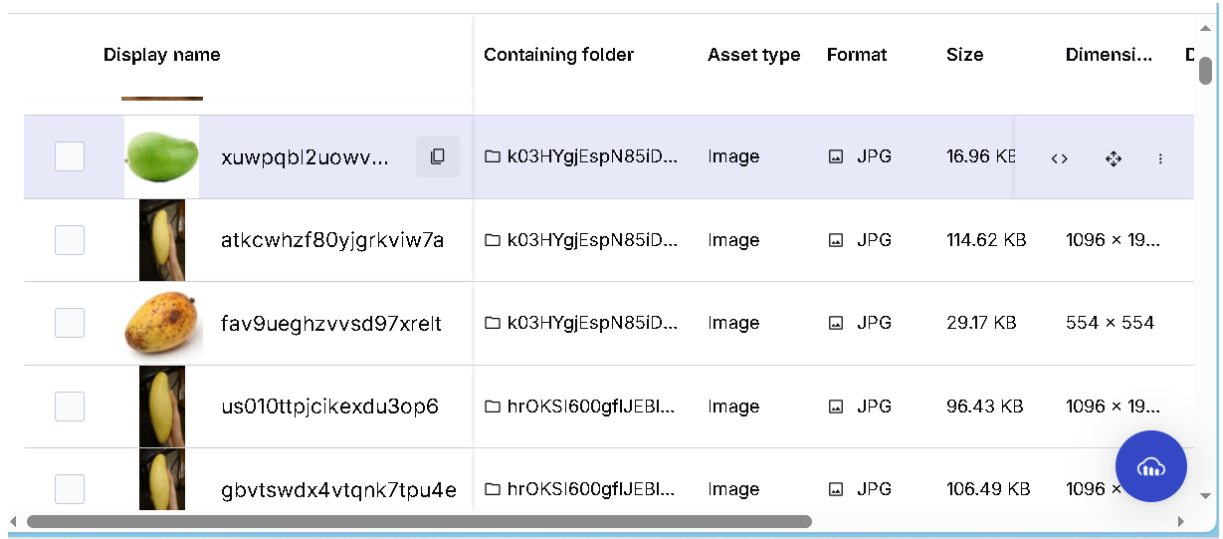
Song song với Firebase, Cloudinary đóng vai trò là "kho lưu trữ" hình ảnh thông minh trên đám mây, giúp giải quyết bài toán giới hạn bộ nhớ của thiết bị di động.

Tải lên và lưu trữ: Sau khi mô hình YOLOv8 thực hiện nhận diện xong, hình ảnh trái xoài sẽ được đẩy lên máy chủ Cloudinary thông qua API. Quá trình này giúp giải phóng dung lượng cho điện thoại người dùng vì ảnh không cần lưu trữ vĩnh viễn trong bộ nhớ máy.

Quản lý phương tiện truyền thông: Hình 2.11 cho thấy giao diện quản trị của Cloudinary, nơi các tập tin hình ảnh được tổ chức theo thư mục dựa trên ID người dùng. Mỗi hình ảnh khi tải lên thành công sẽ được trả về một đường dẫn (URL) duy nhất.

Truy xuất lịch sử nhận diện: Ứng dụng sử dụng các URL này để hiển thị lại hình ảnh trong mục lịch sử. Phương thức này giúp việc tải lại dữ liệu diễn ra nhanh chóng, mượt mà và cho phép người dùng có thể xem lại kết quả nhận diện của mình từ bất kỳ thiết bị nào có kết nối internet.

Việc kết hợp đồng bộ giữa Firebase và Cloudinary đã tạo nên một hạ tầng backend mạnh mẽ, đảm bảo ứng dụng hoạt động như một giải pháp phần mềm chuyên nghiệp, sẵn sàng cho việc mở rộng quy mô người dùng trong tương lai.



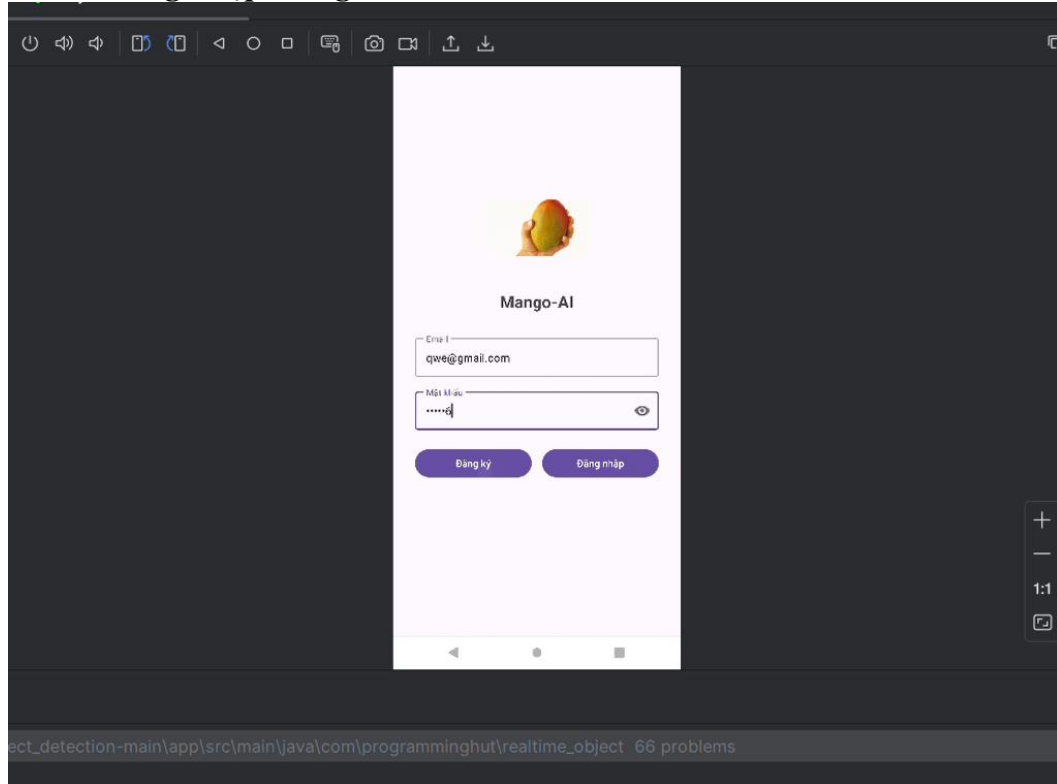
Display name	Containing folder	Asset type	Format	Size	Dimensi...
 xuwprqbl2uowv...	k03HYgjEspN85ID...	Image	JPG	16.96 KB	
 atkcwhzf80yjgrkviw7a	k03HYgjEspN85ID...	Image	JPG	114.62 KB	1096 × 19...
 fav9ueghzvvsd97xrelt	k03HYgjEspN85ID...	Image	JPG	29.17 KB	554 × 554
 us010ttpjcikexdu3op6	hrOKSI600gflJEBI...	Image	JPG	96.43 KB	1096 × 19...
 gbvtswdx4vtqnk7tpu4e	hrOKSI600gflJEBI...	Image	JPG	106.49 KB	1096 ×

Hình 2.11: Quản lý hình ảnh

CHƯƠNG 3: KẾT QUẢ THỰC NGHIỆM VÀ ĐÁNH GIÁ

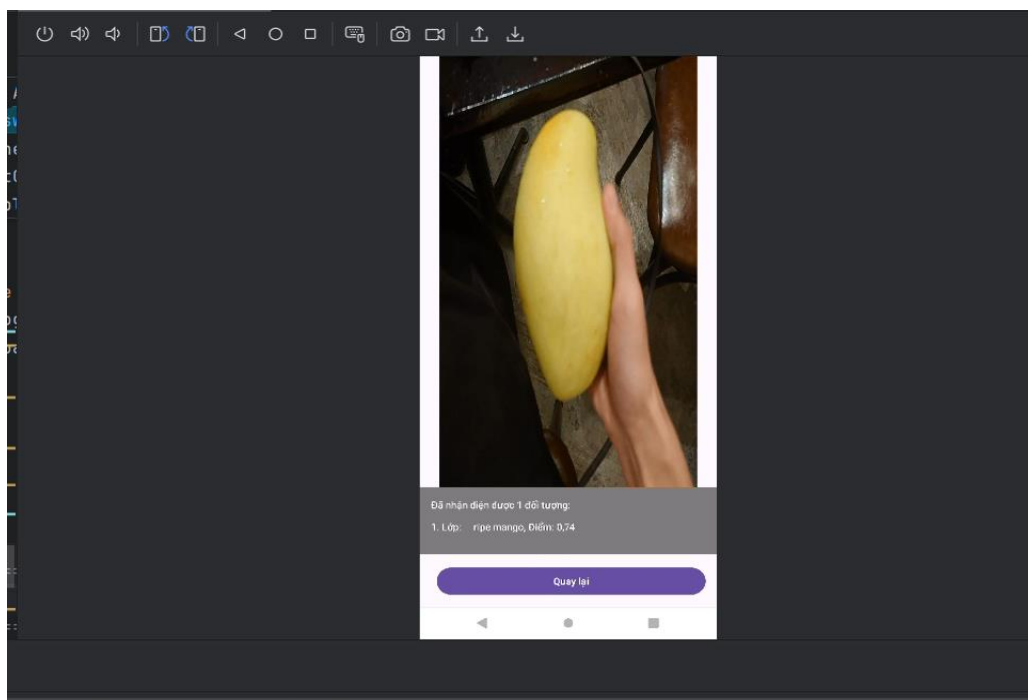
3.1 Demo sản phẩm

3.1.1 Đăng nhập, đăng kí



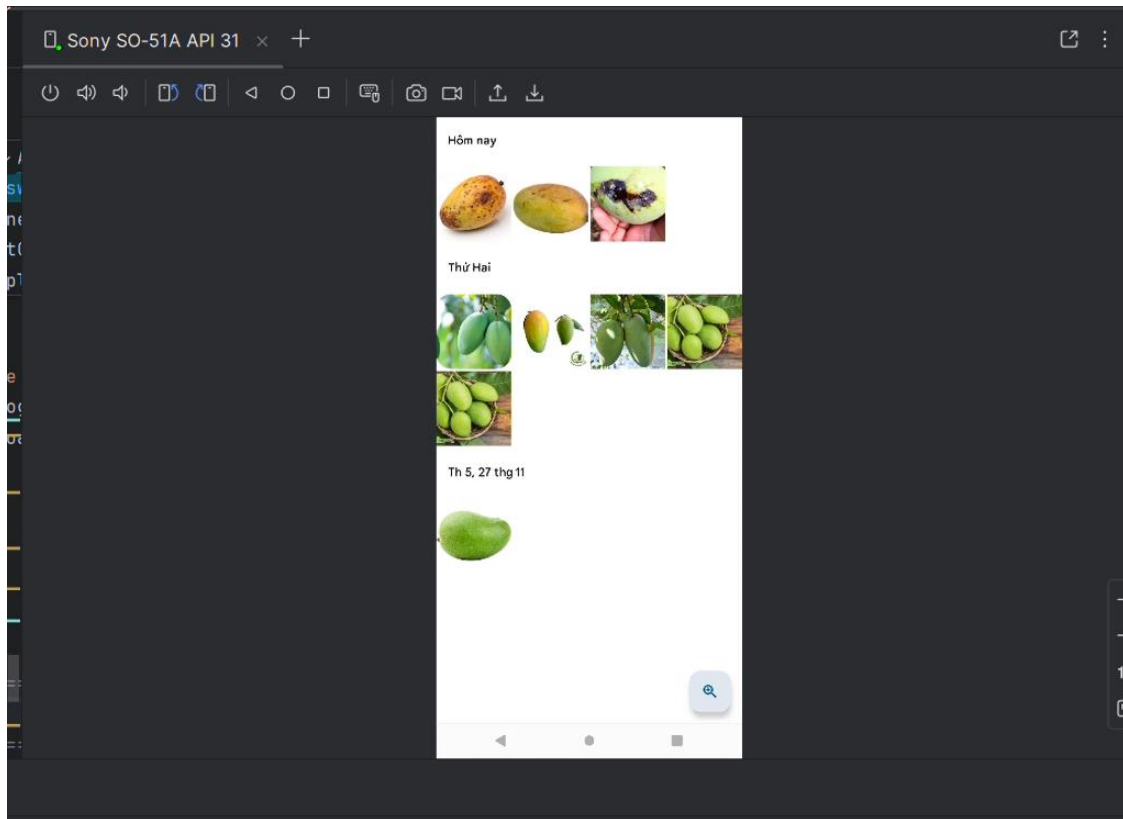
Hình 3.1: Giao diện đăng nhập, đăng kí

3.1.2 Chụp và nhận diện sản phẩm

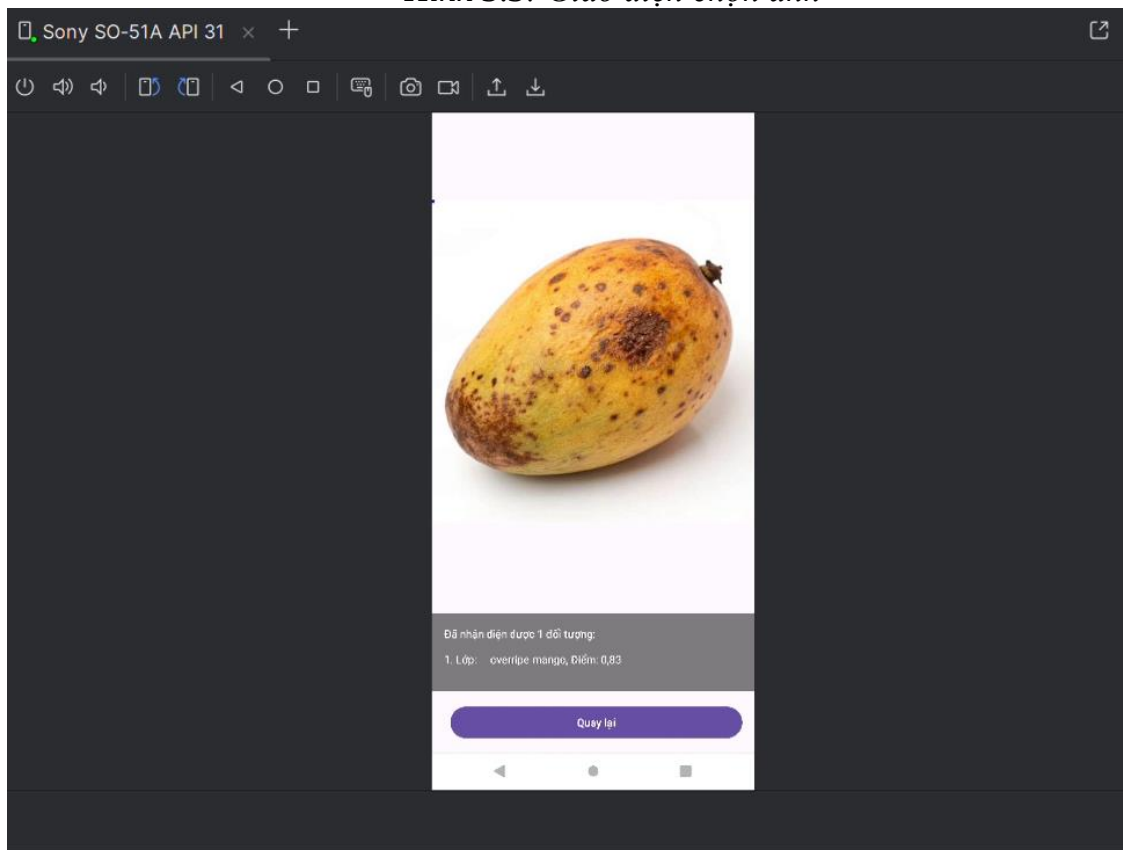


Hình 3.2: Giao diện nhận diện đối tượng

3.1.3 Nhận diện hình ảnh từ điện thoại

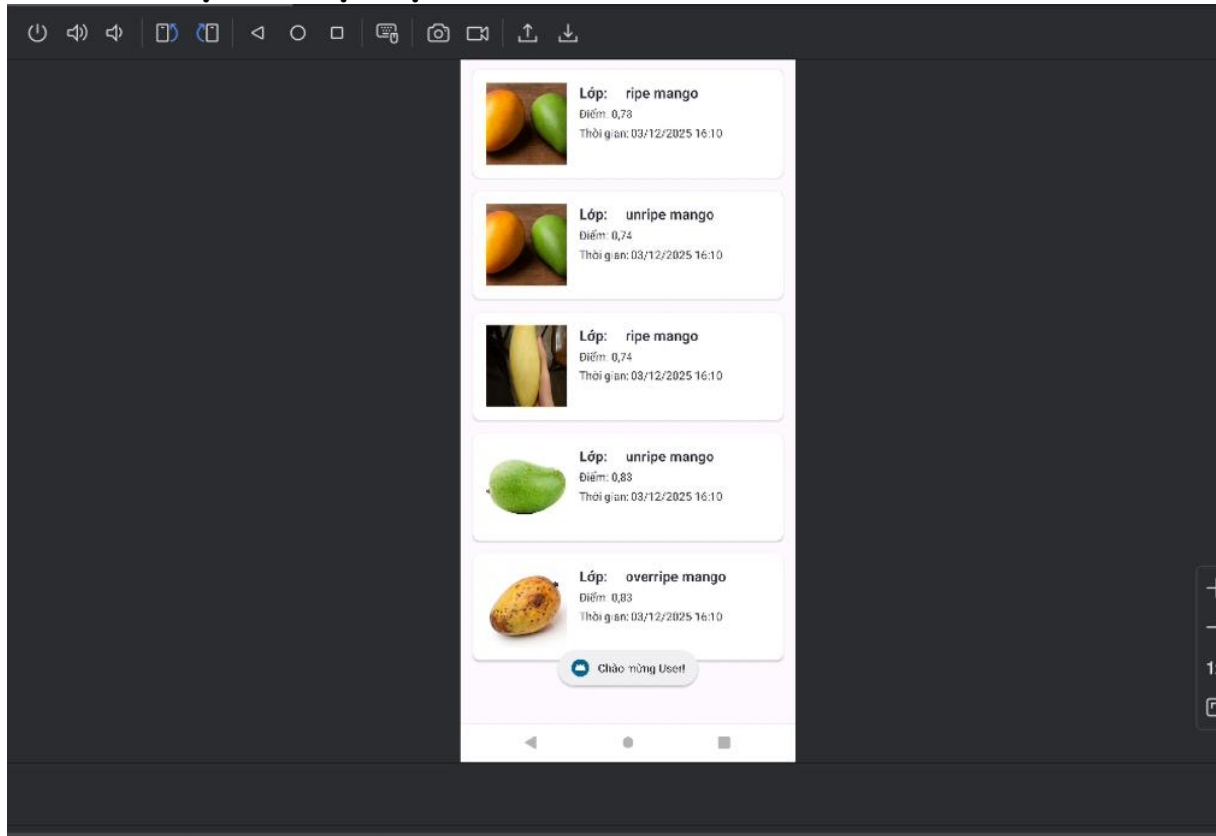


Hình 3.3: Giao diện chọn ảnh



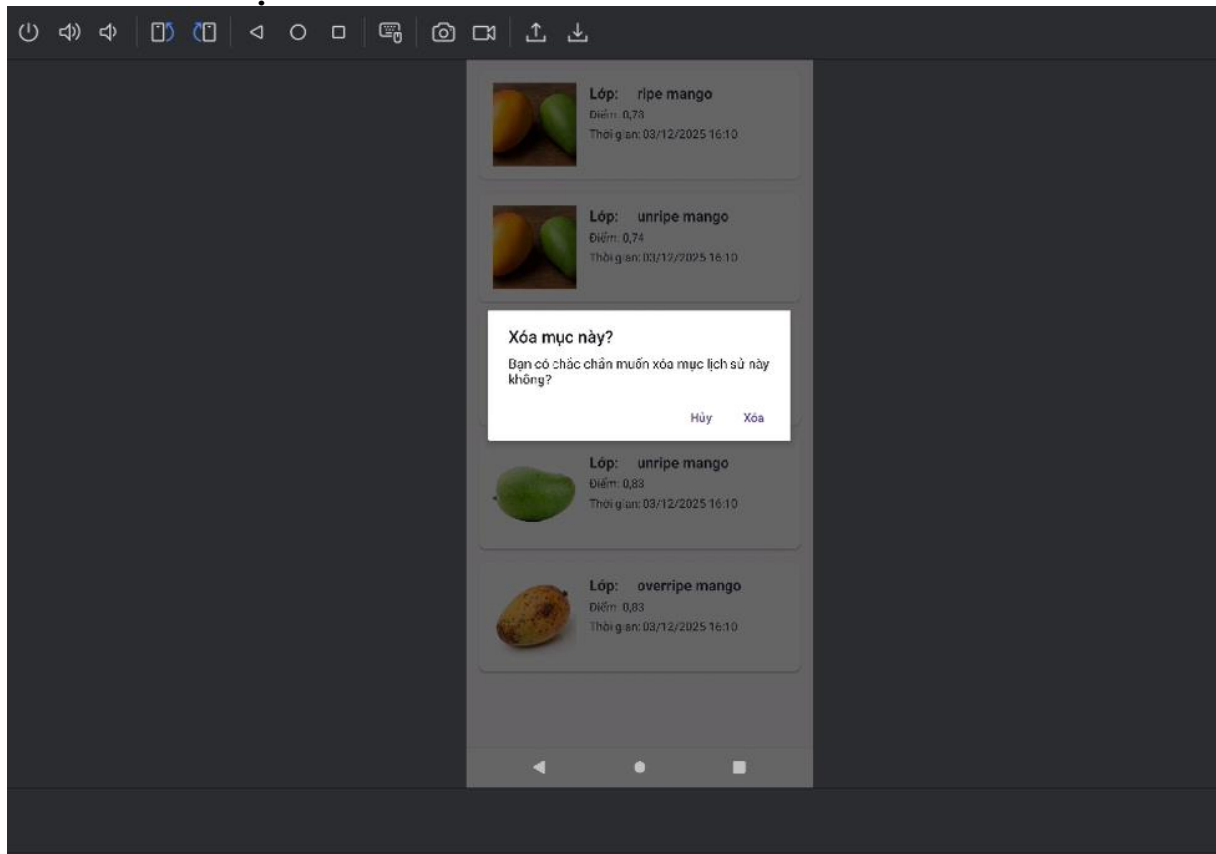
Hình 3.4: Nhận diện hình ảnh được chọn

3.1.4 Lịch sử nhận diện



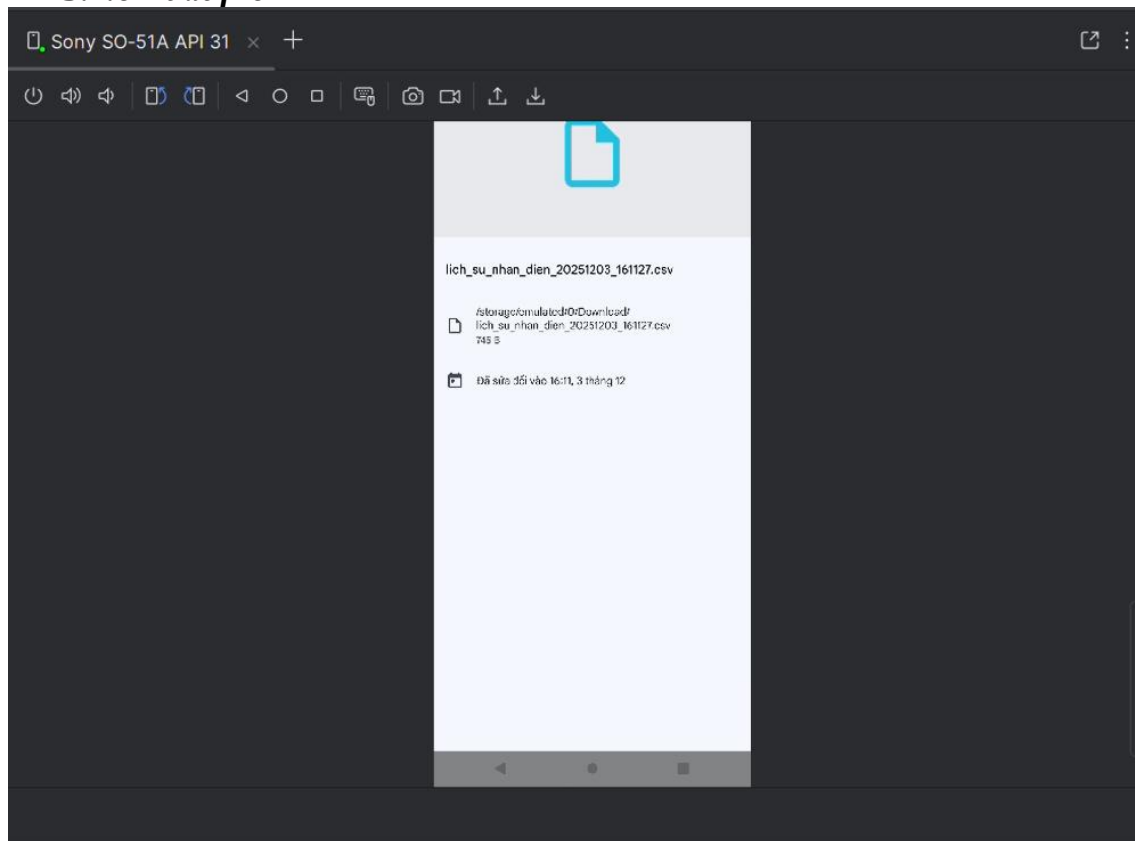
Hình 3.5: Giao diện lịch sử

3.1.5 Xóa lịch sử



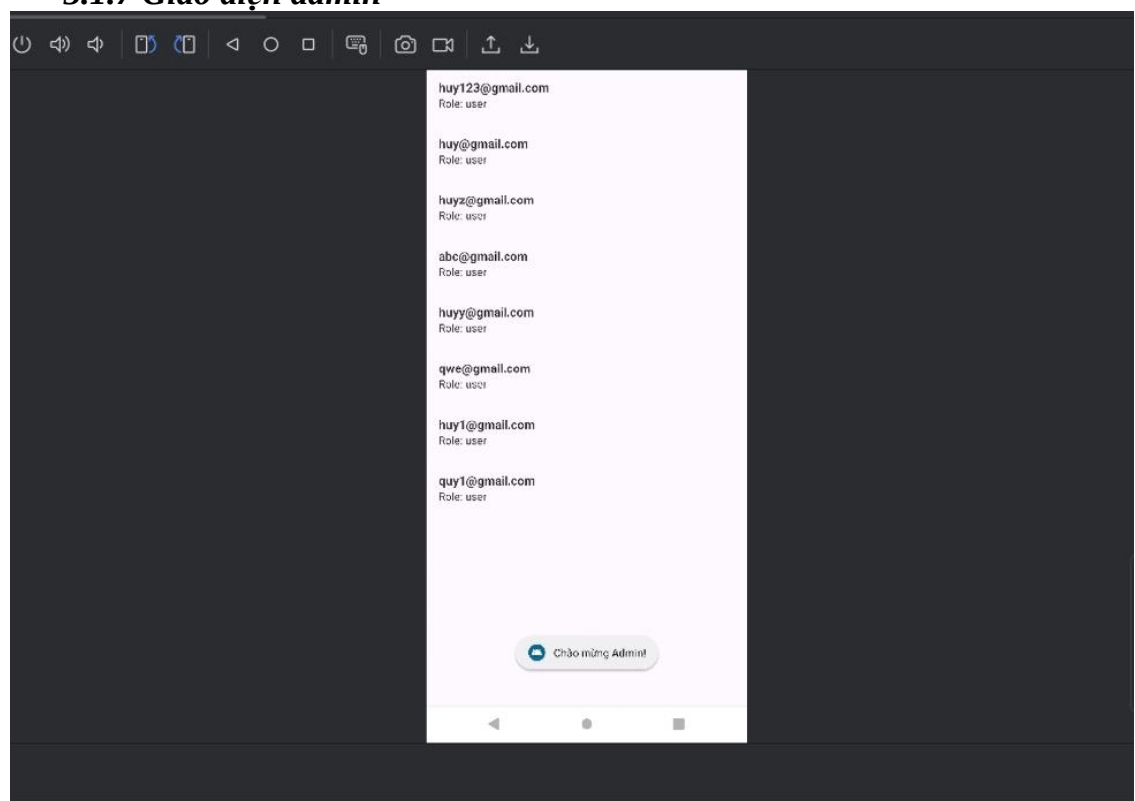
Hình 3.6: Xóa lịch sử

3.1.6 Xuất file

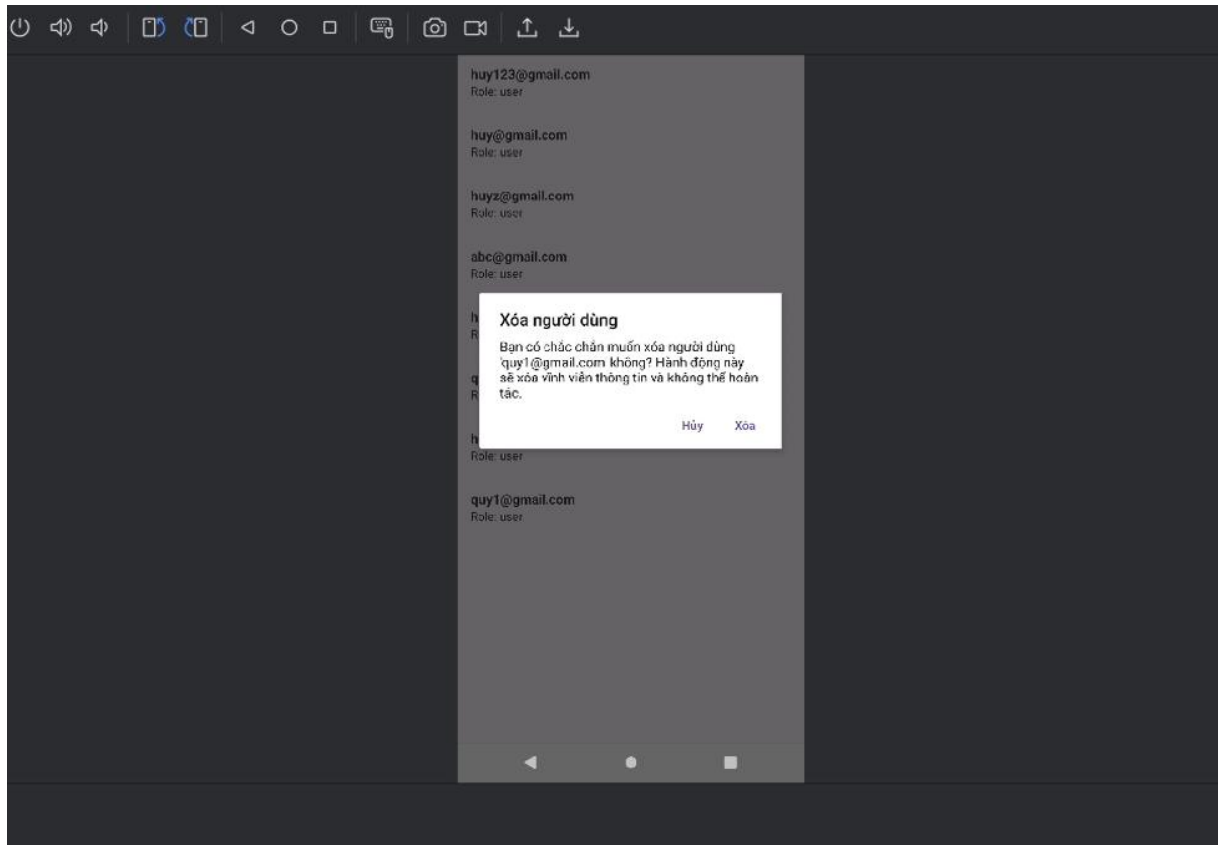


Hình 3.7: Xuất file

3.1.7 Giao diện admin

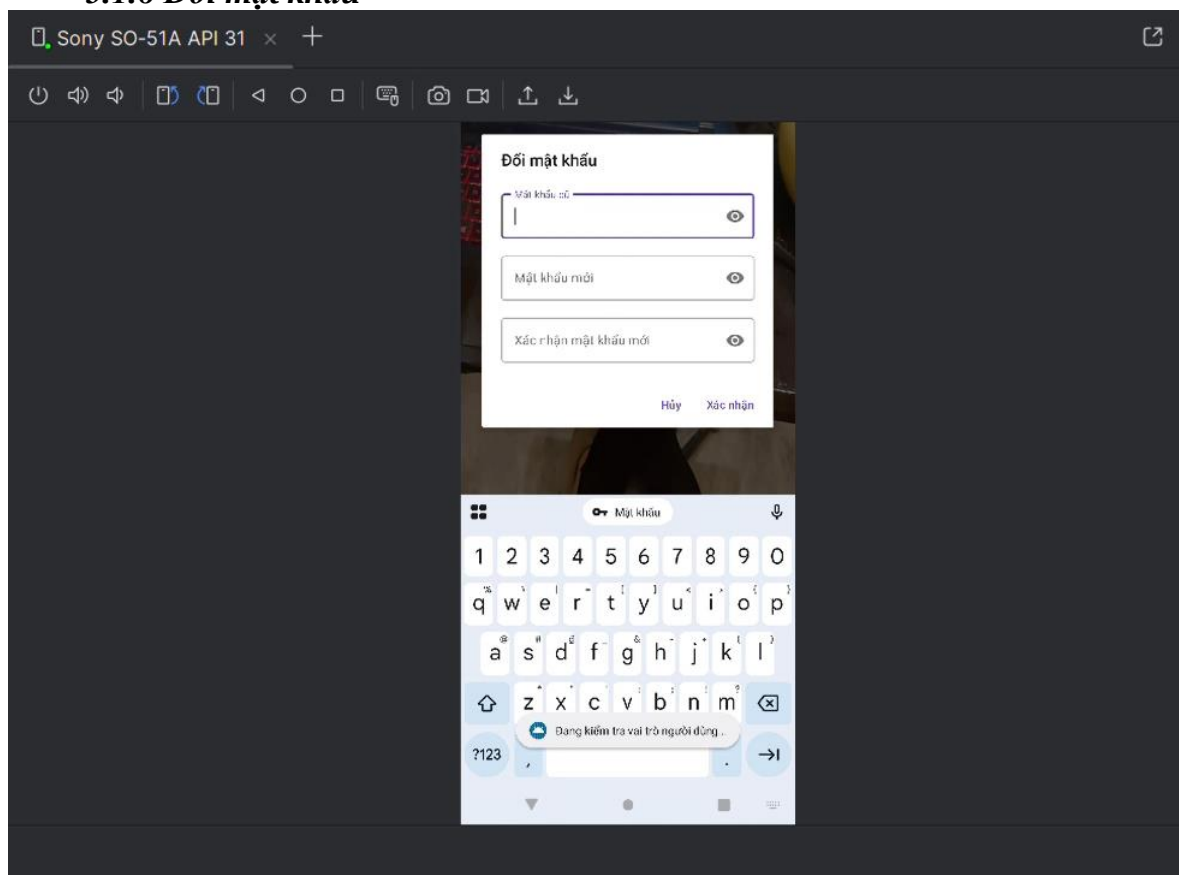


3.8 Giao diện Admin



3.9 Xóa người dùng

3.1.8 Đổi mật khẩu



3.10 Giao diện đổi mật khẩu

3.2 Bảng số liệu

Bảng 3.1 Bảng số liệu nhận diện

Kịch bản kiểm thử	Số lượng mẫu	Nhận diện đúng	Nhận diện sai/Không nhận diện	Tỷ lệ thành công
Trái xoài xanh	10	7	3	70%
Trái xoài chín	10	6	4	60%
Trái xoài hồng	10	7	3	70%
Tổng cộng	30	20	10	66,7%

3.3 Ưu nhược điểm hệ thống

3.3.1 Ưu điểm

Giao diện thân thiện, dễ sử dụng: Ứng dụng Android được thiết kế với giao diện trực quan, tập trung vào chức năng chính là nhận diện. Các trạng thái của xoài (Xanh, Chín, Hồng) được hiển thị rõ ràng, phù hợp với nhiều đối tượng người dùng, kể cả những người không am hiểu sâu về công nghệ

Quy trình quản lý dữ liệu toàn diện: Khác với các ứng dụng demo thông thường, hệ thống cung cấp một quy trình khép kín từ xác thực bảo mật (Firebase Authentication) đến lưu trữ dữ liệu đám mây (Cloudinary). Giải pháp này giúp người dùng không lo mất dữ liệu khi thiết bị và giảm tải dung lượng lưu trữ cho bộ nhớ điện thoại.

Tốc độ nhận diện nhanh nhờ việc tối ưu hóa mô hình sang định dạng TFLite chạy trên nền tảng di động, ứng dụng có khả năng nhận diện và phản hồi ngay lập tức khi người dùng chụp ảnh của đối tượng, giảm thiểu độ trễ, mang lại trải nghiệm mượt mà.

3.3.2 Nhược điểm

Phụ thuộc vào điều kiện môi trường: Độ chính xác của mô hình bị ảnh hưởng bởi điều kiện ánh sáng. Trong môi trường quá tối hoặc chói sáng (ngược sáng), khả năng nhận diện của camera giảm sút, dẫn đến việc mô hình có thể không bắt được hình ảnh chuẩn xác của đối tượng

Giới hạn về sự đa dạng chủng loại: Mô hình hiện tại được huấn luyện tốt trên các dữ liệu phổ biến từ Roboflow. Tuy nhiên, với các giống xoài lạ hoặc có hình dáng quá khác biệt so với tập dữ liệu huấn luyện, hệ thống có thể nhận diện nhầm lẫn (ví dụ: nhận diện nhầm quả đu đủ xanh thành xoài xanh).

Chưa hỗ trợ nhận diện theo thời gian thực (Real-time Detection): Hiện tại, ứng dụng hoạt động dựa trên cơ chế xử lý ảnh tĩnh (Static Image Processing). Người dùng cần thực hiện thao tác chụp ảnh thủ công hoặc chọn ảnh từ thư viện

để hệ thống tiến hành phân tích và trả về kết quả. Tính năng nhận diện liên tục trực tiếp trên luồng video (Live Camera Preview) chưa được tích hợp, điều này làm tăng số lượng thao tác cần thiết và giảm tính tương tác tức thời của người dùng so với trải nghiệm thực tế. Tiêu thụ tài nguyên thiết bị: Tác vụ xử lý ảnh và chạy mô hình AI liên tục (Computer Vision) yêu cầu hiệu năng CPU/GPU cao, dẫn đến việc thiết bị di động có thể bị nóng lên và tiêu hao pin nhanh nếu sử dụng ứng dụng trong thời gian dài liên tục.

KẾT LUẬN

1. Kết luận chung

Đề án “Nghiên cứu YOLOv8, xây dựng ứng dụng Android nhận diện độ chín trái xoài” được thực hiện trong bối cảnh nền nông nghiệp 4.0 đang đòi hỏi những giải pháp công nghệ mang tính thực tiễn cao để tối ưu hóa quy trình kiểm soát chất lượng nông sản. Thông qua quá trình nghiên cứu và triển khai, đề tài đã giải quyết thành công bài toán đưa trí tuệ nhân tạo (AI) từ môi trường nghiên cứu lý thuyết lên các thiết bị di động có tài nguyên giới hạn.

Quá trình thực hiện đề án là một chuỗi các công việc mang tính hệ thống, từ việc thu thập dữ liệu thô, thực hiện gán nhãn chuyên sâu, huấn luyện mô hình học sâu trên điện toán đám mây cho đến việc đóng gói và phát triển ứng dụng phía người dùng cuối. Kết quả cuối cùng không chỉ là một mô hình AI đạt độ chính xác cao mà còn là một sản phẩm phần mềm hoàn thiện, có khả năng vận hành thực tế trên điện thoại thông minh. Đề án đã chứng minh rằng việc kết hợp giữa các thuật toán State-of-the-Art như YOLOv8 và nền tảng Android là một hướng đi khả thi, hiệu quả và có tiềm năng ứng dụng rộng rãi trong việc hỗ trợ người nông dân phân loại sản phẩm nhanh chóng và khách quan.

2. Các kết quả đạt được

Sau một thời gian tập trung nghiên cứu và thực hiện, đề án đã đạt được những thành quả cụ thể và mang tính thực tiễn sau:

- Về mặt khoa học và thuật toán:
 - Nhóm đã làm chủ được quy trình MLOps (Machine Learning Operations), từ giai đoạn quản lý dữ liệu trên Roboflow đến việc tối ưu hóa huấn luyện trên Google Colab.
 - Mô hình YOLOv8 sau khi huấn luyện qua 100 epoch đã đạt được các chỉ số hiệu suất ấn tượng: độ chính xác trung bình toàn tập (mAP50) đạt 93.3%.
 - Đặc biệt, khả năng nhận diện lớp "Xoài hỏng" (Unhealthy) đạt độ chính xác xấp xỉ 98%. Đây là kết quả cực kỳ quan trọng, cho thấy mô hình có khả năng hỗ trợ đắc lực trong việc loại bỏ nông sản kém chất lượng, đáp ứng xuất sắc mục tiêu kiểm định chất lượng sản phẩm đặt ra ban đầu.

- Về mặt sản phẩm ứng dụng:
 - Xây dựng thành công ứng dụng trên nền tảng Android với ngôn ngữ Kotlin, sở hữu giao diện trực quan, thân thiện và dễ tiếp cận đối với cả những người dùng không am hiểu sâu về công nghệ.
 - Tích hợp thành công mô hình đã nén dưới định dạng TFLite, cho phép ứng dụng thực hiện nhận diện hoàn toàn Offline, đảm bảo tốc độ phản hồi nhanh và tính riêng tư cho dữ liệu của người dùng.
 - Hệ thống kết nối đồng bộ với các dịch vụ đám mây: Firebase Authentication giúp quản lý và bảo mật thông tin tài khoản người dùng tuyệt đối ; trong khi Cloudinary cung cấp giải pháp lưu trữ hình ảnh thông minh, giúp tối ưu hóa dung lượng thiết bị và cho phép truy xuất lịch sử nhận diện mọi lúc mọi nơi.
 - Ứng dụng hoạt động ổn định trên các thiết bị thật, có khả năng xử lý ảnh từ camera hoặc thư viện và trả về kết quả nhận diện (nhân lớp và độ tin cậy) với độ trễ thấp

3. Hạn chế của đề tài

Mặc dù đã đạt được những kết quả khả quan trong việc xây dựng mô hình và ứng dụng, nhóm thực hiện vẫn nhận thấy một số hạn chế nhất định cần được xem xét và khắc phục trong tương lai:

- Chưa tích hợp nhận diện thời gian thực (Real-time Detection): Hiện tại, ứng dụng hoạt động dựa trên cơ chế xử lý ảnh tĩnh (Static Image Processing). Người dùng bắt buộc phải thực hiện thao tác chụp ảnh thủ công hoặc chọn ảnh có sẵn từ thư viện để hệ thống tiến hành phân tích. Việc thiếu tính năng nhận diện trực tiếp trên luồng camera (Live Camera Preview) làm tăng số lượng thao tác, gây hạn chế về tính tương tác tức thời và hiệu quả khi cần kiểm tra số lượng lớn nông sản trong thời gian ngắn.
- Độ phụ thuộc cao vào chất lượng bộ dữ liệu: Hiệu suất của mô hình YOLOv8 phụ thuộc rất lớn vào tính đa dạng của dữ liệu huấn luyện. Dù đã thu thập hơn 1.000 ảnh, nhưng bộ dữ liệu chủ yếu tập trung vào các giống xoài phổ biến tại Việt Nam. Đối với các giống xoài ngoại nhập, các giống có hình dạng bất thường hoặc màu sắc đặc biệt không nằm trong tập huấn luyện, độ chính xác của hệ thống có thể bị giảm sút đáng kể, dẫn đến hiện tượng nhận diện nhầm lẫn.
- Nhạy cảm với điều kiện môi trường thực tế: Độ chính xác của mô hình bị ảnh hưởng mạnh bởi điều kiện ánh sáng và bối cảnh. Trong môi trường quá tối, ngược

sáng mạnh hoặc khi trái xoài bị che khuất quá nhiều bởi lá và cành cây, khả năng trích xuất đặc trưng của mô hình TFLite trên điện thoại gặp nhiều khó khăn, dễ dẫn đến sai sót trong việc phân loại chất lượng.

- Sự phụ thuộc vào kết nối Internet cho các tính năng hệ thống: Mặc dù mô hình nhận diện hoạt động Offline , nhưng các tính năng quan trọng khác như đăng nhập, đăng ký và đồng bộ hóa lịch sử nhận diện đều phụ thuộc hoàn toàn vào các dịch vụ đám mây (Firebase và Cloudinary). Điều này có nghĩa là nếu thiết bị không có kết nối mạng ổn định, người dùng không thể lưu trữ hoặc truy xuất lại dữ liệu lịch sử, gây gián đoạn trải nghiệm người dùng trong điều kiện làm việc tại vườn.
- Sự biến thiên về hiệu năng trên các dòng thiết bị: Do Android là một hệ điều hành có sự phân mảnh phần cứng rất lớn , tốc độ suy luận (Inference time) và mức độ tiêu thụ tài nguyên của ứng dụng không đồng nhất trên các dòng điện thoại khác nhau. Trên các thiết bị đời cũ hoặc có cấu hình thấp, tác vụ chạy mô hình AI liên tục có thể gây ra hiện tượng nóng máy và tiêu hao pin nhanh chóng.
- Sự đánh đổi giữa tốc độ và độ chính xác: Việc sử dụng biến thể YOLOv8n (Nano) để ưu tiên tốc độ xử lý trên di động là một sự lựa chọn tối ưu về hiệu năng. Tuy nhiên, về mặt lý thuyết, phiên bản này vẫn có độ chính xác thấp hơn so với các biến thể lớn hơn như YOLOv8m hay YOLOv8x. Do đó, trong một số trường hợp tình vi liên quan đến các vết hỏng nhỏ trên bề mặt quả, mô hình Nano có thể không đạt được độ nhạy tuyệt đối như kỳ vọng.

4. Đề xuất và Kiến nghị

Dựa trên những kết quả thực nghiệm đã đạt được cùng việc nhìn nhận khách quan các hạn chế hiện tại, nhóm thực hiện xin đề xuất các định hướng phát triển và kiến nghị cụ thể nhằm nâng cấp ứng dụng thành một giải pháp công nghệ toàn diện và chuyên nghiệp hơn:

Về khía cạnh kỹ thuật và công nghệ

• Nâng cấp khả năng nhận diện thời gian thực (Real-time Detection):

- Đây là ưu tiên hàng đầu trong giai đoạn tiếp theo. Nhóm đề xuất nghiên cứu tích hợp thư viện **CameraX** – một bộ công cụ hiện đại của Jetpack giúp đơn giản hóa việc quản lý camera trên Android – kết hợp cùng **TensorFlow Lite GPU Delegate**.
- Việc tận dụng sức mạnh xử lý song song của GPU trên điện thoại sẽ giúp mô hình thực hiện suy luận (inference) với tốc độ cực cao, đạt mức 30-60

khung hình trên giây (FPS). Khi đó, người dùng không cần phải thực hiện thao tác chụp ảnh thủ công mà chỉ cần đưa camera lướt qua các sọt nông sản; hệ thống sẽ tự động khoanh vùng, đếm số lượng và phân loại chất lượng các trái xoài ngay lập tức trên màn hình xem trước.

- **Mở rộng và đa dạng hóa tập dữ liệu huấn luyện (Dataset Expansion):**

- Để tăng tính tổng quát hóa (Generalization), cần tiếp tục thu thập dữ liệu về các giống xoài đặc trưng khác của Việt Nam (như xoài Cát Chu, xoài Tượng, xoài Keo...) ở nhiều giai đoạn trưởng thành khác nhau.
- Đặc biệt, cần bổ sung các hình ảnh được chụp trong điều kiện ánh sáng khác nghiệt, hậu cảnh phức tạp (nhiều lá, cành che khuất) hoặc các tư thế quả nằm chồng chéo lên nhau. Việc làm giàu kho dữ liệu này sẽ giúp mô hình "thông minh" hơn, giảm thiểu tối đa các sai sót khi ứng dụng triển khai tại các vùng địa lý và điều kiện canh tác khác nhau.

- **Tối ưu hóa mô hình (Model Quantization):**

- Nghiên cứu áp dụng các kỹ thuật như **Quantization (Lượng tử hóa)** để chuyển đổi mô hình từ định dạng Float32 sang Int8. Kỹ thuật này sẽ giúp giảm dung lượng file mô hình xuống gấp 4 lần và tăng tốc độ xử lý trên các dòng điện thoại cấu hình thấp mà không làm giảm đáng kể độ chính xác, giúp ứng dụng tiếp cận được với nhiều đối tượng người nông dân hơn.

Về khía cạnh ứng dụng và thực tiễn

- **Triển khai thử nghiệm và thu thập phản hồi người dùng (User-Centric Design):**

- Kiến nghị đưa ứng dụng vào giai đoạn chạy thử nghiệm (Pilot Test) tại các chợ thu mua nông sản quy mô nhỏ hoặc các hợp tác xã cây ăn quả.
- Thông qua việc quan sát hành vi sử dụng thực tế và thu thập ý kiến trực tiếp từ người nông dân, nhóm có thể điều chỉnh giao diện (UI/UX) sao cho tinh gọn nhất, tối giản hóa các bước thao tác và làm nổi bật các thông số quan trọng, từ đó đảm bảo tính khả thi cao khi ứng dụng vào đời sống sản xuất thực tế.

- **Phát triển hệ sinh thái "Sổ tay nông vụ thông minh":**

- Xây dựng module thống kê và phân tích dữ liệu chuyên sâu dựa trên lịch sử nhận diện. Hệ thống sẽ tự động tổng hợp số lượng trái đạt chuẩn, tỷ lệ trái hỏng theo từng lô hàng, từng ngày hoặc từng khu vực vườn.
- Các dữ liệu này sẽ được trực quan hóa bằng biểu đồ, cung cấp cái nhìn toàn cảnh về năng suất và chất lượng mùa vụ. Đây sẽ là công cụ đắc lực hỗ trợ người quản lý trong việc đánh giá hiệu quả canh tác, dự báo sản lượng và đưa ra các quyết định kinh doanh, thu hoạch một cách khoa học và chính xác hơn.

• **Tích hợp khả năng định vị và bản đồ số:**

- Đề xuất tích hợp GPS để gắn thẻ địa lý (Geotagging) cho mỗi bức ảnh nhận diện. Điều này giúp người quản lý trang trại lớn có thể xác định chính xác khu vực nào trong vườn đang có tỷ lệ xoài hỏng cao hoặc vùng nào đã đạt độ chín cần thu hoạch, từ đó tối ưu hóa quy trình điều phối nhân công và vận chuyển.

Với các định hướng phát triển này, nhóm thực hiện tin rằng ứng dụng không chỉ dừng lại ở một công cụ nhận diện hình ảnh đơn thuần mà sẽ trở thành một phần không thể thiếu trong hệ sinh thái nông nghiệp kỹ thuật số, góp phần nâng cao giá trị và vị thế của nông sản Việt trên thị trường.

TÀI LIỆU THAM KHẢO

1. Ultralytics. (2023). *YOLOv8 Documentation*. <https://docs.ultralytics.com>
2. Roboflow. (2024). *Roboflow Documentation*. <https://roboflow.com>
3. GitHub – Ultralytics YOLOv8 Repository. <https://github.com/ultralytics/ultralytics>
4. Dự án tham khảo:
<https://www.youtube.com/watch?v=zs43IrWTzB0&t=177s>
<https://www.youtube.com/watch?v=dl7rCmvIyiI>
5. Firebase:
https://firebase.google.com/docs?gclidsrc=aw.ds&gad_source=1&gad_campaignid=20100026061&gbraid=0AAAAADpUD0jfvF5N0mJ2gl_epB9CHwo9B&gclid=Cj0KCQiA_8TJBhDNARIsAPX5qxTiI5zHGegY94KINtyX0xX7SCixJzdpXcxdOm2L8lYsX5jBKxtLOfAaAvtzEALw_wcB&hl=vi
6. Cloudinary: <https://www.tensorflow.org/>
7. TensorFlow: <https://www.tensorflow.org/>
8. Train model: https://www.youtube.com/watch?v=g5-V5d_FYzs