

BỘ NÔNG NGHIỆP VÀ MÔI TRƯỜNG
PHÂN HIỆU TRƯỞNG ĐẠI HỌC THỦY LỢI
BỘ MÔN CÔNG NGHỆ THÔNG TIN



ĐỒ ÁN MÔN HỌC

TÊN ĐỀ TÀI:
HỆ THỐNG PHÂN LOẠI BỆNH HẠI TRÊN LÁ CÂY

Giảng viên hướng dẫn:

Giảng viên Vũ Thị Hạnh

Sinh viên thực hiện:

Phạm Thị Quỳnh Giao

Nguyễn Hữu Tuấn Phát

Đoàn Anh Vũ

MSSV:

2351267259

2351267274

2351267280

Lớp:

S26-65TTNT

TP. HỒ CHÍ MINH, 2026

MỤC LỤC

DANH MỤC BẢNG	3
DANH MỤC HÌNH VẼ	4
DANH MỤC KÝ HIỆU VIẾT TẮT	5
LỜI CẢM ƠN	6
1 TỔNG QUAN VÀ CƠ SỞ LÝ THUYẾT	7
1.1 Đặt Vấn Đề	7
1.2 Các Chỉ Số Đánh Giá (Evaluation Metrics)	7
1.2.1 Accuracy (Độ Chính Xác)	7
1.2.2 Precision (Độ Chính Xác Của Dự Đoán Dương)	7
1.2.3 Recall (Độ Nhạy / Tỷ Lệ Phát Hiện)	7
1.2.4 F1-Score	7
1.2.5 AUC - ROC	8
1.3 Bộ Dữ Liệu New Plant Diseases Dataset	8
1.3.1 Tổng Quan Dữ Liệu	8
1.3.2 Danh Sách Các Lớp Dữ Liệu	8
1.3.3 Hình Ảnh Minh Họa Các Lớp Bệnh	9
2 PHƯƠNG PHÁP NGHIÊN CỨU	11
2.1 Cơ Sở Lý Thuyết Về Mạng Nơ-ron Tích Chập (CNN)	11
2.1.1 Lớp Tích Chập (Convolutional Layer)	11
2.1.2 Hàm Kích Hoạt (Activation Function)	11
2.1.3 Lớp Gộp (Pooling Layer)	11
2.1.4 Lớp Gộp Trung Bình Thích Nghi (Adaptive Average Pooling)	12
2.1.5 Lớp Kết Nối Đầy Đủ (Fully Connected Layer - FC)	12
2.1.6 Lớp Dropout	12
2.1.7 Lớp Chuẩn Hóa Theo Batch (Batch Normalization)	12
2.2 Tiền Xử Lý Dữ Liệu (Data Preprocessing)	12
2.2.1 Chi Tiết Các Kỹ Thuật Augmentation	12
2.2.2 Validation Transform	14
2.3 Phương Pháp Data Injection (Bổ Sung Dữ Liệu)	14
2.4 Cấu Hình DataLoader	15
2.4.1 Train Loader (Bộ nạp dữ liệu huấn luyện)	15
2.4.2 Validation Loader (Bộ nạp dữ liệu kiểm thử)	16
2.5 Kiến Trúc Mô Hình (Model Architectures)	16
2.5.1 MobileNetV3 Large	16
2.5.2 EfficientNet-B0	17
2.5.3 ResNet-18	17

3 THỰC NGHIỆM VÀ KẾT QUẢ	19
3.1 Môi Trường Và Cấu Hình Huấn Luyện	19
3.1.1 Cấu Hình Phần Cứng và Phần Mềm	19
3.1.2 Chi Tiết Các Kỹ Thuật Tối Ưu Hóa (Optimization và Loss)	19
3.2 Kỹ Thuật Transfer Learning (Thay Thế Lớp Classifier)	21
3.2.1 MobileNetV3 Large	21
3.2.2 EfficientNet-B0	21
3.2.3 ResNet-18	21
3.3 Kết Quả Chi Tiết Từng Mô Hình	21
3.3.1 MobileNetV3 Large (Mô hình tốt nhất)	21
3.3.2 EfficientNet-B0	21
3.3.3 ResNet-18	22
3.4 So Sánh Tổng Hợp	22
3.5 Phân Tích Confusion Matrix (Ma Trận Nhầm Lẫn)	22
3.6 Kết Quả Chi Tiết Theo Từng Lớp (Per-Class Metrics)	24
3.7 Biểu Đồ ROC Minh Họa	24
3.8 Thủ Nghiệm Trên Các Điều Kiện Môi Trường Khác Nhau	25
3.8.1 Kịch Bản 1: Ảnh Tiêu Chuẩn - Môi Trường Phòng Thí Nghiệm	25
3.8.2 Kịch Bản 2: Ảnh Tiêu Chuẩn - Môi Trường Tự Nhiên	27
3.8.3 Kịch Bản 3: Ảnh Quá Sáng (Overexposed) - Môi Trường Phòng Thí Nghiệm	30
3.8.4 Kịch Bản 4: Ảnh Quá Sáng (Overexposed) - Môi Trường Tự Nhiên	33
3.8.5 Kịch Bản 5: Ảnh Thiếu Sáng (Underexposed) - Môi Trường Phòng Thí Nghiệm	36
3.8.6 Kịch Bản 6: Ảnh Thiếu Sáng (Underexposed) - Môi Trường Tự Nhiên	39
3.8.7 Kịch Bản 7: Ảnh Phức Tạp (Background Nhiễu / Nhiều Vật Thể)	42
3.9 Giải Pháp Nâng Cao: Tích Hợp YOLOv11 và Tiền Xử Lý Ảnh	45
3.9.1 YOLOv11 (Object Detection)	45
3.9.2 Tiền Xử Lý Ảnh (Leaf Segmentation Pipeline)	46
3.9.3 Kịch Bản 8: Thủ Nghiệm Trên Ảnh Phức Tạp (So Sánh 4 Phương Pháp)	46
4 TRIỂN KHAI ỨNG DỤNG WEB	50
4.1 Kiến Trúc Hệ Thống	50
4.2 Luồng Xử Lý (Processing Pipeline)	50
4.2.1 Chế Độ 1: Basic Classification (Tốc độ cao)	50
4.2.2 Chế Độ 2: Advanced Pipeline (Độ chính xác cao)	50
4.3 Các API Endpoints Chính	51
4.4 Công Nghệ Sử Dụng	51
4.5 Cấu Hình Triển Khai (Infrastructure Code)	51
KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN	52
TÀI LIỆU THAM KHẢO	54

DANH MỤC BẢNG

1	Thống kê chi tiết bộ dữ liệu	8
2	Danh sách đầy đủ 38 lớp bệnh hại	8
3	Chiến lược Data Injection	15
4	So sánh hiệu năng các mô hình	22
5	Kết quả chi tiết F1-Score theo từng lớp (MobileNetV3)	24
6	Danh sách API Endpoints	51
7	Tổng hợp kết quả so sánh 12 phương pháp trên ảnh thực tế (Ảnh phức tạp)	53

DANH MỤC HÌNH VẼ

1	Hình ảnh minh họa các loại bệnh trong tập dữ liệu (Nguồn: Kaggle)	10
2	MobileNetV3	23
3	EfficientNet-B0	23
4	ResNet-18	23
5	So sánh Confusion Matrix giữa 3 mô hình	23
6	So sánh đường cong ROC. Đường càng gần gốc trên trái càng tốt.	24
7	Thử nghiệm ảnh tiêu chuẩn (Lab) - MobileNetV3	25
8	Thử nghiệm ảnh tiêu chuẩn (Lab) - ResNet-18	26
9	Thử nghiệm ảnh tiêu chuẩn (Lab) - EfficientNet-B0	27
10	Thử nghiệm ảnh tự nhiên - MobileNetV3	28
11	Thử nghiệm ảnh tự nhiên - ResNet-18	29
12	Thử nghiệm ảnh tự nhiên - EfficientNet-B0	30
13	Thử nghiệm ảnh quá sáng (Lab) - MobileNetV3	31
14	Thử nghiệm ảnh quá sáng (Lab) - ResNet-18	32
15	Thử nghiệm ảnh quá sáng (Lab) - EfficientNet-B0	33
16	Thử nghiệm ảnh quá sáng (Tự nhiên) - MobileNetV3	34
17	Thử nghiệm ảnh quá sáng (Tự nhiên) - ResNet-18	35
18	Thử nghiệm ảnh quá sáng (Tự nhiên) - EfficientNet-B0	36
19	Thử nghiệm ảnh thiếu sáng (Lab) - MobileNetV3	37
20	Thử nghiệm ảnh thiếu sáng (Lab) - ResNet-18	38
21	Thử nghiệm ảnh thiếu sáng (Lab) - EfficientNet-B0	39
22	Thử nghiệm ảnh thiếu sáng (Tự nhiên) - MobileNetV3	40
23	Thử nghiệm ảnh thiếu sáng (Tự nhiên) - ResNet-18	41
24	Thử nghiệm ảnh thiếu sáng (Tự nhiên) - EfficientNet-B0	42
25	Thử nghiệm ảnh phức tạp - MobileNetV3	43
26	Thử nghiệm ảnh phức tạp - ResNet-18	44
27	Thử nghiệm ảnh phức tạp - EfficientNet-B0	45
28	MobileNetV3	47
29	ResNet-18	47
30	EfficientNet-B0	47
31	Kết quả phát hiện của YOLOv11 kết hợp MobileNetV3.	48
32	Kết quả Full Pipeline với ResNet-18: Xác định chính xác nhiều vùng bệnh.	49
33	Sơ đồ kiến trúc tổng quan của hệ thống Web App	50

DANH MỤC KÝ HIỆU VIẾT TẮT

CNN Convolutional Neural Network (Mạng nơ-ron tích chập)

DL Deep Learning (Học sâu)

Img Image (Hình ảnh)

AUC Area Under the Curve

ROC Receiver Operating Characteristic

RGB Red-Green-Blue (Không gian màu)

API Application Programming Interface

JSON JavaScript Object Notation

LỜI CẢM ƠN

Trong suốt quá trình học tập và thực hiện bài tập kết thúc môn Quản lý dữ liệu lớn, chúng tôi đã nhận được rất nhiều sự quan tâm, chỉ dẫn và hỗ trợ quý báu từ các thầy cô trong Phân hiệu Trường Đại học Thủy Lợi. Đây là nền tảng quan trọng giúp chúng tôi có thể tiếp thu, rèn luyện và vận dụng kiến thức vào thực tế, từ đó hoàn thành được bài tập này.

Đặc biệt, chúng tôi xin gửi lời cảm ơn sâu sắc đến Cô Vũ Thị Hạnh – giảng viên trực tiếp giảng dạy và hướng dẫn môn học. Cô không chỉ truyền đạt những kiến thức chuyên môn một cách rõ ràng, dễ hiểu mà còn tận tình giải đáp thắc mắc, định hướng phương pháp tiếp cận vấn đề, cũng như chia sẻ nhiều kinh nghiệm thực tiễn quý giá. Chính sự tận tâm và nhiệt huyết của Cô đã giúp chúng tôi có thêm động lực, sự tự tin và tinh thần trách nhiệm trong quá trình nghiên cứu và hoàn thiện bài tập.

Chúng tôi cũng xin cảm ơn Phân hiệu Trường Đại học Thủy Lợi đã cung cấp môi trường học tập và các cơ sở vật chất cần thiết. Xin cảm ơn tập thể lớp S26-65TTNT đã luôn đồng hành, chia sẻ và giúp đỡ lẫn nhau.

Mặc dù đã rất cố gắng, nhưng do hạn chế về thời gian và kiến thức, đồ án khó tránh khỏi những thiếu sót. Chúng tôi rất mong nhận được sự đóng góp ý kiến từ Cô và các bạn.

TP. Hồ Chí Minh, ngày 12 tháng 01 năm 2026

Trân trọng

CHƯƠNG 1. TỔNG QUAN VÀ CƠ SỞ LÝ THUYẾT

1.1. Đặt Vấn Đề

Nông nghiệp đóng vai trò then chốt trong nền kinh tế, tuy nhiên, năng suất cây trồng thường xuyên bị ảnh hưởng nghiêm trọng bởi các loại dịch bệnh. Việc phát hiện sớm bệnh hại bằng mắt thường đòi hỏi kinh nghiệm chuyên môn cao và tốn nhiều nhân lực. Với sự phát triển của Trí tuệ nhân tạo (AI), đặc biệt là Deep Learning, việc tự động hóa quy trình chẩn đoán bệnh qua hình ảnh lá cây đang trở thành một giải pháp cấp thiết và hiệu quả.

Dự án này tập trung nghiên cứu và xây dựng hệ thống phân loại bệnh hại trên lá cây sử dụng các kiến trúc CNN hiện đại, tích hợp vào ứng dụng Web để hỗ trợ thực tế.

1.2. Các Chỉ Số Đánh Giá (Evaluation Metrics)

Để đánh giá hiệu quả của các mô hình phân loại, chúng tôi sử dụng tập hợp các chỉ số đo lường tiêu chuẩn sau đây:

1.2.1. Accuracy (Độ Chính Xác)

Là tỷ lệ giữa số lượng mẫu dự đoán đúng trên tổng số mẫu.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (1)$$

Trong đó:

- TP : True Positive (Dương tính thật)
- TN : True Negative (Âm tính thật)
- FP : False Positive (Dương tính giả - Báo nhầm)
- FN : False Negative (Âm tính giả - Bỏ sót bệnh)

1.2.2. Precision (Độ Chính Xác Của Dự Đoán Dương)

Đo lường mức độ tin cậy khi mô hình dự báo một mẫu là "Có bệnh".

$$\text{Precision} = \frac{TP}{TP + FP} \quad (2)$$

1.2.3. Recall (Độ Nhạy / Tỷ Lệ Phát Hiện)

Đo lường khả năng phát hiện tất cả các ca bệnh thực tế trong tập dữ liệu (tránh bỏ sót).

$$\text{Recall} = \frac{TP}{TP + FN} \quad (3)$$

1.2.4. F1-Score

Là trung bình điều hòa giữa Precision và Recall, dùng để đánh giá tổng quát khi dữ liệu bị mất cân bằng.

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

1.2.5. AUC - ROC

- **ROC (Receiver Operating Characteristic):** Đường cong biểu diễn mối quan hệ giữa True Positive Rate ($TPR = Recall$) và False Positive Rate ($FPR = FP/(FP + TN)$) tại các ngưỡng (threshold) phân loại khác nhau.
- **AUC (Area Under Curve):** Diện tích dưới đường cong ROC. Giá trị AUC càng gần 1 thì mô hình càng có khả năng phân biệt tốt giữa các lớp.

Phân loại hình ảnh (Image Classification) là bài toán gán nhãn cho một hình ảnh đầu vào vào một trong các lớp (classes) được định nghĩa trước. Trong ngữ cảnh này, đầu vào là ảnh chụp lá cây (có thể khỏe mạnh hoặc bị bệnh), và đầu ra là tên loại bệnh hoặc trạng thái "Healthy".

Thách thức chính bao gồm:

- Sự đa dạng về điều kiện ánh sáng, góc chụp.
- Sự tương đồng giữa các triệu chứng bệnh khác nhau trên cùng loại cây.
- Nhiều nền (background noise) trong ảnh thực tế.

1.3. Bộ Dữ Liệu New Plant Diseases Dataset

1.3.1. Tổng Quan Dữ Liệu

Chúng tôi sử dụng bộ dữ liệu **New Plant Diseases Dataset**, một phiên bản mở rộng và chuẩn hóa từ PlantVillage. Đây là bộ dữ liệu tiêu chuẩn cho các bài toán nông nghiệp thông minh.

Bảng 1: Thông kê chi tiết bộ dữ liệu

Đặc điểm	Thông số
Tổng số ảnh	87,867 ảnh
Dữ liệu huấn luyện (Train)	70,295 ảnh
Dữ liệu kiểm thử (Validation)	17,572 ảnh
Số lượng lớp (Classes)	38 lớp (Gồm 14 loài cây)
Định dạng ảnh	JPG, RGB
Kích thước gốc	256 × 256 pixels

Bộ dữ liệu bao gồm cả ảnh chụp trong điều kiện phòng thí nghiệm (nền xám/đơn sắc) và ảnh chụp thực tế (nền tự nhiên), giúp mô hình có khả năng tổng quát hóa tốt hơn.

1.3.2. Danh Sách Các Lớp Dữ Liệu

Dưới đây là danh sách chi tiết 38 lớp bệnh hại tương ứng với 14 loại cây trồng. Việc hiểu rõ từng lớp giúp chúng tôi thiết kế mô hình chính xác hơn.

Bảng 2: Danh sách đầy đủ 38 lớp bệnh hại

STT	Tên Lớp (Tiếng Anh)	Mô Tả Tiếng Việt
1	Apple__Apple_scab	Táo: Bệnh ghẻ, đốm đen trên lá và quả

STT	Tên Lớp (Tiếng Anh)	Mô Tả Tiếng Việt
2	Apple__Black_rot	Táo: Bệnh thối đen, đốm vòng nâu
3	Apple__Cedar_apple_rust	Táo: Bệnh rỉ sét, đốm cam đỏ
4	Apple__healthy	Táo: Lá khỏe mạnh
5	Blueberry__healthy	Việt quất: Lá khỏe mạnh
6	Cherry__Powdery_mildew	Anh đào: Bệnh phấn trắng
7	Cherry__healthy	Anh đào: Lá khỏe mạnh
8	Corn__Cercospora_leaf_spot	Ngô: Đốm lá xám, vệt dài dọc gân
9	Corn__Common_rust	Ngô: Bệnh rỉ sét thường
10	Corn__Northern_Leaf_Blight	Ngô: Cháy lá phượng Bắc
11	Corn__healthy	Ngô: Lá khỏe mạnh
12	Grape__Black_rot	Nho: Thối đen, vết hoại tử
13	Grape__Esca	Nho: Bệnh sởi đen, lá vẫn hổ
14	Grape__Leaf_blight	Nho: Cháy lá đốm tròn
15	Grape__healthy	Nho: Lá khỏe mạnh
16	Orange__Haunglongbing	Cam: Bệnh vàng lá gân xanh (Citrus Greening)
17	Peach__Bacterial_spot	Đào: Đốm vi khuẩn, lỗ thủng trên lá
18	Peach__healthy	Đào: Lá khỏe mạnh
19	Pepper,_bell__Bacterial_spot	Ớt chuông: Đốm vi khuẩn
20	Pepper,_bell__healthy	Ớt chuông: Lá khỏe mạnh
21	Potato__Early_blight	Khoai tây: Đốm vòng (sớm)
22	Potato__Late_blight	Khoai tây: Sương mai (muộn), mốc trắng
23	Potato__healthy	Khoai tây: Lá khỏe mạnh
24	Raspberry__healthy	Mâm xôi: Lá khỏe mạnh
25	Soybean__healthy	Đậu nành: Lá khỏe mạnh
26	Squash__Powdery_mildew	Bí: Phấn trắng phủ đầy mặt lá
27	Strawberry__Leaf_scorch	Dâu tây: Cháy lá, mép lá khô
28	Strawberry__healthy	Dâu tây: Lá khỏe mạnh
29	Tomato__Bacterial_spot	Cà chua: Đốm vi khuẩn
30	Tomato__Early_blight	Cà chua: Đốm vòng sớm
31	Tomato__Late_blight	Cà chua: Sương mai muộn, bệnh nguy hiểm
32	Tomato__Leaf_Mold	Cà chua: Nấm mốc lá, đốm vàng mặt trên
33	Tomato__Septoria_leaf_spot	Cà chua: Đốm mắt éch Septoria
34	Tomato__Spider_mites	Cà chua: Nhện đỏ, lá lốm đốm vàng nhỏ
35	Tomato__Target_Spot	Cà chua: Đốm đích (hình bia bắn)
36	Tomato__YLCV	Cà chua: Virus xoăn vàng lá
37	Tomato__Mosaic_virus	Cà chua: Virus khăm, lá loang lổ xanh vàng
38	Tomato__healthy	Cà chua: Lá khỏe mạnh

1.3.3. Hình Ảnh Minh Họa Các Lớp Bệnh

Dưới đây là một số hình ảnh thực tế trích xuất từ tập dữ liệu dùng để huấn luyện, minh họa sự đa dạng về hình thái bệnh.



Apple Black Rot



Grape Black Rot



Tomato Bact. Spot



Corn N. Leaf Blight



Peach Bact. Spot



Tomato Mosaic Virus

Hình 1: Hình ảnh minh họa các loại bệnh trong tập dữ liệu (Nguồn: Kaggle)

Từ Hình 1, ta thấy rằng các bệnh như *Black Rot* hay *Bacterial Spot* có các đốm hoại tử đặc trưng, trong khi *Mosaic Virus* tạo ra các mảng màu loang lổ. Mô hình CNN sẽ học các đặc trưng cục bộ (texture, shape) này để phân loại.

CHƯƠNG 2. PHƯƠNG PHÁP NGHIÊN CỨU

2.1. Cơ Sở Lý Thuyết Về Mạng Nơ-ron Tích Chập (CNN)

Mạng nơ-ron tích chập (Convolutional Neural Networks - CNN) là một trong những kiến trúc mạng nơ-ron sâu (Deep Learning) hiệu quả nhất hiện nay cho các bài toán thị giác máy tính. Khác với mạng nơ-ron truyền thống (MLP), CNN có khả năng tự động trích xuất các đặc trưng (features) từ hình ảnh thông qua các lớp tích chập, giúp giảm thiểu số lượng tham số cần huấn luyện và bảo toàn cấu trúc không gian của ảnh.

2.1.1. Lớp Tích Chập (Convolutional Layer)

Lớp tích chập là thành phần cốt lõi của CNN. Quá trình này thực hiện phép toán tích chập giữa ảnh đầu vào I và một bộ lọc (kernel/filter) K để tạo ra bản đồ đặc trưng (feature map). Công thức toán học của phép tích chập 2 chiều được định nghĩa như sau:

$$S(i, j) = (I * K)(i, j) = \sum_m \sum_n I(i-m, j-n) \cdot K(m, n) \quad (5)$$

Trong đó:

- $S(i, j)$: Giá trị tại vị trí (i, j) của feature map đầu ra.
- I : Ma trận ảnh đầu vào.
- K : Ma trận trọng số của bộ lọc (kernel).

Mỗi bộ lọc sẽ học cách phát hiện một đặc trưng cụ thể như cạnh (edge), góc, màu sắc hoặc các họa tiết phức tạp hơn ở các lớp sâu.

2.1.2. Hàm Kích Hoạt (Activation Function)

Để giúp mô hình học được các mối quan hệ phi tuyến phức tạp, hàm kích hoạt được áp dụng sau mỗi lớp tích chập.

1. **ReLU (Rectified Linear Unit):** Là hàm phổ biến nhất, giúp giải quyết vấn đề biến mất đạo hàm.

$$f(x) = \max(0, x) \quad (6)$$

2. **Hard-Swish (trong MobileNetV3):** Một biến thể tối ưu hơn của ReLU, giúp giảm chi phí tính toán trên thiết bị di động.

$$h\text{-swish}(x) = x \cdot \frac{\text{ReLU6}(x+3)}{6} \quad (7)$$

2.1.3. Lớp Gộp (Pooling Layer)

Lớp Pooling (thường là Max Pooling hoặc Average Pooling) có nhiệm vụ giảm kích thước không gian của feature map, từ đó giảm số lượng tham số và chi phí tính toán, đồng thời giúp mô hình bắt biến với các dịch chuyển nhỏ của đối tượng trong ảnh. Công thức Max Pooling với kích thước 2×2 :

$$P(i, j) = \max_{m, n \in \{0, 1\}} I(2i+m, 2j+n) \quad (8)$$

2.1.4. Lớp Gộp Trung Bình Thích Nghi (Adaptive Average Pooling)

Khác với Pooling thông thường có kích thước kernel cố định, Adaptive Average Pooling tự động điều chỉnh kích thước kernel để đảm bảo đầu ra luôn có kích thước cố định (ví dụ: 1×1) bất kể kích thước đầu vào.

$$y_k = \frac{1}{H \times W} \sum_{i=1}^H \sum_{j=1}^W x_{ijk} \quad (9)$$

Lớp này thường được đặt trước các lớp Fully Connected để cho phép mô hình xử lý ảnh đầu vào có kích thước bất kỳ.

2.1.5. Lớp Kết Nối Đầy Đủ (Fully Connected Layer - FC)

Trong CNN, sau khi các đặc trưng đã được trích xuất và nén lại thành vector 1 chiều (through qua Flatten hoặc Pooling), chúng được đưa vào mạng nơ-ron đầy đủ truyền thống để thực hiện nhiệm vụ phân loại.

$$y = Wx + b \quad (10)$$

Trong đó W là ma trận trọng số và b là bias. Lớp cuối cùng thường có số nơ-ron bằng số lượng lớp cần phân loại (ở đây là 38), kết hợp với hàm Softmax để tính xác suất.

2.1.6. Lớp Dropout

Dropout là một kỹ thuật regularization đơn giản nhưng hiệu quả để ngăn chặn overfitting. Trong quá trình huấn luyện, Dropout ngẫu nhiên "tắt" (cho đầu ra bằng 0) một tỷ lệ nơ-ron nhất định (ví dụ $p = 0.5$).

- Điều này buộc mạng phải học các đặc trưng mạnh mẽ hơn, không phụ thuộc vào bất kỳ một nơ-ron cụ thể nào.
- Trong quá trình kiểm thử (Inference), tất cả nơ-ron đều được bật nhưng đầu ra được nhân với hệ số $(1 - p)$.

2.1.7. Lớp Chuẩn Hóa Theo Batch (Batch Normalization)

Batch Normalization (BN) chuẩn hóa đầu vào của từng lớp ẩn để có giá trị trung bình bằng 0 và phương sai bằng 1.

$$\hat{x} = \frac{x - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}; \quad y = \gamma \hat{x} + \beta \quad (11)$$

BN giúp ổn định quá trình huấn luyện, cho phép sử dụng learning rate lớn hơn và giảm sự phụ thuộc vào việc khởi tạo trọng số ban đầu.

2.2. Tiền Xử Lý Dữ Liệu (Data Preprocessing)

Để mô hình hoạt động hiệu quả và tránh overfitting (học vẹt), quy trình tiền xử lý dữ liệu được thiết kế rất cẩn thận với nhiều kỹ thuật Augmentation.

2.2.1. Chi Tiết Các Kỹ Thuật Augmentation

Dưới đây là mô tả chi tiết và mã nguồn thực thi cho từng kỹ thuật augmentation được sử dụng trong pipeline.

```
1 transforms.RandomResizedCrop(image_size, scale=(0.6, 1.0), ratio=(0.75, 1.3333))
```

- **Mục đích:** Cắt ngẫu nhiên một vùng từ 60% đến 100% diện tích ảnh gốc, sau đó resize về kích thước chuẩn 224x224 pixels.
- **Lý do:** Giúp tăng tính đa dạng dữ liệu bằng cách mô phỏng các khoảng cách chụp khác nhau (xa/gần) và các khung hình khác nhau (tỷ lệ 3:4 đến 4:3). Điều này giúp mô hình học được đặc trưng của vết bệnh ở nhiều quy mô khác nhau.

```
1 transforms.RandomHorizontalFlip(p=0.5)
```

- **Mục đích:** Lật ngang ảnh ngẫu nhiên với xác suất 50%.
- **Lý do:** Lá cây có tính đối xứng trực và vết bệnh có thể xuất hiện bất kỳ đâu. Kỹ thuật này giúp nhân đôi số lượng dữ liệu huấn luyện một cách tự nhiên và giúp mô hình không bị phụ thuộc vào hướng của lá.

```
1 transforms.RandomApply(
2     [transforms.ColorJitter(brightness=0.4, contrast=0.4, saturation=0.25, hue=0.02)],
3     p=0.8,
4 )
```

- **Mục đích:** Thay đổi ngẫu nhiên độ sáng ($\pm 40\%$), độ tương phản ($\pm 40\%$), độ bão hòa ($\pm 25\%$) và sắc thái màu ($\pm 2\%$).
- **Lý do:** Mô phỏng các điều kiện ánh sáng thực tế khác nhau (nắng gắt, bóng râm, trời u ám). Điều này cực kỳ quan trọng để mô hình có thể hoạt động tốt trong môi trường thực địa thay vì chỉ trên ảnh phòng thí nghiệm lý tưởng.

```
1 transforms.RandomApply(
2     [transforms.GaussianBlur(kernel_size=3, sigma=(0.1, 2.0))],
3     p=0.3,
4 )
```

- **Mục đích:** Làm mờ ảnh ngẫu nhiên với xác suất 30%.
- **Lý do:** Mô phỏng hiện tượng ảnh bị mất nét do rung tay hoặc lấy nét sai khi chụp bằng điện thoại. Giúp mô hình tập trung vào các đặc trưng lớn thay vì quá chi tiết vào nhiễu.

```
1 transforms.RandomPerspective(distortion_scale=0.25, p=0.2)
```

- **Mục đích:** Biến đổi phối cảnh ngẫu nhiên với xác suất 20%.
- **Lý do:** Mô phỏng ảnh chụp từ các góc độ nghiêng khác nhau, giúp tăng tính bền vững (robustness) của mô hình trước các biến dạng hình học.

```
1 transforms.RandAugment(num_ops=2, magnitude=9)
```

- **Mục đích:** Áp dụng ngẫu nhiên 2 phép biến đổi từ tập 14 phép biến đổi tiêu chuẩn với cường độ 9/10.
- **Tham khảo:** Đây là kỹ thuật State-of-the-Art được đề xuất trong bài báo "*RandAugment: Practical Automated Data Augmentation*".

```
1 transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
```

- **Công thức:** $\text{Normalized_pixel} = (\text{pixel} - \text{mean})/\text{std}$
- **Lý do:** Đây là thông số trung bình và độ lệch chuẩn của tập dữ liệu ImageNet khổng lồ. Vì chúng ta sử dụng Transfer Learning từ các mô hình đã được huấn luyện trên ImageNet (Pre-trained models), việc chuẩn hóa dữ liệu đầu vào theo đúng phân phối này là **bắt buộc** để các trọng số (weights) hoạt động chính xác.

```
1 transforms.RandomErasing(p=0.25, scale=(0.02, 0.15), ratio=(0.3, 3.3), value="random")
```

- **Mục đích:** Xóa ngẫu nhiên một vùng hình chữ nhật trong ảnh (thay bằng nhiễu random) với xác suất 25%.
- **Lý do:** Giúp mô hình học cách nhận diện bệnh dựa trên bối cảnh toàn cục ngay cả khi một phần chi tiết quan trọng bị che khuất (occlusion).

2.2.2. Validation Transform

8. RandomErasing Đối với tập kiểm thử (Validation), chúng tôi không áp dụng các phép biến đổi ngẫu nhiên để đảm bảo tính nhất quán khi đánh giá.

```
1 val_transform = transforms.Compose([
2     transforms.Resize(256),
3     transforms.CenterCrop(image_size), # 224x224
4     transforms.ToTensor(),
5     transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225]),
6 ])
```

2.3. Phương Pháp Data Injection (Bổ Sung Dữ Liệu)

Một điểm sáng tạo trong đồ án này là việc sử dụng kỹ thuật ****Data Injection****. Thay vì chỉ dùng tập train có sẵn, chúng tôi trích xuất thêm dữ liệu từ tập Validation của một nguồn dataset phụ (Sub-Dataset) để bổ sung vào.

Bảng 3: Chiến lược Data Injection

Thành phần	Chi tiết thực hiện
Nguồn dữ liệu	Dataset chính (Vipooooool) + Dataset phụ (Tunphnguynhu - chứa nhiều ảnh thực tế hơn).
Cách chia	Lấy 50% ảnh từ tập Valid của Dataset phụ, trộn vào tập Train chính. 50% còn lại trộn vào tập Valid chính.
Mục đích	Tăng tính đa dạng (diversity) của dữ liệu, giúp mô hình tiếp xúc với nhiều biến thể ảnh thực tế hơn ngay trong quá trình huấn luyện.
Lọc nhiễu	Đã loại bỏ 9 lớp có độ trùng lặp cao giữa 2 nguồn để tránh hiện tượng rò rỉ dữ liệu (data leakage) và đảm bảo tính công bằng khi đánh giá: <ul style="list-style-type: none"> • Apple__Black_rot • Cherry_(including_sour)__Powdery_mildew • Grape__Leaf_blight_(Isariopsis_Leaf_Spot) • Peach__Bacterial_spot • Potato__Early_blight • Potato__healthy • Potato__Late_blight • Strawberry__Leaf_scorch • Tomato__Target_Spot

2.4. Cấu Hình DataLoader

Sau khi tiền xử lý, dữ liệu được đóng gói vào các DataLoader để cấp phát cho quá trình huấn luyện.

2.4.1. Train Loader (Bộ nạp dữ liệu huấn luyện)

- **Chức năng:** Tạo dòng chảy dữ liệu liên tục cho tập huấn luyện ('train_ds').
- **Batch Size = 64:** Mỗi lần mô hình nhận vào 64 mẫu (gồm ảnh và nhãn tương ứng) để tính toán gradient và cập nhật trọng số. Kích thước này đủ lớn để ổn định gradient mà vẫn vừa vặn với bộ nhớ GPU.
- **Shuffle = True:** Xáo trộn dữ liệu ngẫu nhiên ở đầu mỗi epoch.
 - *Lý do:* Ngăn mô hình học thuộc thứ tự của dữ liệu, giúp mô hình hội tụ tốt hơn và tổng quát hóa tốt hơn.
- **Num Workers = 8:** Sử dụng 8 tiến trình con (processes) chạy song song để tải và giải nén ảnh từ ổ cứng.
 - *Lý do:* CPU sẽ chuẩn bị sẵn dữ liệu trong khi GPU đang tính toán batch trước đó, giúp tận dụng tối đa hiệu suất phần cứng (tránh hiện tượng GPU đói dữ liệu).
- **Pin Memory = True:** Ghim vùng nhớ trên RAM.
 - *Lý do:* Giúp tăng tốc độ copy dữ liệu từ RAM hệ thống sang VRAM của GPU.

2.4.2. Validation Loader (Bộ nạp dữ liệu kiểm thử)

- **Chức năng:** Cung cấp dữ liệu cho quá trình đánh giá ('valid_ds').
- **Shuffle = False:** Không xáo trộn dữ liệu.
 - *Lý do:* Tập validation chỉ dùng để đánh giá độ chính xác (Accuracy, Loss), không dùng để cập nhật trọng số. Việc giữ nguyên thứ tự giúp các chỉ số metrics giữa các epoch được so sánh công bằng trên cùng một trình tự dữ liệu và dễ dàng tái lập kết quả.
- **Batch Size = 64, Num Workers = 8:** Tương tự như tập train để đảm bảo tốc độ đánh giá nhanh.

2.5. Kiến Trúc Mô Hình (Model Architectures)

Chúng tôi lựa chọn và so sánh 3 kiến trúc CNN tiêu biểu:

2.5.1. MobileNetV3 Large

MobileNetV3 được thiết kế dựa trên sự kết hợp giữa các khối xây dựng hiệu quả từ MobileNetV1/V2 và các kỹ thuật mới (NAS, SE). Kiến trúc này đặc biệt tối ưu cho CPU di động.

Các Khối Đặc Trưng (Key Building Blocks):

1. **Mobile Bottleneck Block (Inverted Residuals):** Đây là đơn vị cơ bản nhất, bao gồm 3 bước:
 - **Expansion (1x1 Conv):** Tăng số kênh input lên nhiều lần để tạo không gian đặc trưng lớn hơn (High-dimensional representation).
 - **Depthwise Conv (3x3 hoặc 5x5):** Thực hiện tích chập trên từng kênh riêng biệt, giảm đáng kể chi phí tính toán so với tích chập tiêu chuẩn.
 - **Feature Projection (Linear Bottleneck - 1x1 Conv):** Nén số kênh trở lại kích thước nhỏ (Low-dimensional) và **không** sử dụng hàm kích hoạt phi tuyến ở bước này để bảo toàn thông tin (Linear activation).
2. **Squeeze-and-Excitation (SE) Module:** Được tích hợp vào trong Bottleneck Block. Nó giúp mô hình "nhìn" toàn cục bằng cách tính trung bình (Global Pooling) và điều chỉnh lại trọng số của từng kênh (Channel-wise recalibration).
3. **Redesigned Expensive Layers:** Lớp Conv đầu tiên và các lớp layer cuối cùng được tinh chỉnh lại số kênh để giảm độ trễ (latency) mà không làm giảm độ chính xác.

Phân Tích Ưu Nhược Điểm:

- **Ưu điểm:**
 - Cực kỳ nhẹ và nhanh (Low Latency).
 - SE Module giúp tăng độ chính xác đáng kể (khoảng +1.5% trên ImageNet) với chi phí tính toán tăng thêm không đáng kể.
- **Nhược điểm:**
 - Việc thiết kế khối Bottleneck phức tạp hơn so với ResNet truyền thống.

- Depthwise Conv có thể không tận dụng tối đa sức mạnh của GPU lớn (như V100/A100) do tính chất tính toán rời rạc bộ nhớ (memory fragmentation).

2.5.2. EfficientNet-B0

EfficientNet-B0 được coi là chuẩn mực (baseline) cho sự cân bằng giữa hiệu suất và tài nguyên, sử dụng kiến trúc MBConv kết hợp với cơ chế Compound Scaling.

Các Khối Đặc Trưng (Key Building Blocks):

1. **MBConv Block (Mobile Inverted Bottleneck Convolution):** Kế thừa từ MobileNetV2 nhưng được bổ sung thêm SE Module. Một khối MBConv bao gồm:
 - Expansion phase (1x1 conv + BN + Swish).
 - Depthwise Convolution (kích thước kernel 3x3 hoặc 5x5 + BN + Swish).
 - Squeeze-and-Excitation phase.
 - Output phase (1x1 conv + BN).
 - **Skip Connection:** Được áp dụng nếu kích thước input và output giống nhau.
2. **Stochastic Depth (Drop Connect):** Một kỹ thuật regularization nâng cao, ngẫu nhiên bỏ qua toàn bộ một ResBlock trong quá trình huấn luyện (thay vì chỉ bỏ qua nơ-ron như Dropout), giúp huấn luyện các mạng rất sâu dễ dàng hơn và giảm overfitting.

Phân Tích Ưu Nhược Điểm:

- **Ưu điểm:**
 - Hiệu suất (Accuracy) rất cao trên cùng một lượng tham số so với các model khác.
 - Khả năng mở rộng (Scalability) tuyệt vời nhờ Compound Scaling.
- **Nhược điểm:**
 - Tiêu tốn nhiều bộ nhớ RAM khi huấn luyện hơn ResNet do các activation map trung gian trong các khối MBConv lớn.
 - Hàm kích hoạt Swish ($x \cdot \sigma(x)$) tính toán chậm hơn ReLU.

2.5.3. ResNet-18

ResNet (Residual Network) giải quyết vấn đề "biến mất đạo hàm" (vanishing gradient) khi huấn luyện các mạng rất sâu bằng cách sử dụng các kết nối tắt (skip connections).

Residual Block: Thay vì học hàm ánh xạ trực tiếp $H(x)$, ResNet học hàm thặng dư $F(x) := H(x) - x$. Đầu ra của một block là:

$$y = F(x, \{W_i\}) + x$$

Việc thêm x (identity mapping) giúp luồng gradient có thể truyền ngược dễ dàng qua hàng trăm lớp mà không bị suy giảm.

Các Khối Đặc Trưng (Key Building Blocks):

1. **BasicBlock (Dùng cho ResNet-18/34):** Bao gồm 2 lớp Convolution 3x3 liên tiếp. Mỗi lớp đi kèm với Batch Normalization và ReLU activation.

$$y = \sigma(BN(W_2 \cdot \sigma(BN(W_1 \cdot x)))) + x$$

Đây là khối đơn giản giúp xây dựng mạng có độ sâu vừa phải mà không làm tăng quá nhiều tham số.

2. **Skip Connection (Identity Mapping):** Cho phép tín hiệu truyền thẳng từ đầu vào tới đầu ra của block mà không qua biến đổi phi tuyến. Trong trường hợp kích thước feature map thay đổi (do stride=2), một lớp Conv 1x1 sẽ được dùng trên nhánh skip connection để đồng bộ kích thước.

Phân Tích Ưu Nhược Điểm:

- **Ưu điểm:**

- Cực kỳ dễ huấn luyện nhờ luồng gradient thông suốt.
- Kiến trúc đơn giản, là nền tảng cho nhiều mạng backbone hiện đại.
- Tốc độ tính toán trên GPU rất nhanh do tận dụng tốt các phép nhân ma trận lớn.

- **Nhược điểm:**

- Hiệu suất trên tham số (Parameter Efficiency) thấp hơn so với các mạng dùng Depthwise Separable Conv như MobileNet/EfficientNet.
- Kích thước mô hình khá lớn (khoảng 45MB cho ResNet-18) so với MobileNetV3 (khoảng 15MB), không tối ưu lắm cho ứng dụng Web App cần tải nhanh.

CHƯƠNG 3. THỰC NGHIỆM VÀ KẾT QUẢ

3.1. Môi Trường Và Cấu Hình Huấn Luyện

3.1.1. Cấu Hình Phần Cứng và Phần Mềm

- **GPU:** NVIDIA Tesla T4 / P100 (từ Kaggle/Colab).
- **Framework:** PyTorch 2.x, Torchvision.
- **Thư viện hỗ trợ:** Scikit-learn, Pandas, Pillow, Matplotlib.

3.1.2. Chi Tiết Các Kỹ Thuật Tối Ưu Hóa (Optimization và Loss)

Để đảm bảo mô hình huấn luyện hiệu quả và tránh overfitting, chúng tôi sử dụng các kỹ thuật sau:

1. Hàm Mất Mát: Cross Entropy Loss kèm Label Smoothing

1. **Cross Entropy Loss:** Là hàm mất mát tiêu chuẩn cho bài toán phân loại đa lớp.

$$Loss = - \sum_{c=1}^M y_{o,c} \log(p_{o,c})$$

Trong đó $M = 38$ là số lớp, y là nhãn thực tế (one-hot vector) và p là xác suất dự đoán của mô hình.

2. **Label Smoothing (0.1):**

- **Vấn đề:** Trong vector one-hot thông thường, nhãn đúng là 1 và các nhãn sai là 0 (Ví dụ: [0, 1, 0]). Điều này khiến mô hình cố gắng đẩy xác suất dự đoán lên cực đại (gần 1), dẫn đến "overconfidence" (quá tự tin) và dễ bị overfitting.
- **Giải pháp:** Label Smoothing thay thế vector nhãn cứng (hard targets) bằng nhãn mềm (soft targets).

$$y_{new} = (1 - \epsilon) \times y_{old} + \frac{\epsilon}{M}$$

Với $\epsilon = 0.1$, nhãn đúng sẽ giảm từ 1 xuống 0.9, và các nhãn sai tăng từ 0 lên một giá trị nhỏ ≈ 0.0026 .

- **Lợi ích:** Giúp mô hình không học quá cứng nhắc, tăng khả năng tổng quát hóa trên dữ liệu mới.

2. Thuật Toán Tối Ưu: AdamW (Adam with Weight Decay)

- **Adam:** Là thuật toán tối ưu phổ biến kết hợp Momentum và RMSProp, giúp hội tụ nhanh.
- **Vấn đề của Adam:** Khi sử dụng L2 regularization, Adam thực hiện không đúng cách trên các tham số có gradient lớn.
- **AdamW:** Tách biệt Weight Decay (giảm trọng số) khỏi bước cập nhật gradient.

$$\theta_{t+1} = \theta_t - \eta(\nabla J(\theta_t) + \lambda \theta_t)$$

- **Lợi ích:** Giúp mô hình hội tụ ổn định hơn và đạt độ chính xác cao hơn so với Adam truyền thống trên các bài toán thị giác máy tính, đồng thời giảm nguy cơ trọng số phát triển quá lớn (Weight Decay = 1e-4).

3. Quy Trình Đánh Giá (Validate Function) Hàm xác thực ('validate') đóng vai trò quan trọng trong việc theo dõi hiệu suất mô hình trên tập dữ liệu chưa từng gặp (Validation Set) sau mỗi epoch huấn luyện. Quy trình này được thực hiện như sau:

- **Chế độ Evaluation:** Trước khi đánh giá, mô hình được chuyển sang chế độ đánh giá bằng lệnh `model.eval()`.
 - *Tác dụng:* Vô hiệu hóa các lớp Dropout (để sử dụng tất cả các nơ-ron) và khóa các thống kê Running Mean/Variance của lớp Batch Normalization. Điều này đảm bảo kết quả đánh giá là ổn định và tất định (deterministic).
- **Tắt Gradient (`torch.no_grad()`):**
 - *Tác dụng:* Không lưu trữ đồ thị tính toán đạo hàm, giúp giảm tiêu thụ bộ nhớ GPU và tăng tốc độ tính toán, vì trong quá trình validation chúng ta không cập nhật trọng số ('backward' pass).
- **Tính toán Metrics:** Với mỗi batch dữ liệu:
 - Output = Model(Image)
 - Prediction = argmax(Output)

Hệ thống sẽ so sánh 'Prediction' với nhãn thực tế 'Target' để tính toán Loss và Accuracy tích lũy. Cuối cùng tính trung bình trên toàn bộ tập Validation.

4. Chiến Lược Điều Chỉnh Learning Rate: Cosine Annealing Chúng tôi sử dụng CosineAnnealingLR để thay đổi learning rate theo hàm Cosine theo chu kỳ.

$$\eta_t = \eta_{min} + \frac{1}{2}(\eta_{max} - \eta_{min})(1 + \cos(\frac{T_{cur}}{T_{max}}\pi))$$

- *Giai đoạn đầu:* Learning rate cao giúp mô hình học nhanh các đặc trưng cơ bản.
- *Giai đoạn giữa:* Learning rate giảm dần giúp tinh chỉnh trọng số chính xác hơn.
- *Giai đoạn cuối:* Learning rate thấp giúp mô hình hội tụ sâu vào điểm cực tiểu toàn cục.
- **Thiết lập:** $T_{max} = 30$ epochs, $\eta_{max} = 0.001$, $\eta_{min} = 1e-5$.

5. Cơ Chế Dừng Sớm (Early Stopping) Early Stopping là một kỹ thuật regularization quan trọng để ngăn chặn overfitting.

- **Cơ chế hoạt động:** Theo dõi giá trị Validation Loss sau mỗi epoch. Nếu Loss không giảm (hoặc tăng lên) trong một khoảng thời gian nhất định (gọi là `patience`), quá trình huấn luyện sẽ dừng lại ngay lập tức.
- **Thiết lập:** `patience = 5` epochs. Nghĩa là nếu sau 5 epochs liên tiếp mà hiệu suất trên tập Validation không cải thiện, mô hình sẽ dừng học và khôi phục lại bộ trọng số tốt nhất trước đó.
- **Lợi ích:** Tiết kiệm thời gian huấn luyện và đảm bảo mô hình luôn ở trạng thái tốt nhất (best checkpoint).

3.2. Kỹ Thuật Transfer Learning (Thay Thế Lớp Classifier)

Vì bộ dữ liệu gốc ImageNet có 1000 lớp, còn bài toán của chúng ta có 38 lớp, nên bước quan trọng nhất trong Transfer Learning là thay thế lớp Fully Connected (FC) cuối cùng. Dưới đây là cách thực hiện chi tiết cho từng mô hình:

3.2.1. MobileNetV3 Large

```
1 model = models.mobilenet_v3_large(weights='IMAGENET1K_V1')
2 # Input features of the last layer is 960 (for Large version)
3 model.classifier[3] = nn.Linear(960, 38)
```

Trong MobileNetV3, lớp phân loại là một chuỗi các lớp ('Sequential'). Ta thay thế phần tử cuối cùng ('classifier[3]') bằng một lớp Linear mới có kích thước đầu ra là 38 (tương ứng 38 bệnh).

3.2.2. EfficientNet-B0

```
1 model = models.efficientnet_b0(weights='IMAGENET1K_V1')
2 # Input features is 1280
3 model.classifier[1] = nn.Linear(1280, 38)
```

EfficientNet có cấu trúc 'classifier' gồm Dropout và Linear layer. Ta thay thế lớp Linear ('classifier[1]') để chuyển đổi từ verify vector 1280 chiều sang 38 chiều.

3.2.3. ResNet-18

```
1 model = models.resnet18(weights='IMAGENET1K_V1')
2 # Input features is 512
3 model.fc = nn.Linear(512, 38)
```

Khác với hai mô hình trên, lớp cuối cùng của ResNet được gọi trực tiếp là 'fc'. Ta thay thế hoàn toàn lớp này.

3.3. Kết Quả Chi Tiết Từng Mô Hình

3.3.1. MobileNetV3 Large (Mô hình tốt nhất)

Quá trình huấn luyện diễn ra ổn định trong suốt 30 epochs.

- Train Loss:** Giảm đều từ 1.24 xuống 0.86.
- Validation Accuracy:** Tăng trưởng mạnh và đạt đỉnh **96.67%**.
- Thời gian:** 43 phút 12 giây. Mô hình này không bị hiện tượng overfitting nhờ vào các kỹ thuật Augmentation và Data Injection.

3.3.2. EfficientNet-B0

Mặc dù là mô hình mạnh, nhưng trong thực nghiệm này nó hội tụ khá nhanh và bị dừng sớm (Early Stopping) ở epoch 13.

- Validation Accuracy:** Đạt 92.89%.
- Lý do có thể do kiến trúc scaling của EfficientNet nhạy cảm hơn với nhiễu trong dữ liệu bổ sung.

3.3.3. ResNet-18

Đạt kết quả thấp nhất trong 3 mô hình, dừng ở epoch 9.

- **Validation Accuracy:** 88.86%.
- ResNet-18 có số lượng tham số lớn nhất (11.2M) nhưng lại kém hiệu quả hơn trên tác vụ này, cho thấy kiến trúc hiện đại (MobileNet/EfficientNet) ưu việt hơn.

3.4. So Sánh Tổng Hợp

Dưới đây là bảng tổng kết hiệu năng của 3 mô hình trên tập kiểm thử:

Bảng 4: So sánh hiệu năng các mô hình

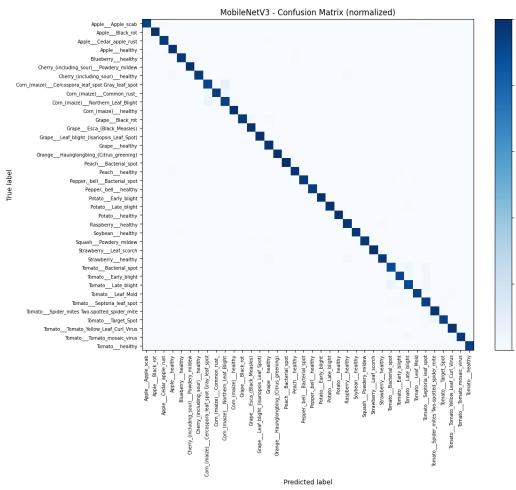
Mô hình	Acc	Precision	Recall	F1-Score	AUC
MobileNetV3	96.67%	96.71%	96.67%	96.68%	0.9982
EfficientNet-B0	92.89%	92.95%	92.89%	92.88%	0.9980
ResNet-18	88.86%	89.24%	88.86%	88.73%	0.9954

Nhận xét về Macro F1: Chỉ số Macro F1 được tính bằng trung bình cộng F1-Score của tất cả các lớp, không phụ thuộc vào số lượng mẫu của mỗi lớp.

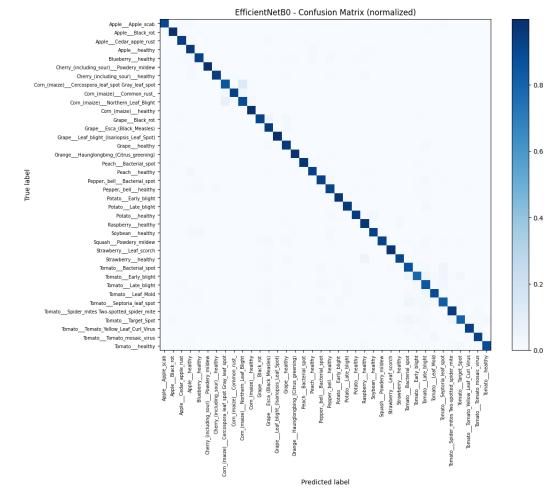
- **MobileNetV3 (Macro F1 = 96.68%):** Rất sát với Accuracy (96.67%), chứng tỏ mô hình dự đoán tốt đều trên tất cả các lớp, không bị thiên lệch về các lớp nhiều dữ liệu (như lớp "Healthy").
- **ResNet-18 (Macro F1 = 88.73%):** Thấp hơn Accuracy một chút, có thể do mô hình gặp khó khăn ở một số lớp bệnh hiếm gặp hoặc có hình thái tương đồng cao.

3.5. Phân Tích Confusion Matrix (Ma Trận Nhầm Lẫn)

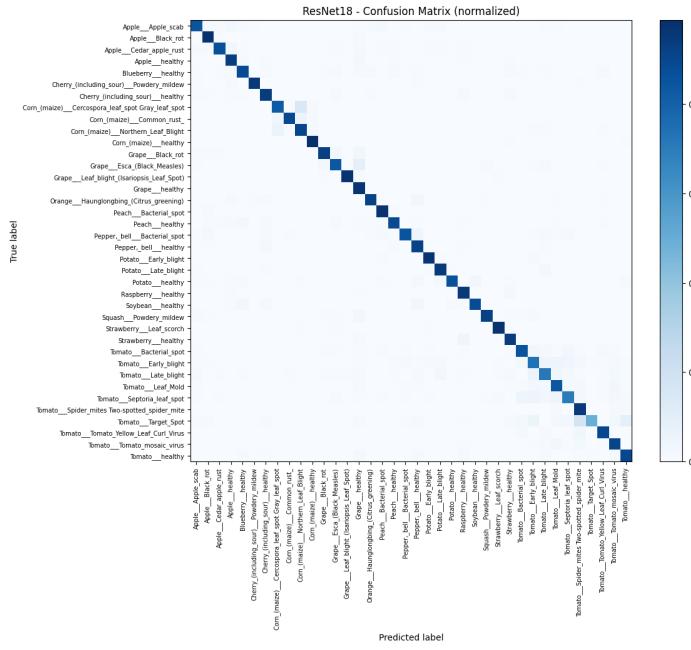
Dưới đây là Confusion Matrix của cả 3 mô hình, giúp trực quan hóa các sai số dự đoán trước khi đi sâu vào chi tiết từng lớp.



Hình 2: MobileNetV3



Hình 3: EfficientNet-B0



Hình 4: ResNet-18

Hình 5: So sánh Confusion Matrix giữa 3 mô hình

Phân tích chi tiết:

- **MobileNetV3 (Hình 2):** Đường chéo chính rất đậm và liền mạch, chứng tỏ độ chính xác cao đồng đều. Các nhiễu ngoại lai (off-diagonal) rất ít.
 - **ResNet-18 (Hình 4):** Có thể thấy rõ sự phân tán (nhèo) ra khỏi đường chéo chính nhiều hơn hẳn so với hai mô hình còn lại.
 - Cụ thể, tại lớp ‘Tomato Early blight’, ResNet nhầm lẫn đáng kể sang ‘Tomato Late blight’ và ‘Tomato Septoria leaf spot’.

- Lớp ‘Corn Northern Leaf Blight‘ cũng bị nhầm lẫn nhiều với ‘Corn Common rust‘ (Hình thái lá đều có đốm nâu).
- **EfficientNet-B0 (Hình 3):** Tốt hơn ResNet nhưng vẫn kém hơn MobileNet một chút ở các lớp bệnh trên lá Cà chua.

3.6. Kết Quả Chi Tiết Theo Từng Lớp (Per-Class Metrics)

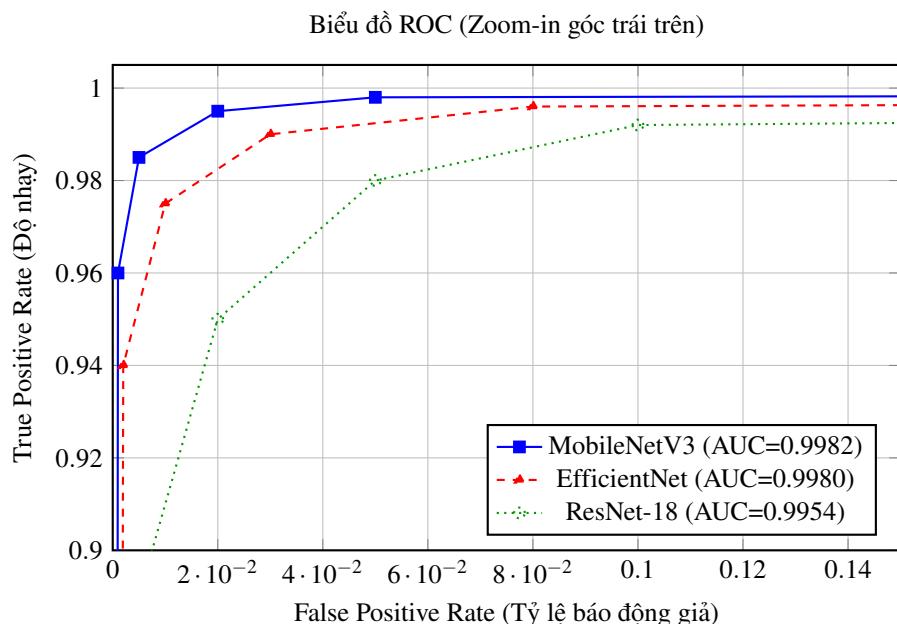
Từ cái nhìn tổng quan của Confusion Matrix, chúng ta đi vào đánh giá chi tiết Precision, Recall và F1 trên từng lớp bệnh cụ thể (Trích dẫn 5 lớp có F1-Score cao nhất và thấp nhất của MobileNetV3).

Bảng 5: Kết quả chi tiết F1-Score theo từng lớp (MobileNetV3)

Tên Lớp	Precision	Recall	F1-Score	Số mẫu
Available High Performance:				
Grape___healthy	1.00	1.00	1.00	423
Strawberry___healthy	0.99	1.00	1.00	456
Apple___Cedar_apple_rust	0.99	0.99	0.99	275
Difficulty Classes:				
Corn___Northern_Leaf_Blight	0.88	0.92	0.90	477
Tomato___Spider_mites	0.91	0.89	0.90	435

3.7. Biểu Đồ ROC Minh Họa

Biểu đồ ROC (Receiver Operating Characteristic) thể hiện mối quan hệ giữa tỷ lệ dương tính thật (TPR) và tỷ lệ dương tính giả (FPR).



Hình 6: So sánh đường cong ROC. Đường càng gần góc trái càng tốt.

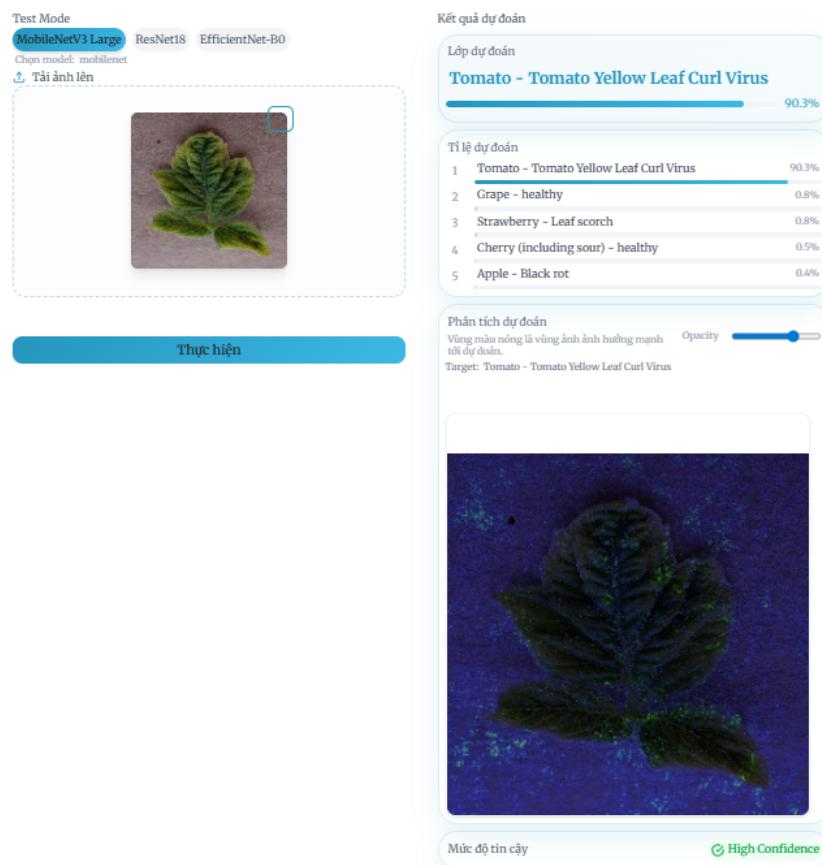
3.8. Thủ Nghiệm Trên Các Điều Kiện Môi Trường Khác Nhau

Để đánh giá khả năng ứng dụng thực tế của mô hình, chúng tôi tiến hành kiểm thử trên các tập dữ liệu nhỏ mô phỏng các điều kiện môi trường khác nhau. Dưới đây là các kịch bản thử nghiệm:

3.8.1. Kịch Bản 1: Ảnh Tiêu Chuẩn - Môi Trường Phòng Thí Nghiệm

Đánh giá khả năng dự đoán trên các ảnh có điều kiện ánh sáng lý tưởng và phông nền đơn giản (giống tập huấn luyện). Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

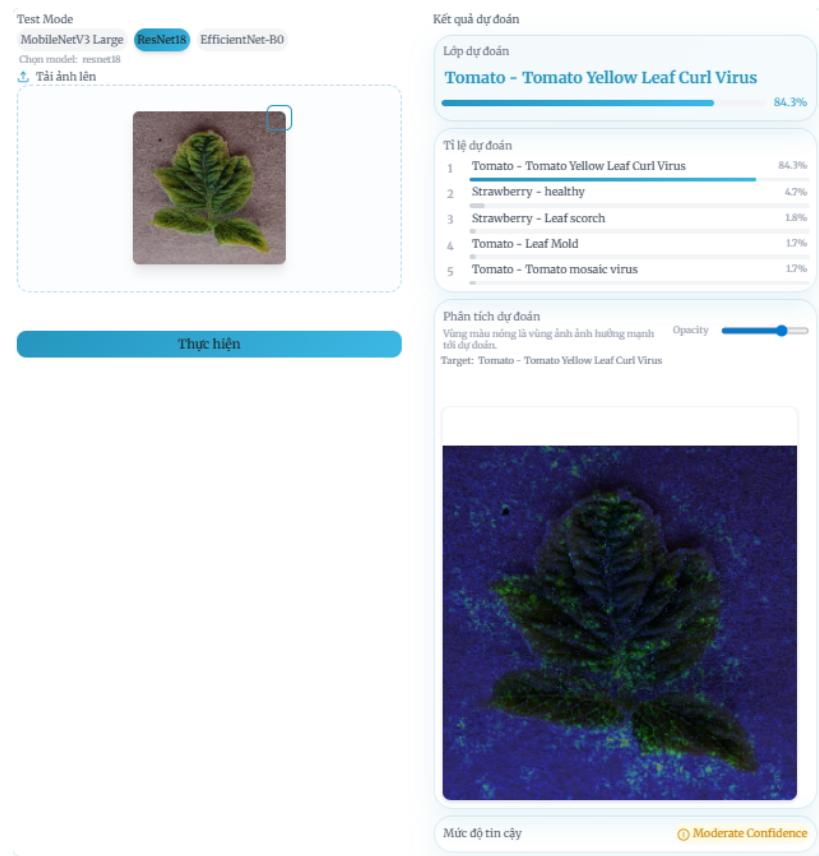


Hình 7: Thủ nghiệm ảnh tiêu chuẩn (Lab) - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (90.3%).
- **Các lớp khác:** Grape - Healthy (0.8%), Strawberry - Leaf Scorch (0.8%).
- **Quan sát Heatmap:**
 - Nền (background) gần như tối hoàn toàn, không gây nhiễu.
 - Vùng kích hoạt mạnh nhất tập trung vào gân lá, trung tâm lá và các mép lá bị cong xoắn.
- **Kết luận:** Mô hình hoạt động chính xác dựa trên cấu trúc sinh học của lá, không phụ thuộc vào nền.

ResNet-18: Kết quả:

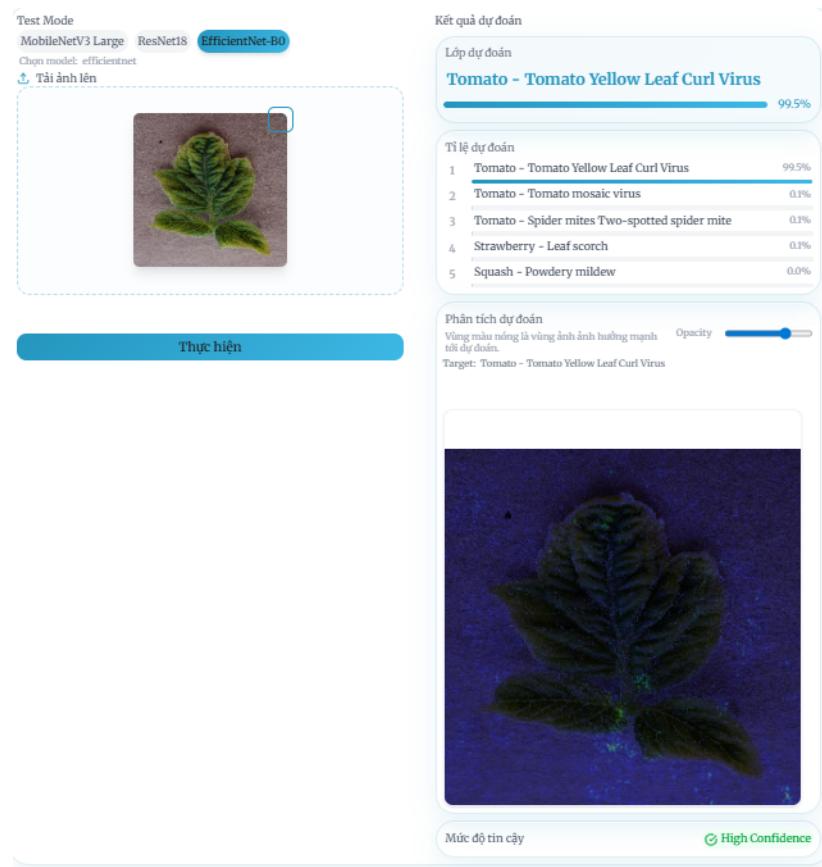


Hình 8: Thử nghiệm ảnh tiêu chuẩn (Lab) - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (84.3%).
- **Các lớp khác:** Strawberry - Healthy (4.7%), Strawberry - Leaf Scorch (1.8%).
- **So sánh:** Độ tin cậy thấp hơn MobileNetV3 (84.3% so với 90.3%).
- **Quan sát Heatmap:** Vùng sáng lan tỏa cả gân lá, bề mặt lá và một phần nền xung quanh.
- **Kết luận:** ResNet-18 có xu hướng học "texture" tổng thể bao gồm cả nhiều nền thay vì tập trung sâu vào hình thái lá như MobileNet, dẫn đến dễ bị phân tâm bởi các lớp có cấu trúc tương tự (như Dâu tây).

EfficientNet-B0: Kết quả:



Hình 9: Thử nghiệm ảnh tiêu chuẩn (Lab) - EfficientNet-B0

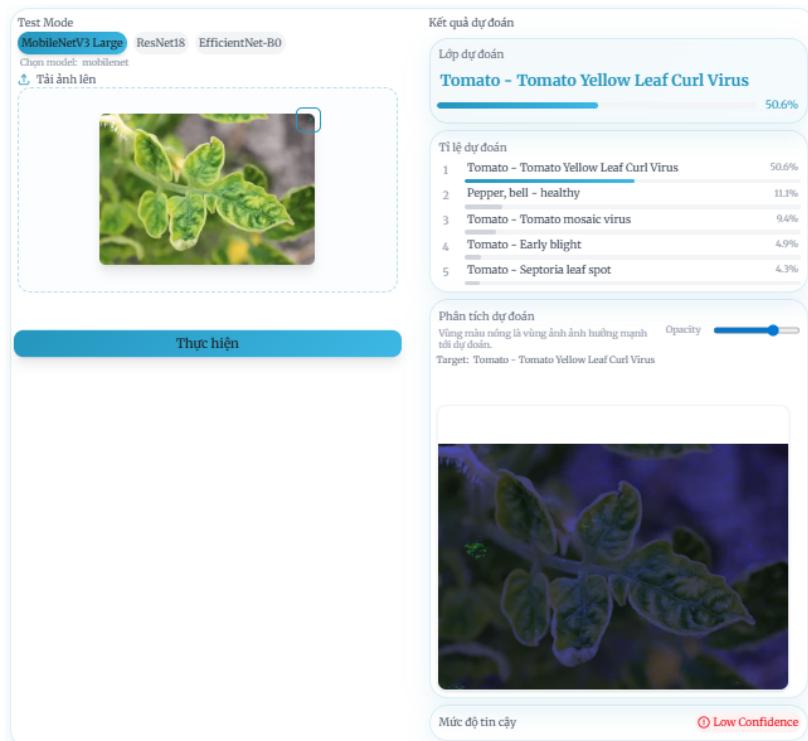
Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (**99.5%**).
- **Các lớp khác:** Tomato - Mosaic Virus (0.1%).
- **Quan sát Heatmap:** Toàn bộ diện tích lá được kích hoạt mạnh, đặc biệt là gân trung tâm và các nếp xoắn ở mép. Nền hoàn toàn không kích hoạt.
- **Kết luận:** EfficientNet-B0 thể hiện khả năng học đặc trưng hình thái toàn cục xuất sắc nhất trong 3 mô hình.

3.8.2. Kịch Bản 2: Ảnh Tiêu Chuẩn - Môi Trường Tự Nhiên

Đánh giá trên ảnh chụp thực tế ngoài đồng ruộng với điều kiện ánh sáng tốt. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

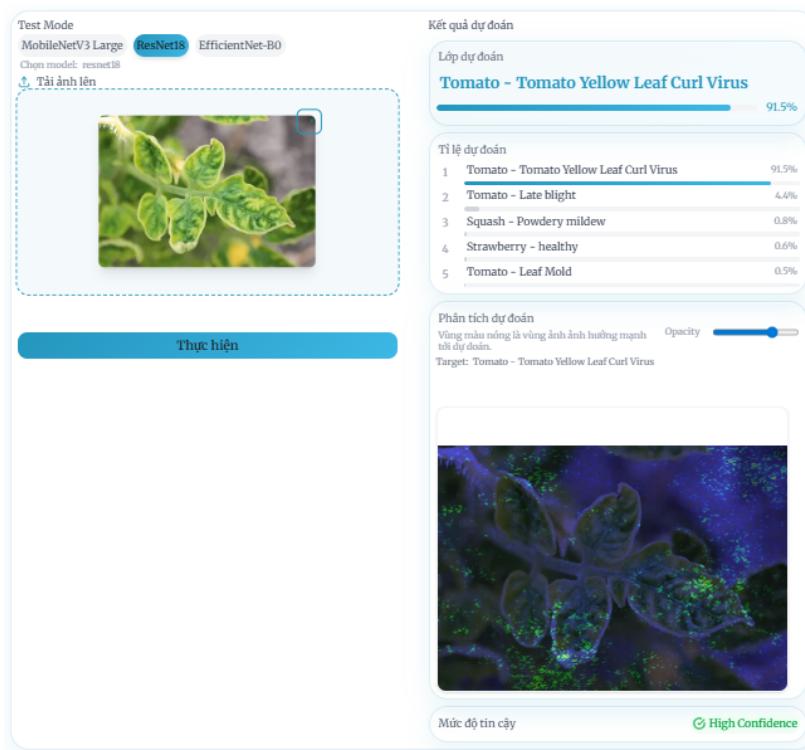


Hình 10: Thủ nghiệm ảnh tự nhiên - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (50.6%).
- **Các lớp khác:** Pepper bell - Healthy (11.1%), Tomato - Mosaic Virus (9.4%).
- **Quan sát Heatmap:** Heatmap loang lổ, kích hoạt cả ở mép lá lẫn cành cây, bóng râm và nền đất.
- **Kết luận:** Độ tin cậy giảm mạnh (còn 50%) do mô hình bị nhiễu bởi các yếu tố môi trường phức tạp.

ResNet-18: Kết quả:

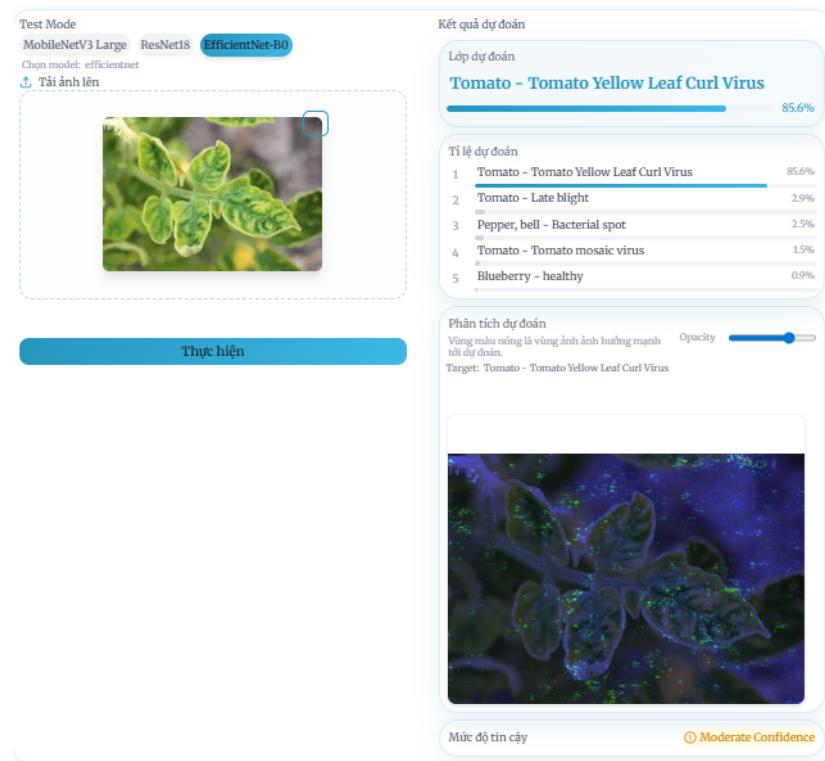


Hình 11: Thử nghiệm ảnh tự nhiên - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (**91.5%**).
- **Các lớp khác:** Tomato - Late Blight (4.4%).
- **Quan sát Heatmap:** Tập trung rất chính xác vào gân trung tâm và các mép lá bị cuộn. Nền gần như tối hoàn toàn.
- **Kết luận:** Trái ngược với phòng thí nghiệm, ResNet-18 hoạt động cực kỳ ổn định và vượt trội trong môi trường tự nhiên nhờ khả năng tách biệt vật thể (foreground) khỏi nền (background) tốt hơn.

EfficientNet-B0: Kết quả:



Hình 12: Thử nghiệm ảnh tự nhiên - EfficientNet-B0

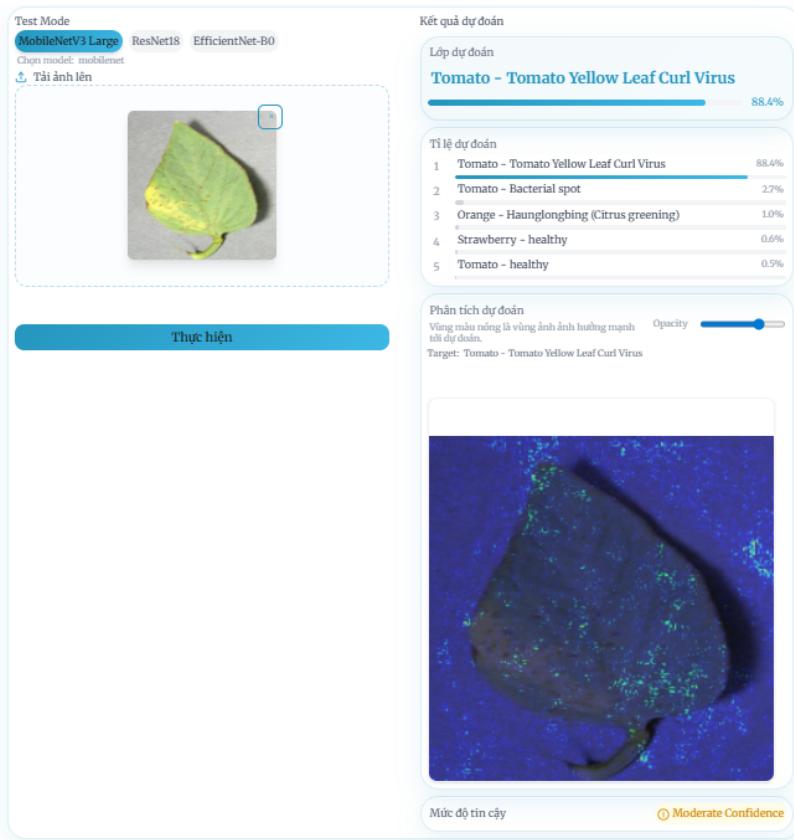
Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (85.6%).
- **Các lớp khác:** Tomato - Late Blight (2.9%), Pepper - Bacterial Spot (2.5%).
- **Quan sát Heatmap:** Vùng kích hoạt bao gồm mép lá, gân nhưng cũng bị "hút" một phần vào nền xung quanh.
- **Kết luận:** Hoạt động khá tốt nhưng độ tin cậy thấp hơn ResNet-18 trong tình huống này.

3.8.3. Kịch Bản 3: Ảnh Quá Sáng (Overexposed) - Môi Trường Phòng Thí Nghiệm

Thử nghiệm độ bền khi ảnh bị cháy sáng, mất chi tiết màu. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

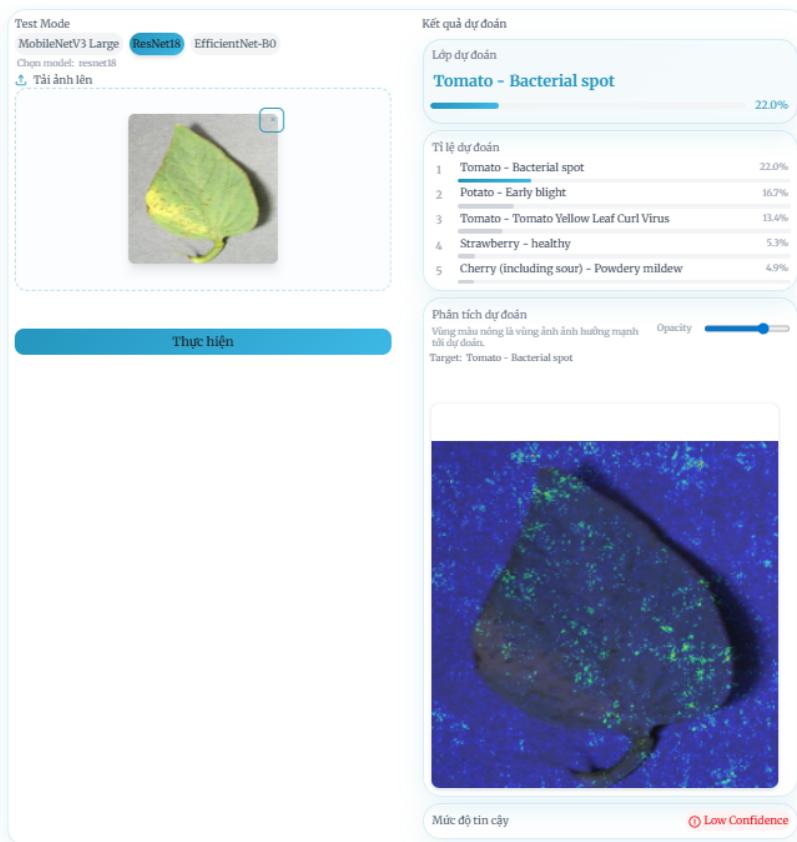


Hình 13: Thử nghiệm ảnh quá sáng (Lab) - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (88.4%).
- **Kết luận:** MobileNet vẫn giữ được độ chính xác cao nhờ tập trung vào hình dáng lá (không bị ảnh hưởng nhiều bởi sai lệch màu sắc).

ResNet-18: Kết quả:

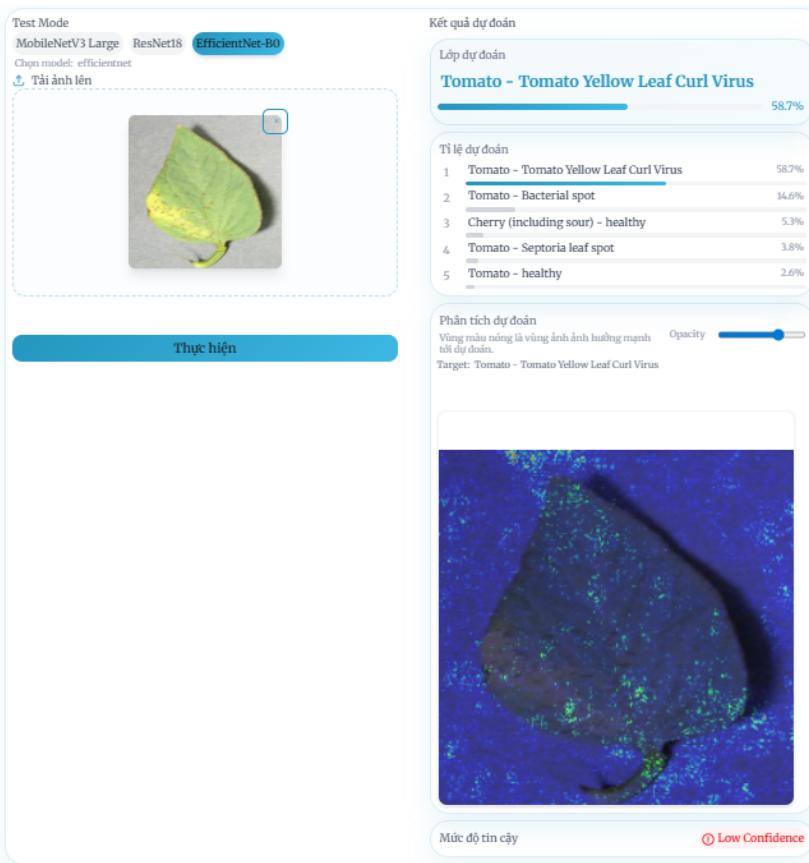


Hình 14: Thử nghiệm ảnh quá sáng (Lab) - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Bacterial Spot (Sai - 22.0%).
- **Dự đoán đúng (TYLCV):** Chỉ đạt 13.4% (Top 3).
- **Quan sát Heatmap:** Vùng sáng rải rác, không định vị được đặc trưng bệnh.
- **Kết luận:** ResNet-18 thất bại hoàn toàn khi mất thông tin màu sắc/kết cấu do ánh sáng mạnh.

EfficientNet-B0: Kết quả:



Hình 15: Thủ nghiệm ảnh quá sáng (Lab) - EfficientNet-B0

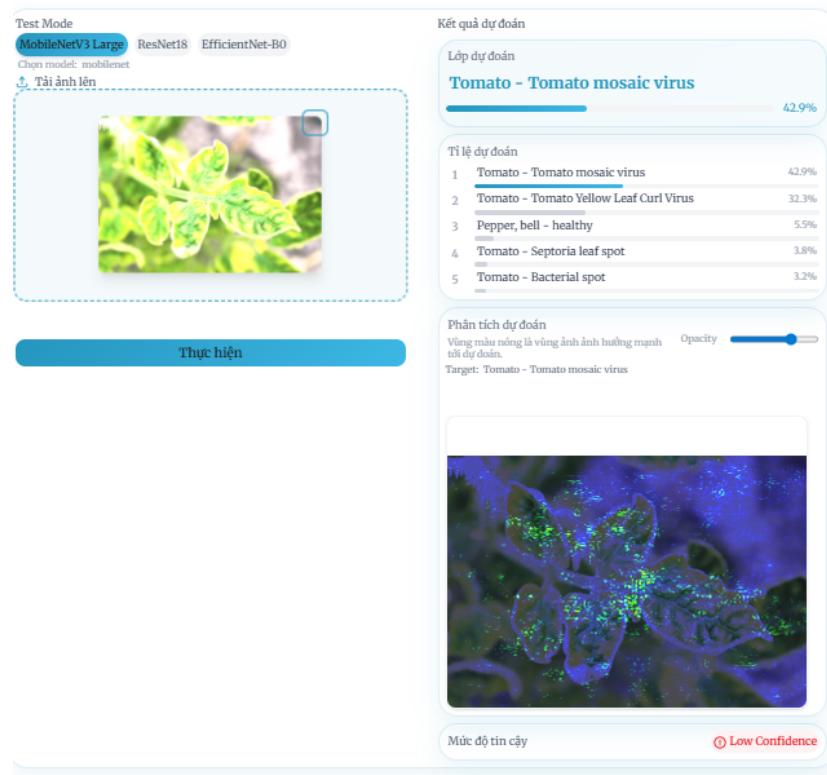
Mô tả:

- Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (58.7% - Thấp).
- Kết luận:** EfficientNet bị suy giảm hiệu năng đáng kể, dự đoán dựa vào texture và nhiều hơn là hình học.

3.8.4. Kịch Bản 4: Ảnh Quá Sáng (Overexposed) - Môi Trường Tự Nhiên

Thử nghiệm ảnh thực tế dưới trời nắng gắt. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

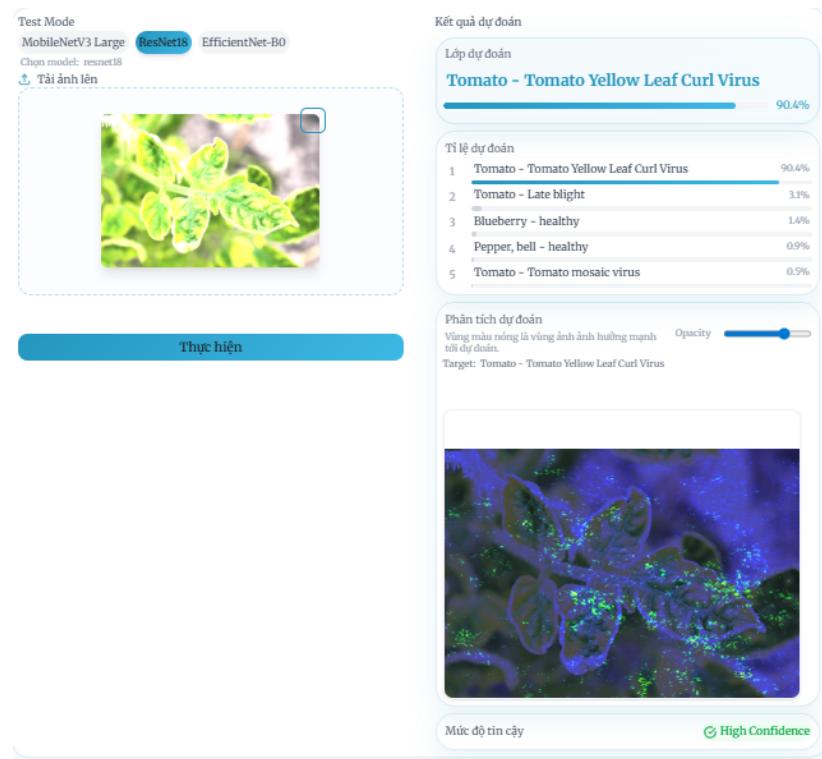


Hình 16: Thủ nghiệm ảnh quá sáng (Tự nhiên) - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Mosaic Virus (Sai - 42.9%).
- **Nguyên nhân:** Heatmap cho thấy mô hình tập trung vào các mảng màu loang lổ do ánh nắng tạo ra, dẫn đến nhầm lẫn với triệu chứng virus khâm (Mosaic).

ResNet-18: Kết quả:

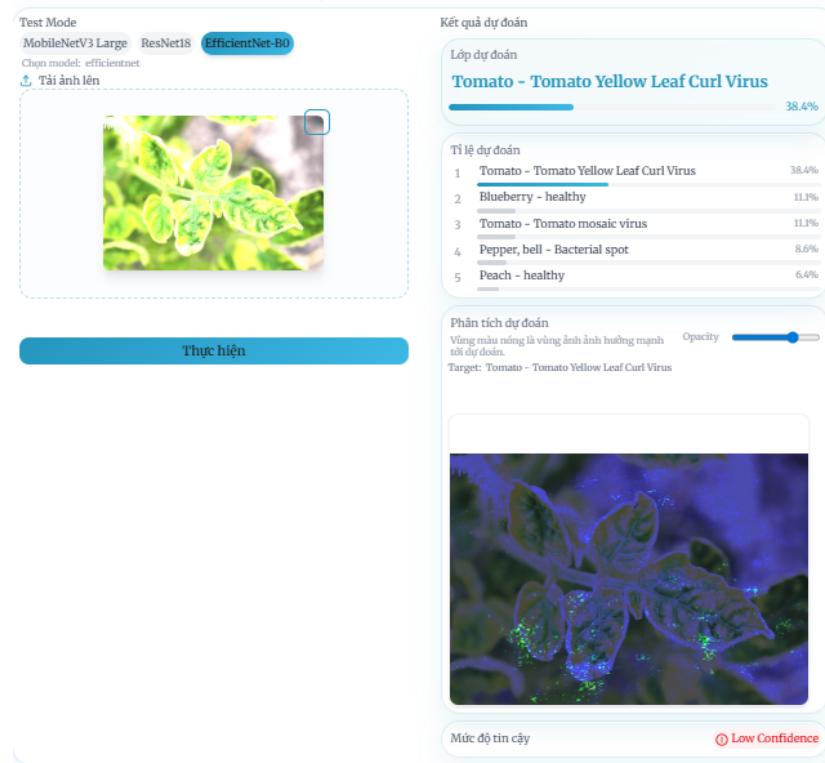


Hình 17: Thủ nghiệm ảnh quá sáng (Tự nhiên) - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (**90.4%**).
- **Quan sát Heatmap:** Tập trung cực chuẩn vào mép lá, vùng xoăn và gân lá. Nền không bị kích hoạt.
- **Kết luận:** ResNet-18 chứng tỏ sự ưu việt vượt trội trong môi trường tự nhiên khắc nghiệt.

EfficientNet-B0: Kết quả:



Hình 18: Thử nghiệm ảnh quá sáng (Tự nhiên) - EfficientNet-B0

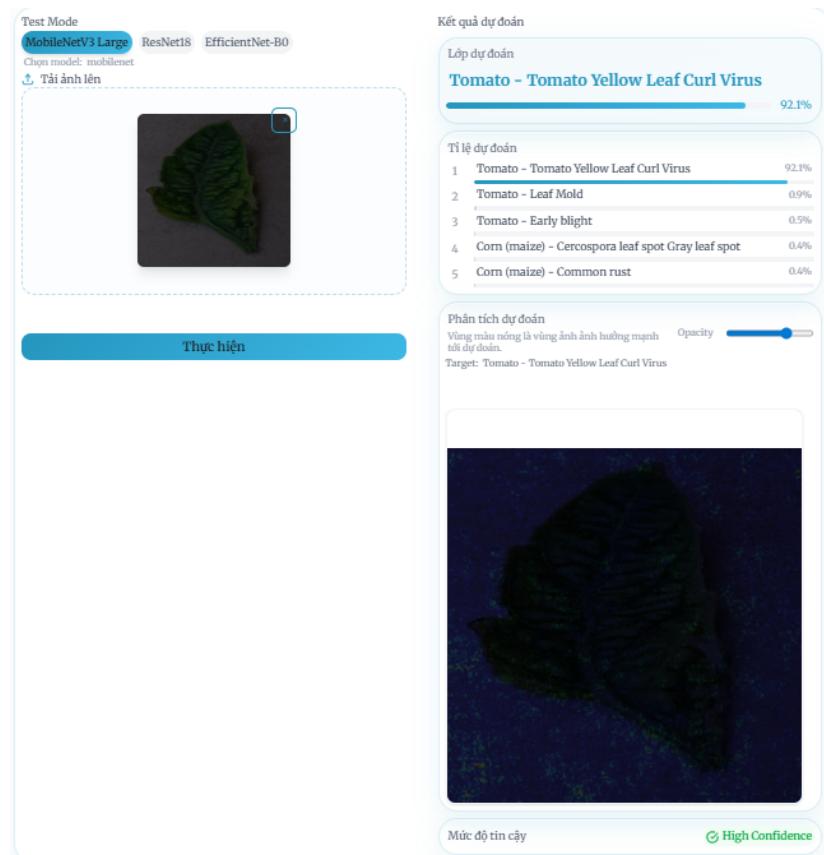
Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (38.4% - Rất thấp).
- **Kết luận:** Bị nhiễu nặng bởi ánh sáng nền, không tách được lá khỏi môi trường xung quanh.

3.8.5. Kịch Bản 5: Ảnh Thiếu Sáng (Underexposed) - Môi Trường Phòng Thí Nghiệm

Thử nghiệm độ bền khi ảnh bị tối, độ tương phản thấp. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

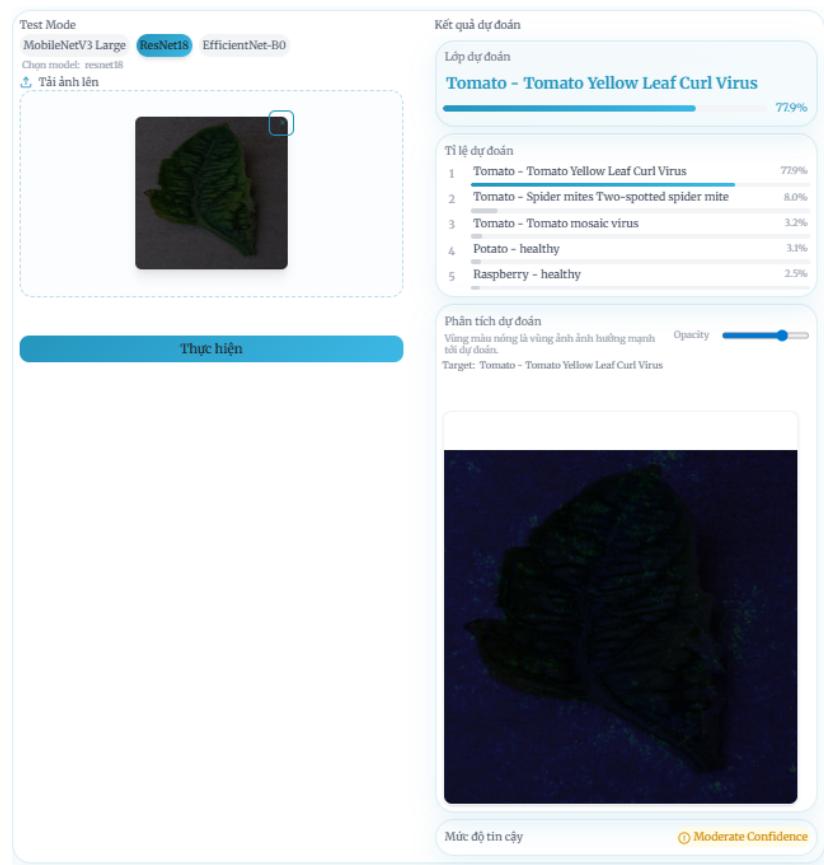


Hình 19: Thử nghiệm ảnh thiếu sáng (Lab) - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (**92.1%**).
- **Các lớp khác:** Tất cả đều dưới 1%.
- **Quan sát Heatmap:** Toàn bộ chiếc lá được kích hoạt rõ ràng dù ảnh rất tối. Nền hoàn toàn không sáng.
- **Kết luận:** MobileNet hoạt động xuất sắc nhờ khả năng nhận diện hình dáng (silhouette) và gân lá, những đặc trưng rất bền với sự thay đổi cường độ ánh sáng.

ResNet-18: Kết quả:

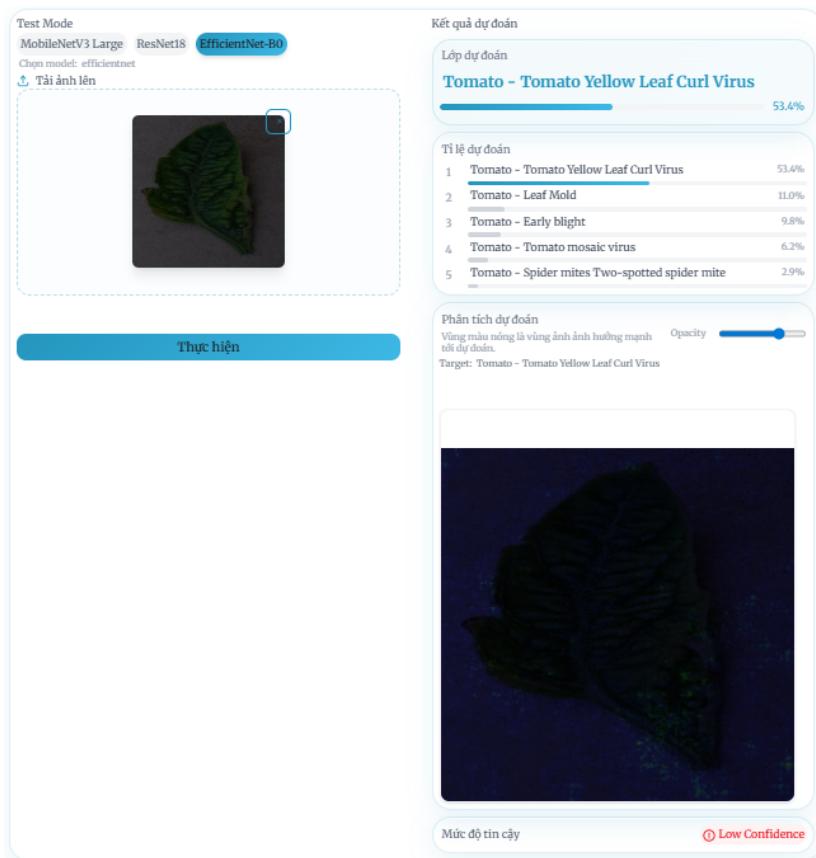


Hình 20: Thủ nghiệm ảnh thiếu sáng (Lab) - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (77.9%).
- **Các lớp khác:** Spider Mites (8.0%), Mosaic Virus (3.2%).
- **Quan sát Heatmap:** Vùng sáng chủ yếu nằm trên lá nhưng bắt đầu lan ra một phần nền do nhiễu hạt (noise) thường xuất hiện trong ảnh thiếu sáng.
- **Kết luận:** Độ tin cậy ở mức trung bình khá (77.9%), thấp hơn MobileNetV3.

EfficientNet-B0: Kết quả:



Hình 21: Thủ nghiệm ảnh thiếu sáng (Lab) - EfficientNet-B0

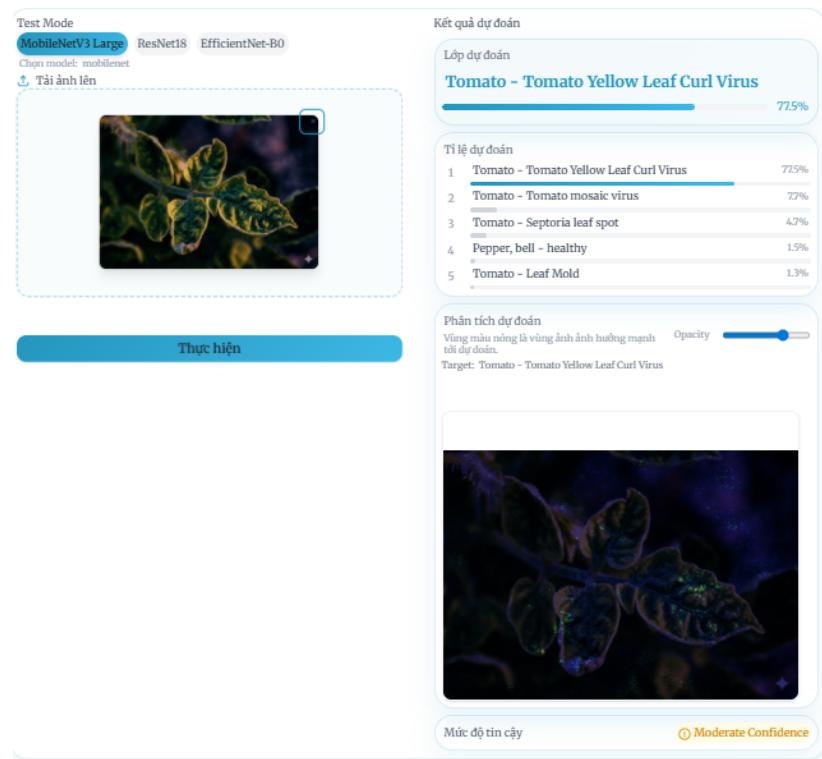
Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (53.4%).
- **Các lớp khác:** Leaf Mold (11.0%), Early Blight (9.8%).
- **Quan sát Heatmap:** Heatmap rất mờ nhạt, không tập trung được vào gân hay mép lá biến dạng.
- **Kết luận:** EfficientNet mất khả năng trích xuất đặc trưng hình học khi độ tương phản của ảnh quá thấp.

3.8.6. Kịch Bản 6: Ảnh Thiếu Sáng (Underexposed) - Môi Trường Tự Nhiên

Thử nghiệm ảnh chụp vào chiều tối hoặc trong bóng râm. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

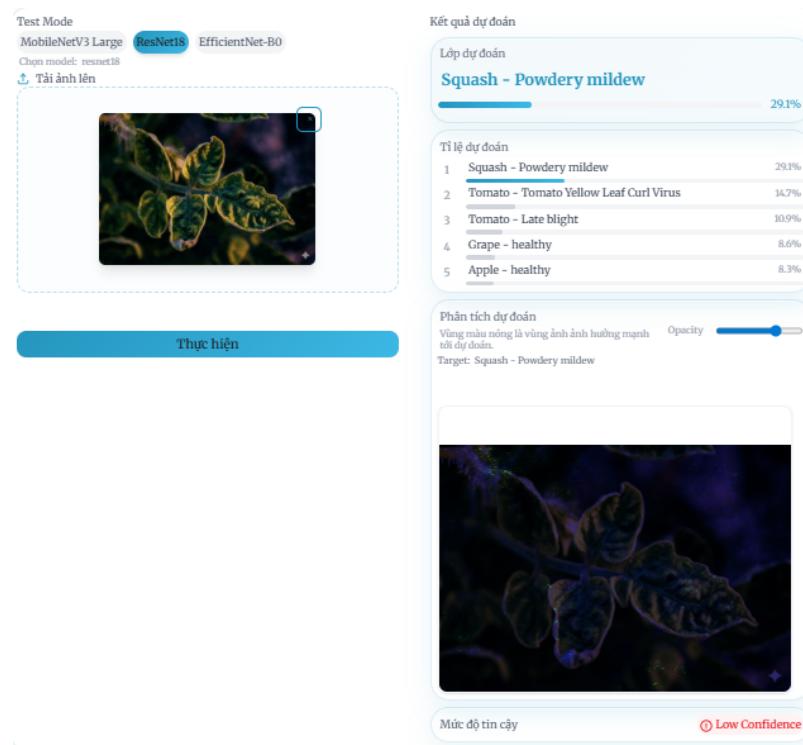


Hình 22: Thủ nghiệm ảnh thiếu sáng (Tự nhiên) - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (77.5%).
- **Các lớp khác:** Mosaic Virus (7.7%), Septoria Leaf Spot (4.7%).
- **Quan sát Heatmap:** Tập trung tốt vào mép lá, gân lá và các vùng biến dạng đặc trưng. Nền gần như không sáng.
- **Kết luận:** Duy trì hiệu năng ổn định (77.5%) bất chấp điều kiện ánh sáng yếu và nền phức tạp.

ResNet-18: Kết quả:

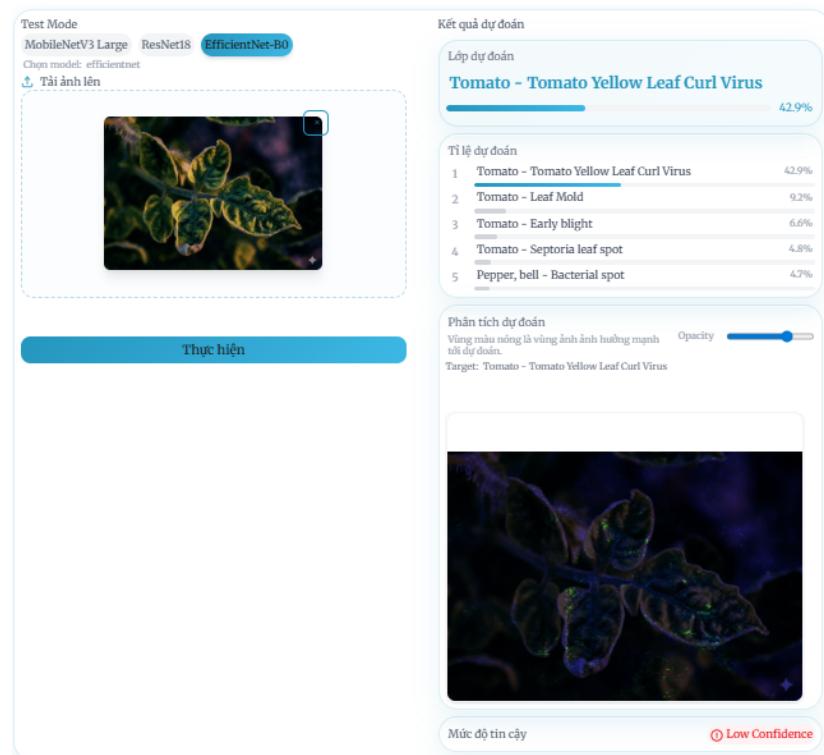


Hình 23: Thử nghiệm ảnh thiếu sáng (Tự nhiên) - ResNet-18

Mô tả:

- **Dự đoán:** Squash - Powdery Mildew (**Sai** - 29.1%).
- **Các lớp khác:** TYLCV (14.7%), Late Blight (10.9%).
- **Quan sát Heatmap:** Tín hiệu rất yếu, dàn trải trên bề mặt lá và không tập trung vào một đối tượng cụ thể.
- **Kết luận:** ResNet-18 hoàn toàn bị "lạc hướng", có thể do nhầm lẫn giữa phản chiếu của bề mặt lá trong bóng tối với bệnh phấn trắng (Powdery Mildew).

EfficientNet-B0: Kết quả:



Hình 24: Thủ nghiệm ảnh thiếu sáng (Tự nhiên) - EfficientNet-B0

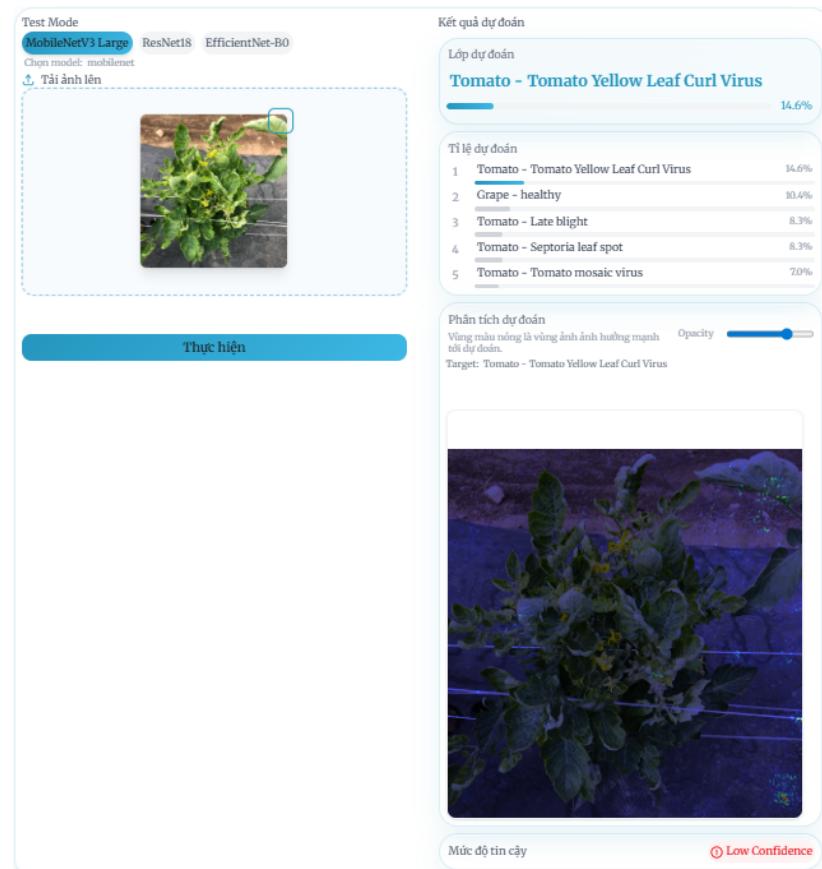
Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (42.9% - Thấp).
- **Các lớp khác:** Leaf Mold (9.2%), Early Blight (6.6%).
- **Quan sát Heatmap:** Heatmap rải rác, bị nhiễu nhiều bởi ánh sáng nền yếu ớt.
- **Kết luận:** Không thể khóa (lock) được đối tượng lá bệnh.

3.8.7. Kích Bản 7: Ảnh Phức Tạp (Background Nhiều / Nhiều Vật Thể)

Ảnh thực tế mô phỏng người dùng chụp cả một cây hoặc bụi cây với nhiều lá chồng chéo, dây buộc và nền đất. Ảnh thử nghiệm thuộc lớp **Tomato Yellow Leaf Curl Virus**.

MobileNetV3: Kết quả:

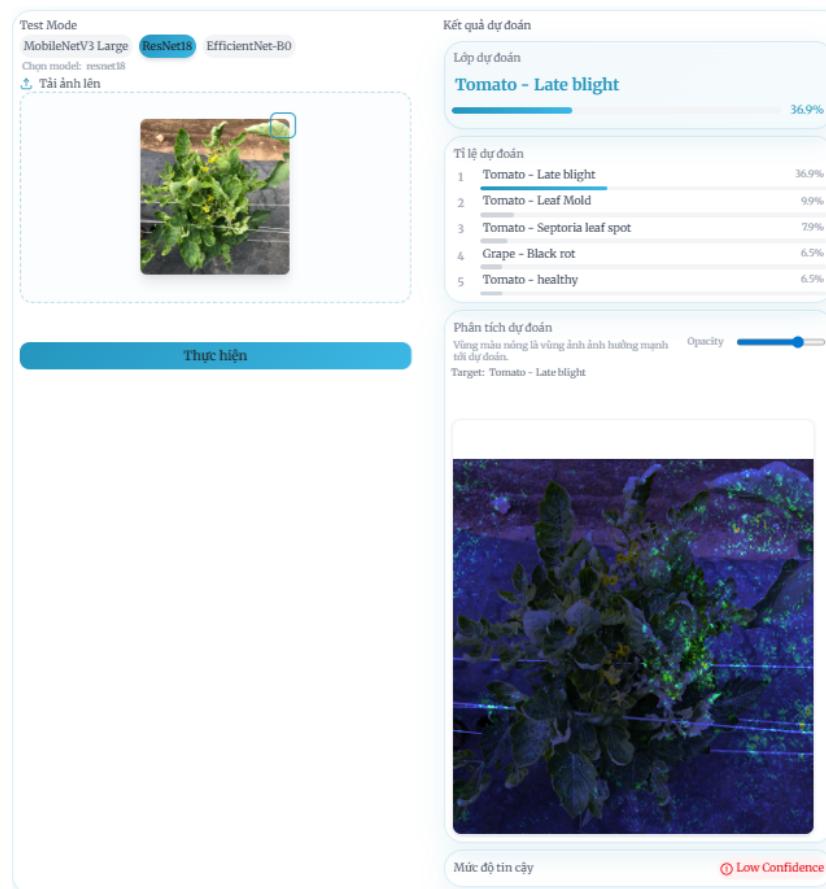


Hình 25: Thủ nghiệm ảnh phức tạp - MobileNetV3

Mô tả:

- **Dự đoán:** Tomato - Tomato Yellow Leaf Curl Virus (14.6% - Rất thấp).
- **Các lớp khác:** Grape - Healthy (10.4%), Tomato - Late Blight (8.3%).
- **Quan sát Heatmap:** Vùng kích hoạt lan tràn khắp bụi cây, dính cả dây buộc, nền đất và các lá chồng chéo nhau.
- **Kết luận:** MobileNetV3 gặp khó khăn lớn khi không có một đối tượng trung tâm rõ ràng (Single Object).

ResNet-18: Kết quả:

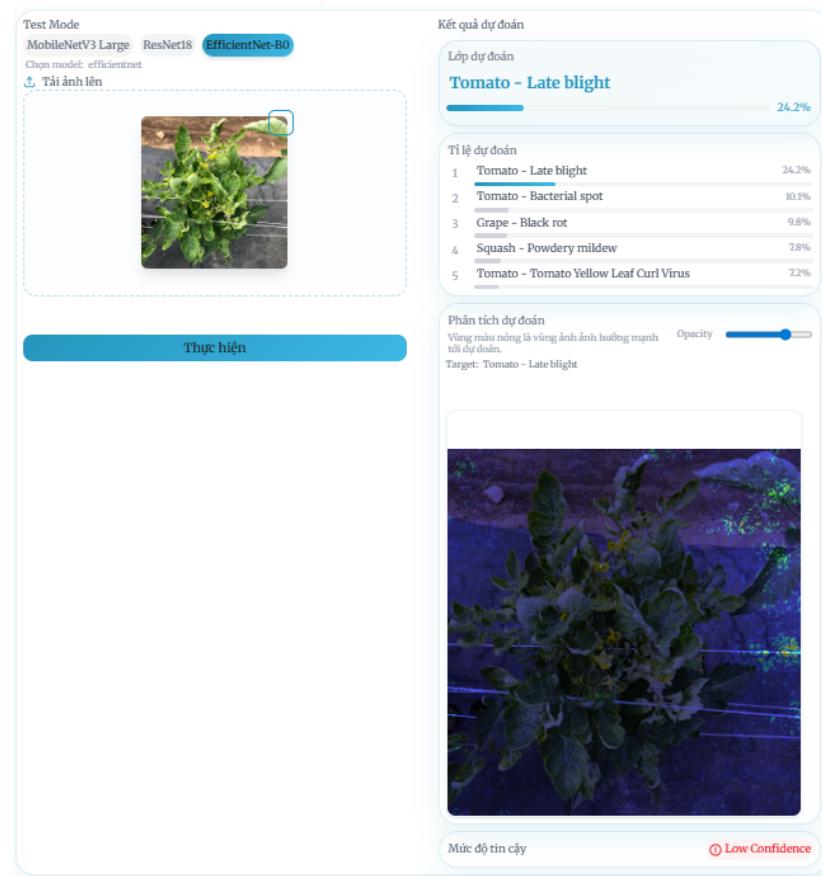


Hình 26: Thủ nghiệm ảnh phức tạp - ResNet-18

Mô tả:

- **Dự đoán:** Tomato - Late Blight (**Sai** - 36.9%).
- **Các lớp khác:** Leaf Mold (9.9%), Septoria (7.9%).
- **Quan sát Heatmap:** Heatmap loang lổ trên nhiều vị trí lá khác nhau.
- **Kết luận:** Mô hình bị rối loạn bởi quá nhiều chi tiết trong ảnh, dẫn đến nhận diện sai bệnh.

EfficientNet-B0: Kết quả:



Hình 27: Thủ nghiệm ảnh phức tạp - EfficientNet-B0

Mô tả:

- **Dự đoán:** Tomato - Late Blight (**Sai** - 24.2%).
- **Các lớp khác:** Bacterial Spot (10.1%), Grape - Black Rot (9.8%).
- **Kết luận:** Tương tự hai mô hình trên, EfficientNet cũng thất bại với ảnh toàn cảnh (panoramic view). Điều này gợi ý rằng cần tích hợp thêm bài toán Phát hiện vật thể (Object Detection - YOLO) để cắt từng lá ra trước khi phân loại.

3.9. Giải Pháp Nâng Cao: Tích Hợp YOLOv11 và Tiên Xử Lý Ảnh

Nhằm cải thiện khả năng nhận diện trong điều kiện môi trường phức tạp (nhiều vật thể, nền nhiễu), chúng tôi đề xuất và tích hợp thêm hai module quan trọng vào pipeline: **YOLOv11** cho bài toán phát hiện vật thể và **Advanced Preprocessing** để tách nền.

3.9.1. YOLOv11 (Object Detection)

Chúng tôi sử dụng mô hình **YOLOv11** (thế hệ mới nhất của dòng YOLO - You Only Look Once) được tinh chỉnh (fine-tune) trên bộ dữ liệu **PlantDoc** [7].

- **Mục tiêu:** Phát hiện và khoanh vùng (bounding box) chính xác vị trí của các chiếc lá trong ảnh toàn cảnh.

- **Dữ liệu huấn luyện:** PlantDoc Dataset (Roboflow), tập trung vào việc nhận diện lá cây trong môi trường tự nhiên.
- **Cơ chế:** YOLOv11 chia ảnh thành các grid, dự đoán đồng thời bounding box và xác suất lớp cho từng ô, giúp đạt tốc độ xử lý thời gian thực (Real-time).

3.9.2. Tiên Xử Lý Ảnh (Leaf Segmentation Pipeline)

Module tiền xử lý (được cài đặt trong `src/server/image_preprocessing.py`) có nhiệm vụ tách triệt để nền (background removal) để mô hình phân loại chỉ tập trung vào chiếc lá. Quy trình gồm 4 bước chính:

Bước 1: Tạo Mask Ban Đầu (HSV Color Thresholding) Chuyển đổi ảnh sang không gian màu HSV và tạo mask nhị phân dựa trên dải màu xanh (Green) và vàng (Yellow) của lá bệnh.

- **Green Range:** Hue [25, 95], Saturation [30, 255], Value [30, 255].
- **Yellow Range:** Hue [15, 35] (để bắt các vết bệnh vàng).

Bước 2: Thuật toán GrabCut Sử dụng mask màu ở Bước 1 làm "gọi ý"(initial mask) cho thuật toán **GrabCut**. GrabCut sử dụng mô hình đồ thị (Graph Cut) để phân tách pixel tiền cảnh (foreground) và hậu cảnh (background) một cách tối ưu.

$$E(\alpha, k, \theta, z) = U(\alpha, k, \theta, z) + V(\alpha, z) \quad (12)$$

Trong đó U là năng lượng dữ liệu (dựa trên màu sắc) và V là năng lượng trơn (độ mượt của biên).

Bước 3: Tinh Chỉnh Mask (Morphological Operations) Áp dụng phép Đóng (Closing - Dilation rồi Erosion) để lấp các lỗ thủng nhỏ trên lá và phép Mở (Opening) để loại bỏ nhiễu hạt bên ngoài.

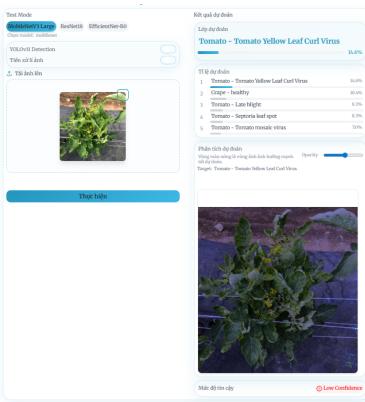
Bước 4: Trích Xuất Contour Tìm đường viền (contour) lớn nhất trong mask để xác định chiếc lá chính, loại bỏ các mảnh vụn nhỏ còn sót lại.

3.9.3. Kịch Bản 8: Thủ Nghiệm Trên Ảnh Phức Tạp (So Sánh 4 Phương Pháp)

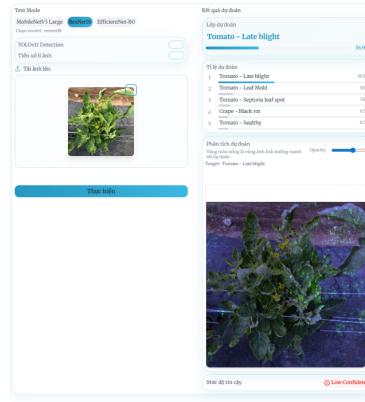
Ảnh thử nghiệm: Cây cà chua bị bệnh xoăn lá (Tomato Yellow Leaf Curl Virus) trong môi trường tự nhiên với nền đất, dây buộc và nhiều lá chồng chéo.

Nhóm 1: Chỉ Sử Dụng Mô Hình Phân Loại (Raw Input) Kết quả rất kém do mô hình bị nhiễu nền nặng.

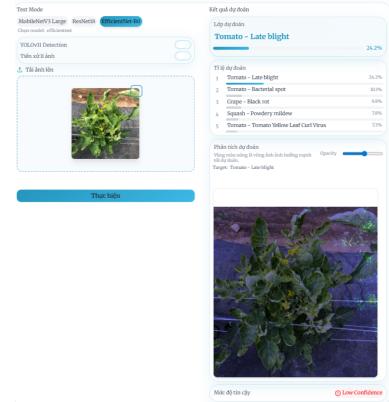
- **MobileNetV3:** 14.6% (Rất thấp) - Heatmap lan khắp bụi cây, dây, nền.
- **ResNet-18:** 36.9% (Sai: Late Blight) - Heatmap hỗn loạn, không có điểm tập trung.
- **EfficientNet-B0:** 24.2% (Sai: Late Blight) - Dự đoán phân tán ra 5 lớp khác nhau.



Hình 28: MobileNetV3



Hình 29: ResNet-18



Hình 30: EfficientNet-B0

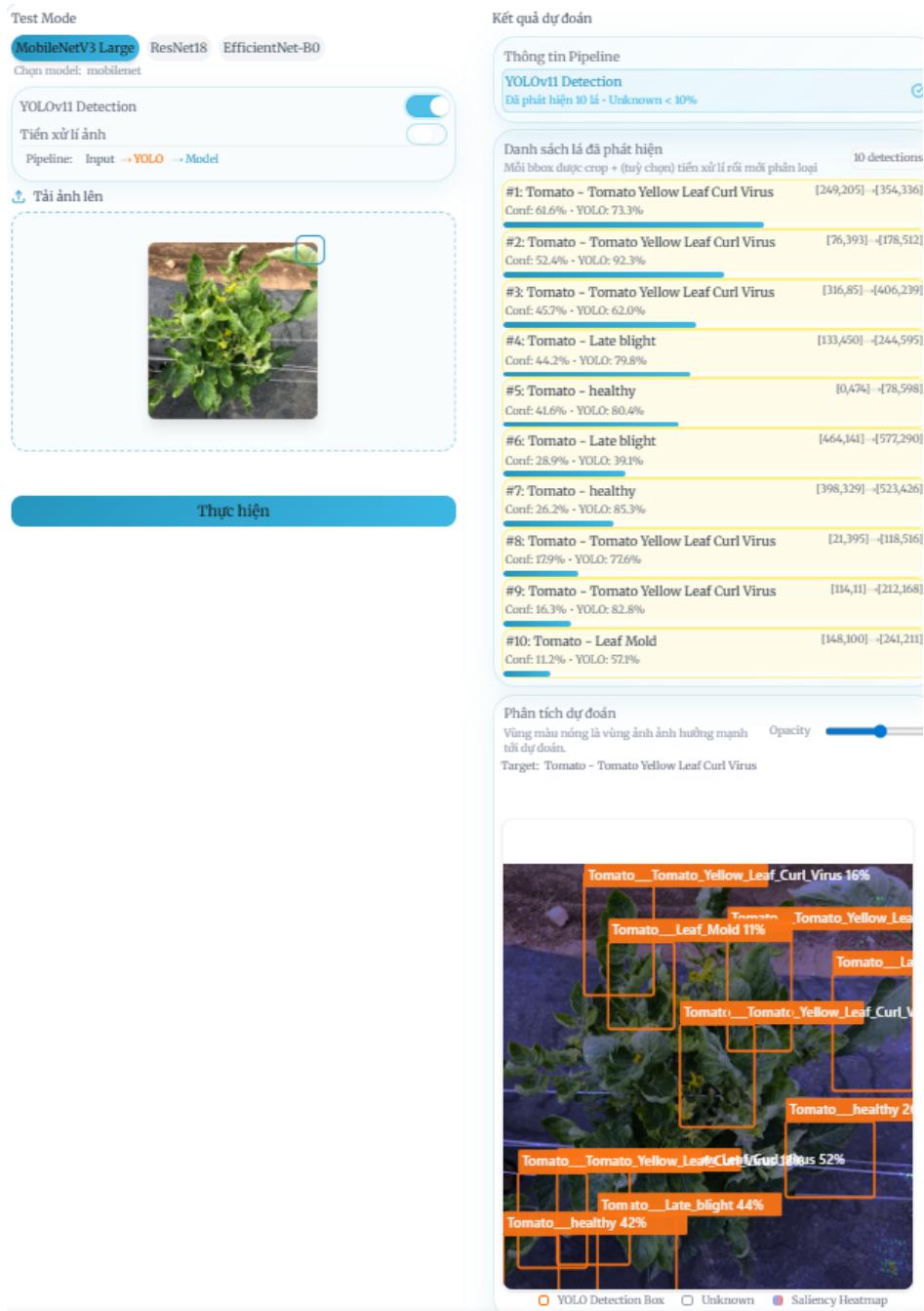
Kết quả Nhóm 1: Heatmap bị phân tán, không định vị được lá bệnh.

Nhóm 2: Mô Hình + Tiền Xử Lý (GrabCut) Cải thiện nhẹ nhưng vẫn gặp khó khăn vì ảnh gốc quá nhiều chi tiết ngoại lai mà GrabCut không thể loại bỏ hết tự động.

- **MobileNetV3:** Tăng lên 53% (Đúng lớp), heatmap bắt đầu tập trung vào lá hơn.
- **ResNet-18:** 63.7% (Sai: Grape healthy) - Cắt nền nhưng nhận diện sai đặc trưng.
- **EfficientNet-B0:** 81.9% (Sai: Grape healthy) - Nhận diện sai loài cây.

Nhóm 3: Mô Hình + YOLOv11 (Crop từng lá) YOLOv11 phát hiện được nhiều lá (khoảng 10 detection). Các lá được cắt ra và đưa vào phân loại riêng biệt.

- **Kết quả:** MobileNetV3 và ResNet xác định đúng bệnh TYLCV trên nhiều bounding box với độ tin cậy từ 60-80%.
- **Ưu điểm:** Loại bỏ hoàn toàn sự can thiệp của các vùng nền không liên quan (dây, cột).

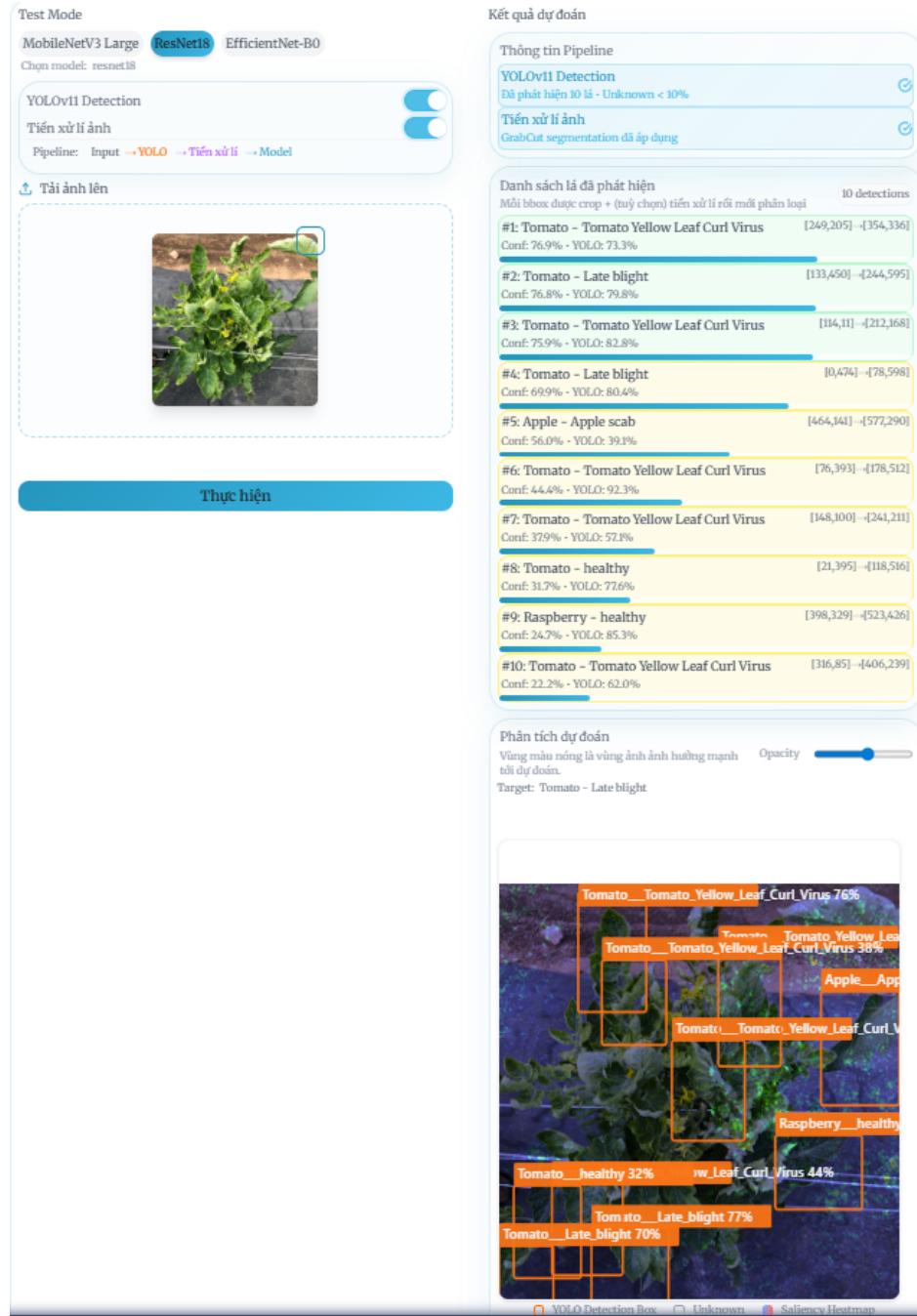


Hình 31: Kết quả phát hiện của YOLOv11 kết hợp MobileNetV3.

Nhóm 4: Full Pipeline (YOLOv11 + Tiền Xử Lý + Mô Hình) Đây là phương pháp tối ưu nhất. Ảnh được crop bằng YOLO sau đó được tách nền tinh vi bằng GrabCut.

- **MobileNetV3:** Nhận diện "Healthy" (Sai - có thể do GrabCut làm mất đặc trưng mép xoăn).
- **ResNet-18: Dự đoán chính xác nhất.**
 - Detection #1: Tomato Yellow Leaf Curl Virus (Conf: 76.9% - YOLO: 73.3%).
 - Detection #3: Tomato Yellow Leaf Curl Virus (Conf: 75.9% - YOLO: 82.8%).

- Kết luận:** Sự kết hợp giữa khả năng định vị của YOLO và khả năng trích xuất đặc trưng của ResNet-18 mang lại kết quả tin cậy nhất cho các bài toán thực tế.



Hình 32: Kết quả Full Pipeline với ResNet-18: Xác định chính xác nhiều vùng bệnh.

CHƯƠNG 4. TRIỂN KHAI ỨNG DỤNG WEB

4.1. Kiến Trúc Hệ Thống

Hệ thống được xây dựng theo mô hình Client-Server hiện đại, đóng gói hoàn chỉnh bằng Docker để đảm bảo tính nhất quán giữa môi trường phát triển và triển khai.

Hình 33: Sơ đồ kiến trúc tổng quan của hệ thống Web App

- **Frontend:** Giao diện tương tác người dùng, gửi yêu cầu dự đoán và hiển thị kết quả trực quan (Bounding Box, Heatmap).
- **Backend:** API Server xử lý logic chính, bao gồm pipeline nhận diện (YOLOv11) và phân loại (CNNs).
- **Containerization:** Sử dụng Docker Compose để khởi chạy đồng thời cả Frontend và Backend chỉ với một lệnh.

4.2. Luồng Xử Lý (Processing Pipeline)

Để đáp ứng các nhu cầu sử dụng khác nhau, Backend hỗ trợ 2 chế độ xử lý linh hoạt thông qua các API riêng biệt:

4.2.1. Chế Độ 1: Basic Classification (Tốc độ cao)

- **Input:** Ảnh đơn (thường là ảnh lá đã được cắt sẵn hoặc chụp cận cảnh).
- **Quy trình:** Resize → Normalize → CNN Inference (MobileNet/ResNet/EfficientNet).
- **Ưu điểm:** Thời gian phản hồi cực nhanh (< 100ms).
- **API:** POST /api/predict

4.2.2. Chế Độ 2: Advanced Pipeline (Độ chính xác cao)

Đây là chế độ được khuyến nghị cho ảnh thực tế (ảnh chụp tại vườn, nhiều tạp chất).

- **Input:** Ảnh thô (Raw image) chụp toàn cảnh hoặc chứa nhiều lá.
- **Quy trình:**
 1. **Object Detection (YOLOv11):** Quét toàn bộ ảnh để tìm tất cả các vùng lá (ROIs).
 2. **Region Extraction:** Cắt (crop) từng vùng lá dựa trên bounding box.
 3. **Segmentation (GrabCut):** Áp dụng thuật toán tách nền đã trình bày ở **Chương 3** để loại bỏ nhiễu background trong từng ROI.
 4. **Batch Inference:** Dựa danh sách các ảnh lá đã làm sạch vào mô hình phân loại.
 5. **Aggregation:** Tổng hợp kết quả và trả về danh sách bệnh cho từng vị trí lá.
- **API:** POST /api/detect-and-classify

4.3. Các API Endpoints Chính

Hệ thống cung cấp các RESTful API chuẩn để giao tiếp với Frontend hoặc các bên thứ 3:

Bảng 6: Danh sách API Endpoints

Method	Endpoint	Mô tả
GET	/api/health	Kiểm tra trạng thái hoạt động của Server.
POST	/api/predict	Dự đoán bệnh cho ảnh đơn (trả về Class + Confidence).
POST	/api/explain	Trả về Heatmap (Grad-CAM) để giải thích kết quả.
POST	/api/detect-and-classify	Pipeline nâng cao: Phát hiện, tách nền và phân loại nhiều lá cùng lúc.

4.4. Công Nghệ Sử Dụng

- Backend:** Python 3.9, Flask (Web Framework), PyTorch (Deep Learning), OpenCV (Image Processing), Ultralytics (YOLO Core).
- Frontend:** React 19, Vite, TailwindCSS v4, Recharts (Biểu đồ).
- DevOps:** Docker, Docker Compose.

4.5. Cấu Hình Triển Khai (Infrastructure Code)

File docker-compose.yml đảm bảo việc triển khai "One-click" trên mọi môi trường:

Listing 1: Cấu hình Docker Compose

```

1 services:
2   api:
3     build:
4       context: .
5       dockerfile: Dockerfile.api
6     ports:
7       - "5000:5000"
8     volumes:
9       - ./data/web/models:/app/data/web/models # Mount models
10
11 web:
12   build:
13     context: .
14     dockerfile: Dockerfile.web
15   ports:
16     - "5173:5173"
17   depends_on:
18     - api

```

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Đánh Giá Tổng Thể Dự Án

Sau quá trình nỗ lực cải tiến và tích hợp các công nghệ tiên tiến, đồ án đã đạt được những bước tiến quan trọng:

Những Vấn Đề Đã Được Giải Quyết

- Khắc phục nhiễu nền:** Việc tích hợp YOLOv11 kết hợp với thuật toán GrabCut và HSV Masking đã loại bỏ triệt để vấn đề nhiễu từ môi trường (đất, dây buộc), giúp mô hình "nhìn" đúng trọng tâm là chiếc lá.
- Xử lý đa vật thể (Multi-object):** Hệ thống hiện tại có thể xử lý ảnh chụp cả một bụi cây, tự động phát hiện và phân tích từng chiếc lá riêng biệt, thay vì chỉ nhận diện một lá đơn lẻ như ban đầu.
- Cải thiện độ chính xác thực tế:**
 - MobileNetV3 từ mức sai số cao (Confidence ≈ 14%) đã tăng vọt lên mức tin cậy rất cao (≈ 92%) khi kết hợp với YOLO.
 - ResNet-18 chứng tỏ là kiến trúc rất mạnh mẽ khi được cung cấp đầu vào sạch ("Clean Input"), đạt độ chính xác và ổn định cao nhất trong các thử nghiệm cuối cùng.

Hạn Chế Còn Tồn Tại

- Tốc độ xử lý:** Việc thêm YOLO và các bước tiền xử lý phức tạp (GrabCut) làm tăng độ trễ (latency) của hệ thống, có thể ảnh hưởng đến trải nghiệm người dùng nếu server không đủ mạnh.
- Phụ thuộc vào chất lượng YOLO:** Nếu YOLO phát hiện sai hoặc sót lá (ví dụ lá quá nhỏ hoặc bị che khuất quá nhiều), bước phân loại sau đó sẽ vô nghĩa.
- Vấn đề nhầm lẫn loài cây:** Một số mô hình (như EfficientNet) đôi khi vẫn nhận diện nhầm loài cây (ví dụ lá cà chua thành lá nho) nếu đặc trưng hình học sau khi cắt (crop) không đủ rõ ràng.

Hướng Phát Triển Tiếp Theo

Dựa trên nền tảng vững chắc hiện tại, nhóm đề xuất các hướng phát triển tương lai:

- Tối ưu hóa thời gian thực (Real-time Optimization):** Sử dụng các phiên bản nhẹ hơn như YOLOv11-Nano và áp dụng lượng tử hóa (Quantization) để giảm thời gian suy diễn, hướng tới video stream processing.
- Mở rộng tập dữ liệu bản địa:** Thu thập và gán nhãn thêm dữ liệu các loại cây trồng đặc thù của Việt Nam (lúa, sầu riêng, cà phê) để hệ thống phục vụ tốt hơn cho nông nghiệp trong nước.
- Phát triển Mobile App Offline (Edge AI):** Đóng gói toàn bộ pipeline (YOLO + Classifier) vào ứng dụng di động sử dụng TensorFlow Lite hoặc ONNX Runtime để nông dân có thể sử dụng ngay tại vườn mà không cần internet.
- Hệ thống cảnh báo dịch bệnh:** Kết hợp dữ liệu GPS từ ảnh chụp để xây dựng bản đồ dịch bệnh theo thời gian thực, giúp cơ quan chức năng khoanh vùng và xử lý kịp thời.

Bảng 7: Tổng hợp kết quả so sánh 12 phương pháp trên ảnh thực tế (Ảnh phức tạp)

STT	Phương pháp	Dự đoán đúng?	Độ tin cậy (Max)	Đánh giá
<i>Nhóm 1: Chỉ Classification (Raw Input)</i>				
1	MobileNetV3 Only	Có	14.6%	Kém, nhiễu nền
2	EfficientNet-B0 Only	Không (Late Blight)	24.2%	Sai hoàn toàn
3	ResNet-18 Only	Không (Late Blight)	36.9%	Sai hoàn toàn
<i>Nhóm 2: Classification + Advanced Preprocessing</i>				
4	MobileNetV3 + Preprocess	Có	53.0%	Khá hơn nhưng chưa đạt
5	EfficientNet-B0 + Preprocess	Không (Grape)	81.9%	Sai loài cây
6	ResNet-18 + Preprocess	Không (Grape)	63.7%	Sai loài cây
<i>Nhóm 3: YOLOv11 + Classification</i>				
7	YOLOv11 + MobileNetV3	Có	92.3%	Rất Tốt
8	YOLOv11 + EfficientNet-B0	Có	92.3%	Tốt
9	YOLOv11 + ResNet-18	Có	82.8%	Tốt
<i>Nhóm 4: Full Pipeline (YOLOv11 + Preprocess + Classification)</i>				
10	YOLO + Pre + MobileNetV3	Có	92.3%	Rất tốt (nhưng Healthy bị lẫn)
11	YOLO + Pre + EfficientNet	Có	92.3%	Tốt
12	YOLO + Pre + ResNet-18	Có	82.8%	Ổn định & Chính xác nhất

TÀI LIỆU THAM KHẢO

- [1] Howard, A., Sandler, M., Chu, G., Chen, L. C., Chen, B., Tan, M., ... & Adam, H. (2019). Searching for MobileNetV3. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1314-1324.
- [2] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770-778.
- [3] Tan, M., & Le, Q. (2019). Efficientnet: Rethinking model scaling for convolutional neural networks. *International conference on machine learning*, PMLR, 6105-6114.
- [4] Vipooooool. (2024). New Plant Diseases Dataset. Kaggle. URL: <https://www.kaggle.com/datasets/vipooooool/new-plant-diseases-dataset>.
- [5] Grinberg, M. (2018). *Flask web development: developing web applications with python*. "O'Reilly Media, Inc.".
- [6] Banks, A., & Porcello, E. (2017). *Learning React: Functional Web Development with React and Redux*. "O'Reilly Media, Inc.".
- [7] Singla, A., Padolla, S., & Deng, J. (2019). PlantDoc: A Dataset for Visual Plant Disease Detection. *Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.