ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO CUỐI KÌ

Đề tài tìm hiểu: Phân vùng cải bó xôi, phân loại bệnh lá lúa

Môn học: Học Máy
Lớp: INT3405_52
Nguyễn Đức Thiện – 21020150
Ngành học: Kỹ thuật Robot

Giảng viên: PGS.TS.Phạm Minh Triển

Hà Nội, ngày 1 tháng 6 năm 2025

BÁO CÁO 1: PHÂN VÙNG CẢI BÓ XÔI

1. Giới thiệu và phân tích dữ liệu

- Mục tiêu: Phân vùng vùng lá cải bó xôi trong ảnh.
- Nguồn dữ liệu: Tập ảnh được cung cấp + gán nhãn bằng CVAT.
- Kích thước dữ liệu: Số lượng ảnh 180 ảnh
- Kiểu dữ liệu: Ảnh RGB lưu bằng file JPG

2. Tiền xử lý dữ liệu

- Gán nhãn: Thực hiện gán nhãn polygon shape trên CVAT, sau đó xuất ra dạng phù hợp để huấn luyện với model

Hình 1.1: Dữ liệu khi gán nhãn xong

- Chia dữ liệu: Train (144 ảnh), Val (36 ảnh)
- Sau khi phân chia dữ liệu thì nén dữ liệu bao gồm thư mục ảnh gốc; thư mục labels được xuất ra để huấn luyện và đường dẫn để sử dụng.
- Tải tệp dữ liệu lên để làm đầu vào
- Resize: Chuẩn hóa về kích thước 640x640 px.
- Chuẩn hoá pixel về [0, 1]

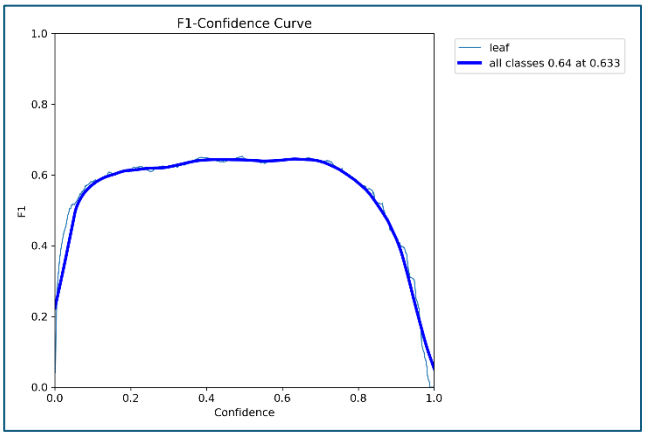
3. Trích xuất đặc trưng

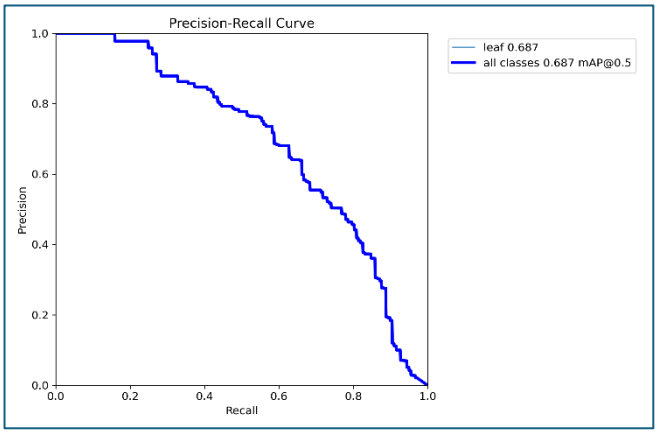
Mô hình YOLOv11 sử dụng backbone CNN để trích xuất đặc trưng từ ảnh, với các đặc trưng được tinh chỉnh trong quá trình huấn luyện.

4. Kết quả huấn luyện

Khi huấn luyện bằng model YOLOv11 sau 100 bước thì chúng ta sẽ có output bao gồm:

- Thư mục /runs: Chứa các kết quả huấn luyện và dự đoán của YOLOv11.
  - Thư mục segment: Chứa kết quả của tác vụ phân đoạn (segmentation), bao gồm dự đoán và huấn luyện.
  - Thư mục predict: Chứa các hình ảnh dự đoán (kết quả phân đoạn) trên tập kiểm tra hoặc tập dữ liệu mới.
  - Thư mục train: Chứa các file kết quả từ quá trình huấn luyện mô hình.
  - Thư mục weights: Chứa các file trọng số (weights) của mô hình sau huấn luyện.
- File yolo11n-seg.pt và yolo11n.pt: Các file mô hình YOLOv11 (phiên bản nano) được sử dụng hoặc tải về, có thể là mô hình gốc hoặc được tinh chỉnh.

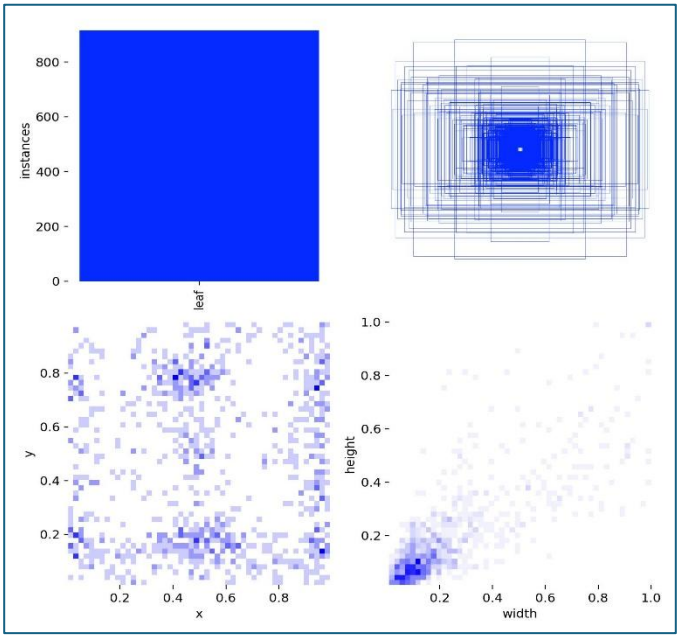




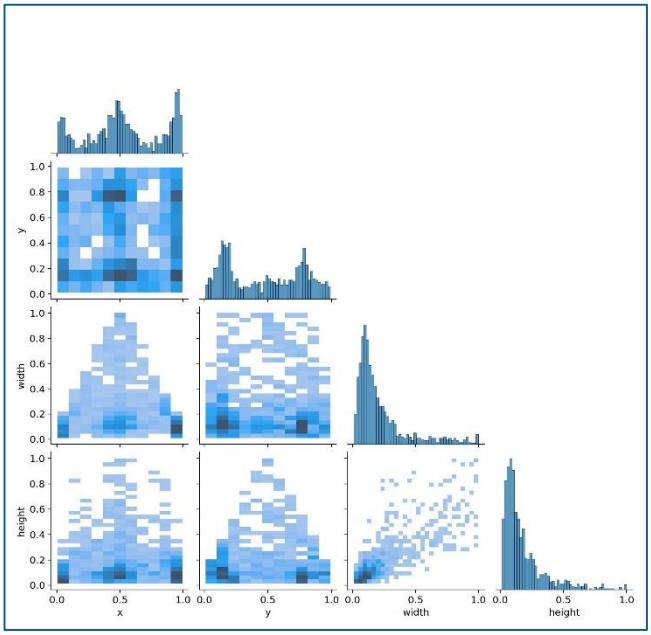
Hình 1.2:

Hình 1.3a: Hình ảnh hiển thị phân bố nhãn (labels) trong tập dữ liệu huấn luyện.

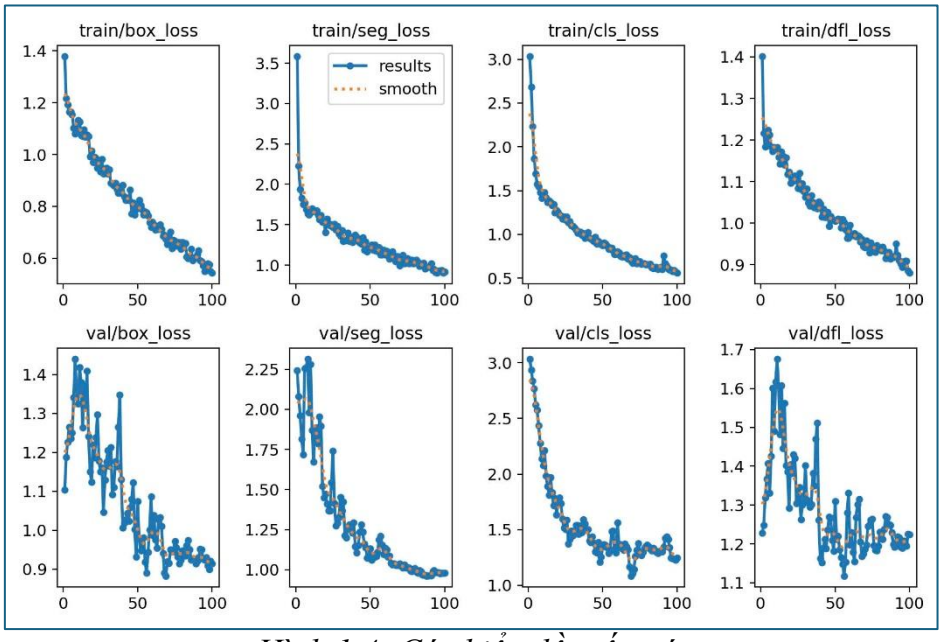
Hình 1.3b: Biểu đồ tương quan (correlogram) giữa các nhãn hoặc đặc trưng.



Hình 1.3a



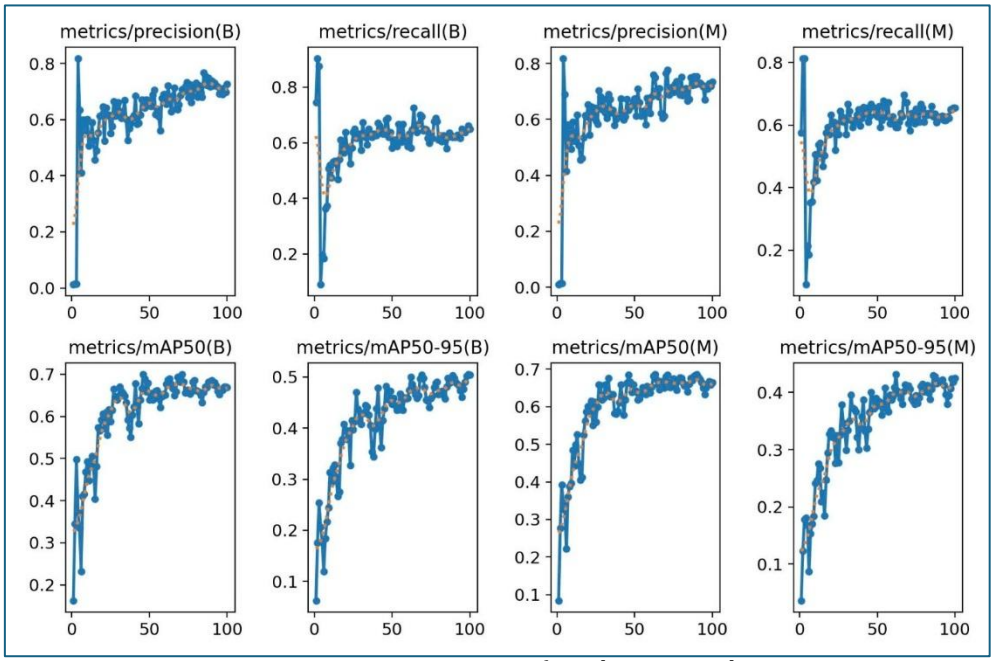
Hình 1.3b



Hình 1.4: Các biểu đồ mất mát

Đồ thị ở hình 1.4 thể hiện sự mất mát trong quá trình huấn luyện:

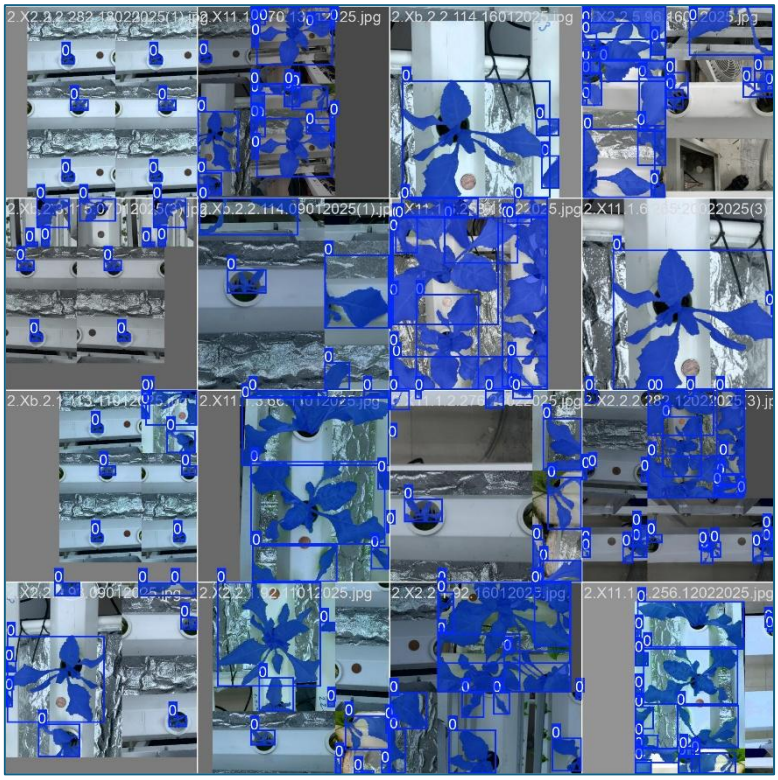
- train/box_loss, train/seg_loss, train/cls_loss, train/dfl_loss : Đây là các giá trị mất mát trong quá trình huấn luyện.
- val/box_loss, val/seg_loss, val/cls_loss, val/dfl_loss : Đây là các giá trị bị mất mát trên tập kiểm tra (xác thực).



Hình 1.5: Các biểu đồ hiệu suất

- precision(B): Độ chính xác (tỷ lệ dự đoán đúng trong số các dự đoán là tích cực).
- recall(B): Độ bao phủ (tỷ lệ các đối tượng thực sự được phát hiện).
- mAP50(B): Mean Average Precision ở ngưỡng IoU (Intersection over Union) = 0.5.
- mAP50-95(B): Mean Average Precision trung bình từ ngưỡng IoU 0.5 đến 0.95, phản ánh độ chính xác tổng quát hơn.

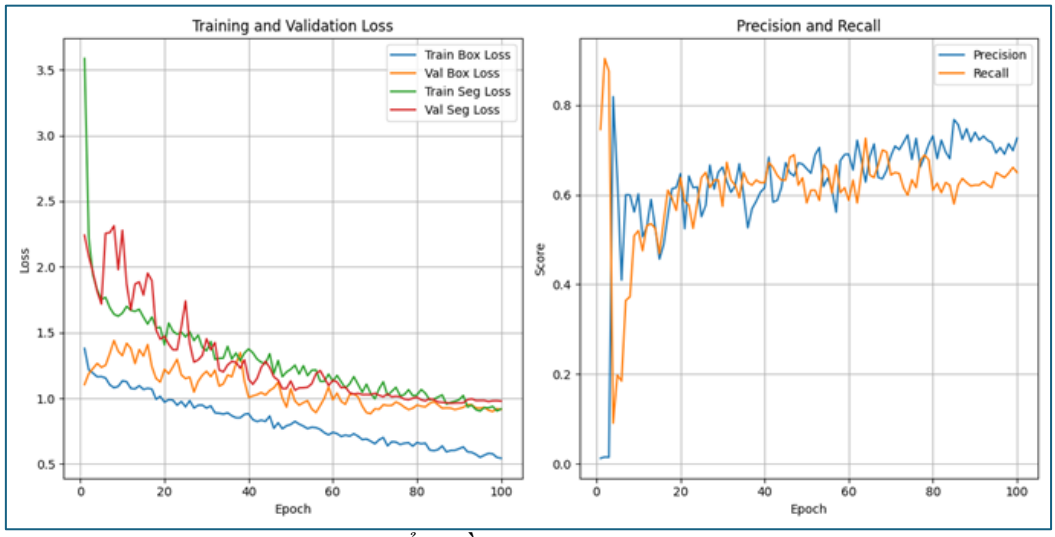
Ngoài các thông tin ảnh được lưu trữ trên, bên trong thư mục train còn chứa các file khác như: **results.csv**: File CSV chứa chi tiết các chỉ số hiệu suất qua từng epoch, hình ảnh minh họa các batch dữ liệu huấn luyện từ các epoch khác nhau như.



Hình 1.6: Batch dữ liệu huấn luyện bước 0

5. Đánh giá kết quả

A. Biểu đồ Loss và Precision/Recall:



Hình 1.7: biểu đồ Loss và Precision/Recall

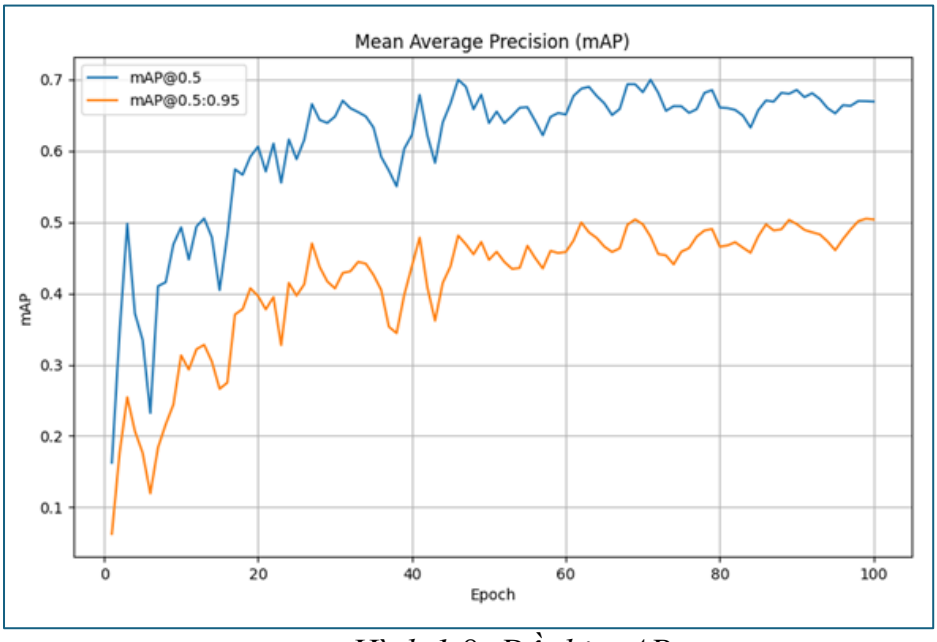
Đồ thị Loss: *Training and Validation Loss*:

- Box Loss và Segmentation Loss đều có xu hướng giảm ổn định, chứng tỏ mô hình học được đặc trưng tốt.
- Khoảng cách giữa Train và Validation Loss không quá xa → ít overfitting.
- Sau khoảng epoch 60, các giá trị loss hội tụ và dao động nhẹ → mô hình đã ổn định.

Đồ thị Precision và Recall:

- Cả Precision và Recall đều tăng ổn định trong 20–30 epoch đầu và dao động nhẹ sau đó.
- Precision đạt khoảng 0.75–0.8, Recall khoảng 0.65–0.7 → mức độ nhận diện đối tượng khá tốt.
- Chênh lệch giữa Precision và Recall nhỏ → mô hình khá cân bằng, ít thiên lệch.

B. Biểu đồ mAP

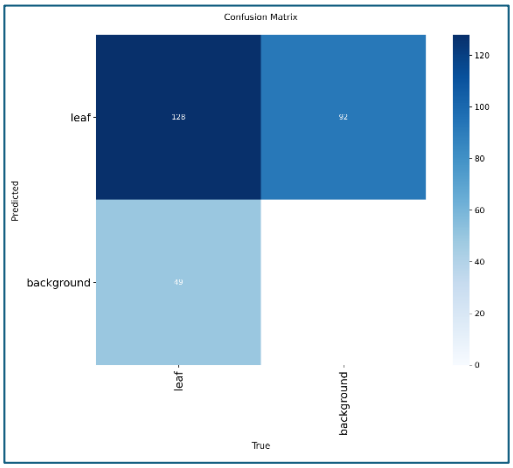


Hình 1.8: Đồ thị mAP

Cung cấp đánh giá rõ ràng về hiệu năng phân loại

- Cả Precision và Recall đều tăng ổn định trong 20–30 epoch đầu và dao động nhẹ sau đó.
- Precision đạt khoảng 0.75–0.8, Recall khoảng 0.65–0.7 → mức độ nhận diện đối tượng khá tốt.
- Chênh lệch giữa Precision và Recall nhỏ → mô hình khá cân bằng, ít thiên lệch.

C. Confusion Matrix



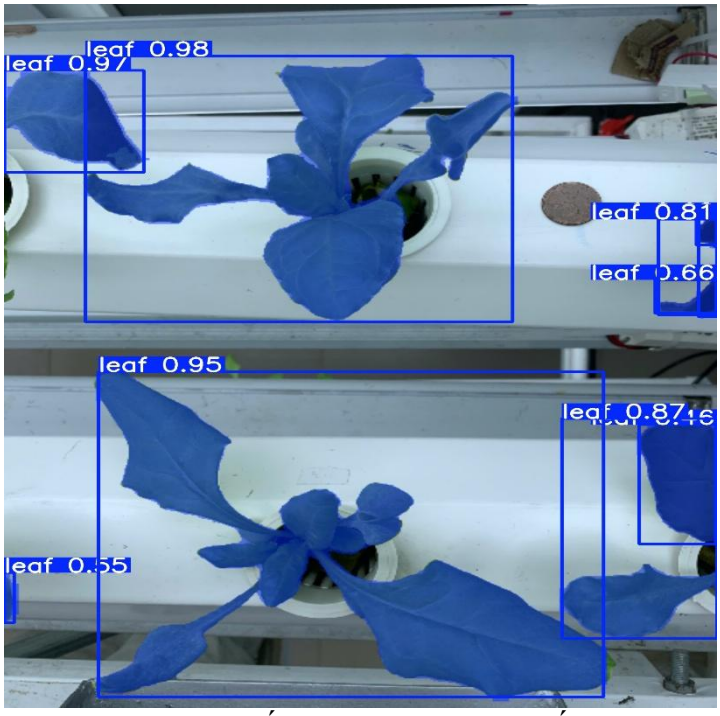
Hình 1.9: Confusion Matrix

Ma trận hiển thị 2 lớp: leaf và background.

- **True Leaf, Predicted Leaf**: 128 (đúng).
- **True Leaf, Predicted Background**: 92 (sai, nhầm lẫn cao).
- **True Background, Predicted Leaf**: 49 (sai, nhưng ít hơn).

Mô hình dự đoán tốt lớp leaf khi thực tế là leaf (128), nhưng nhầm lẫn đáng kể khi dự đoán background cho leaf (92). Điều này cho thấy mô hình có thể gặp vấn đề với việc phân biệt ranh giới giữa leaf và background, có thể do dữ liệu không đủ đa dạng hoặc nhãn không rõ ràng.

D. Kết quả test



Hình 1.10: Kết quả test sau khi huấn luyện

Vị trí	Nhãn	Confidence	Đánh giá mask & box
Trên bên trái	leaf	0.98	Bao khít hình lá, sát mép, rất chính xác
Trên giữa	leaf	0.92	Chính xác cao, mask trùm đều
Trên phải	leaf	0.81, 0.66	Tương đối ổn, nhưng có dấu hiệu gộp 2 lá thành 1 object
Dưới trái	leaf	0.55	Confidence thấp → có thể là nhiễu, hình lá nhỏ hoặc mờ
Dưới giữa	leaf	0.95	Rất tốt, mask đẹp và chính xác
Dưới phải	leaf	0.87	Bao tương đối chính xác, nhưng mask hơi thiếu viền lá nhỏ

Mô hình YOLO sau huấn luyện thể hiện hiệu quả tốt trong việc nhận diện và phân đoạn đối tượng "leaf". Tuy nhiên, vẫn cần cải thiện:

- Phân biệt lá riêng lẻ trong vùng dày đặc
- Loại bỏ dự đoán nhiễu có confidence thấp
- Tăng tính chi tiết cho mask viền lá

BÁO CÁO 2: PHÂN LOẠI BỆNH LÁ LÚA

1. Giới thiệu và phân tích dữ liệu

- Mục tiêu bài toán: Phân loại bệnh trên là lúa có 4 loại bệnh chính: Đốm nâu, bạc lá vi khuẩn, đạo ôn, vàng lùn.
- Nguồn dữ liệu sử dụng: Kaggle – Rice Leaf Disease Images
- Bộ dữ liệu này bao gồm khoảng 5932 ảnh được chia thành 4 loại bệnh chính cần phân loại kể trên
    - Tungro – 1308 ảnh (Bệnh vàng lùn)
    - Bacterialblight – 1584 ảnh (Bệnh bạc lá vi khuẩn)
    - Blast – 1440 ảnh (Bệnh đạo ôn lá)
    - BRownspot – 1600 ảnh (Bệnh đốm nâu)
- Kiểu dữ liệu: Ảnh RGB.

2. Tiền xử lý dữ liệu

Trong quá trình huấn luyện mô hình học sâu, tiền xử lý dữ liệu đóng vai trò quan trọng để đảm bảo chất lượng đầu vào ổn định, tăng độ chính xác và giảm hiện tượng overfitting. Trong đoạn mã dưới đây, ta sử dụng lớp ImageDataGenerator của Keras để thực hiện cả chuẩn hóa dữ liệu và tăng cường dữ liệu (data augmentation):

Kỹ thuật	Mô tả
rescale=1./255	Chuẩn hóa giá trị pixel từ [0, 255] về [0, 1] nhằm giúp mô hình học hiệu quả hơn.
validation_split=0.2	Tự động chia 20% dữ liệu từ thư mục thành tập validation.
rotation_range=20	Xoay ngẫu nhiên ảnh trong khoảng ±20 độ.
zoom_range=0.2	Phóng to/thu nhỏ ảnh ngẫu nhiên trong khoảng ±20%.
horizontal_flip=True	Lật ảnh theo chiều ngang giúp mô hình học được tính đối xứng.
shear_range=0.2	Biến dạng ảnh theo trục chéo, mô phỏng sự biến đổi góc chụp.
width_shift_range=0.2	Dịch chuyển ảnh theo chiều ngang (tối đa ±20%).
height_shift_range=0.2	Dịch chuyển ảnh theo chiều dọc (tối đa ±20%).
brightness_range=[0.8, 1.2]	Tăng/giảm độ sáng của ảnh trong khoảng từ 80% đến 120%.

Sau khi thiết lập các phép biến đổi, dữ liệu ảnh sẽ được nạp từ thư mục chứa các lớp. Trong bài toán này chúng ta sẽ có 4 lớp đông thời chia tập dữ liệu thành như sau để tiến hành huấn luyện, kiểm thử và kiểm tra khi huấn luyện xong

Số lượng ảnh train: 3799

Số lượng ảnh validation: 948

Số lượng ảnh test: 1185

3. Trích xuất đặc trưng

Trong bài toán sử dụng 6 model và mỗi loại sẽ có những đặc trưng riêng

A. Phân tích chi tiết từng mô hình

Mô hình	Cách trích xuất đặc trưng	Ưu điểm	Nhược điểm
1. CNN tự xây	Dùng các lớp Conv2D, MaxPooling, Flatten để học đặc trưng trực tiếp từ dữ liệu. Đặc trưng thường đơn giản, tổng quát.	Dễ tuỳ chỉnh, nhẹ, phù hợp bài toán nhỏ.	Độ chính xác thường thấp hơn, khó học đặc trưng phức tạp.
2. MobileNetV2	Dựa trên kiến trúc nhẹ sử dụng khối depthwise separable convolution và linear bottleneck. Trích đặc trưng tiết kiệm tài nguyên.	Nhẹ, nhanh, tối ưu cho thiết bị di động.	Độ chính xác thấp hơn so với mô hình lớn
3. ResNet50	Sử dụng residual block để trích đặc trưng sâu mà tránh hiện tượng gradient vanish. Mạng sâu hơn học được đặc trưng trừu tượng.	Học được đặc trưng sâu, tổng quát tốt.	Mạng phức tạp, nặng, tốc độ chậm hơn.
4. DenseNet201	Dùng dense connections, mỗi lớp nhận đầu vào từ tất cả lớp trước => trích xuất đặc trưng phong phú, tái sử dụng nhiều.	Hiệu suất cao, ít thông số hơn ResNet cùng độ sâu.	Rất tốn bộ nhớ và thời gian huấn luyện.
5. EfficientNetB0	Cân bằng độ sâu, chiều rộng và độ phân giải bằng AutoML. Trích xuất đặc trưng hiệu quả cao, được tối ưu toàn diện.	Cân bằng tốt giữa hiệu năng và chi phí tính toán.	Cấu trúc khó tùy chỉnh, cần tải pretrained model.
6. VGG16	Dùng 13 lớp conv và 3 lớp dense. Trích xuất đặc trưng tuần tự từ đơn giản đến phức tạp.	Đơn giản, dễ hiểu, hiệu quả ổn định.	Mạng rất nặng, nhiều tham số, dễ overfitting nếu dữ liệu nhỏ.

B. Quy trình trích xuất đặc trưng

Khi sử dụng mô hình CNN, quy trình trích xuất đặc trưng thường gồm các bước sau:

1. Resize ảnh về đúng kích thước yêu cầu của từng mô hình (224x224, v.v).
2. Chuẩn hóa ảnh về [0, 1] hoặc chuẩn hóa mean/variance tùy theo yêu cầu pretrained model.
3. Tải mô hình gốc (thường là bản pretrained trên ImageNet).
4. Bỏ phần top (fully connected layers) nếu chỉ cần trích đặc trưng.
5. Trích đặc trưng từ ảnh thông qua mô hình bằng model.predict() hoặc model(x).

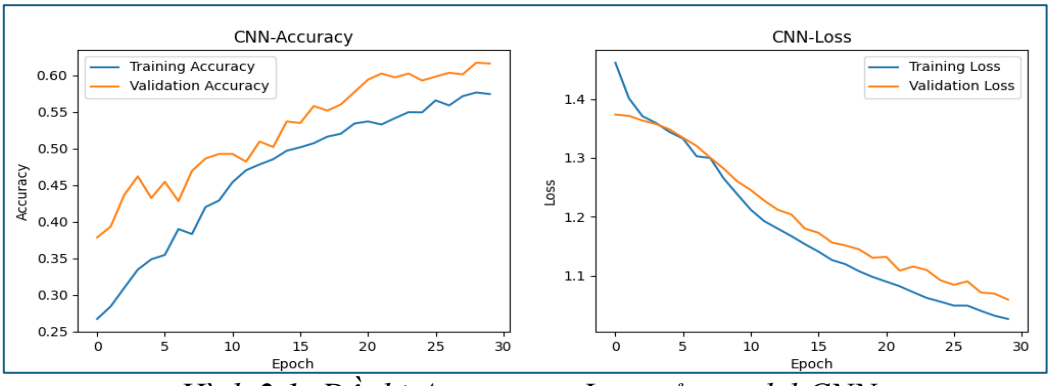
C. Bảng tổng hợp so sánh trích xuất đặc trưng

Mô hình	Kích thước đầu vào	Số tham số	Trích đặc trưng qua	Pretrained	Tốc độ xử lý	Hiệu suất (Accuracy)
CNN tự xây	128x128	~0.1 triệu	Conv + Pool + Flatten	Không	Nhanh	Trung bình
MobileNetV2	224x224	~3.5 triệu	Depthwise Separable	Có	Rất nhanh	Tốt
ResNet50	224x224	~25 triệu	Residual Block	Có	Trung bình	Cao
DenseNet201	224x224	~20 triệu	Dense Block	Có	Chậm	Rất cao
EfficientNetB0	224x224	~5 triệu	Compound Scaling	Có	Nhanh	Trung bình
VGG16	224x224	~138 triệu	Conv Layer tuần tự	Có	Rất chậm	Khá tốt

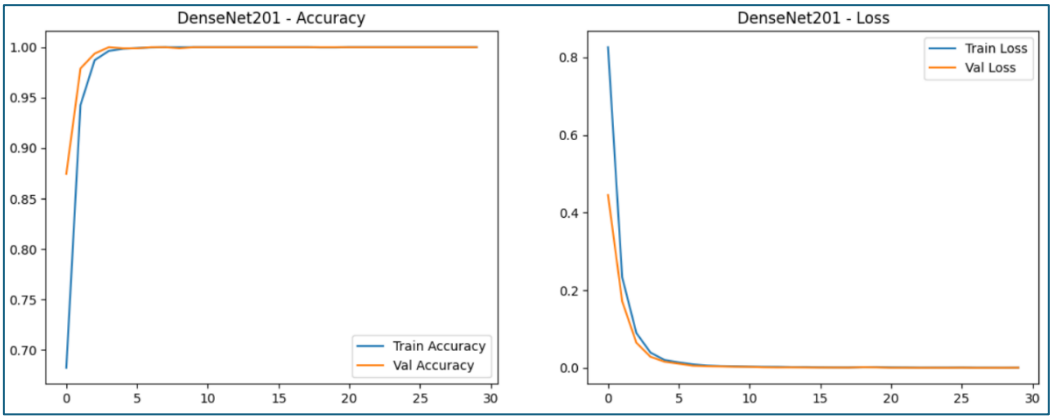
4. Kết quả huấn luyện

Sau đây là kết quả khi huấn luyện các mô hình trong EPOCHS = 30 với BATCH_SIZE = 64

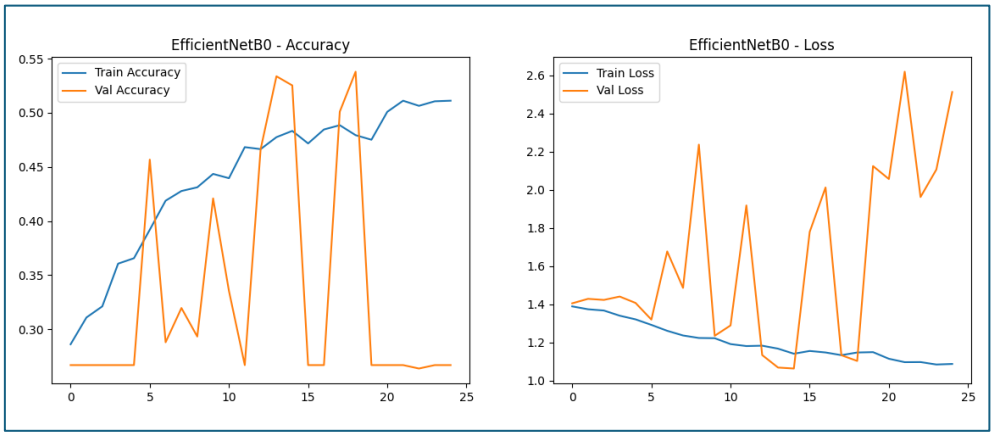
A. Đánh giá bằng: Accuracy, loss



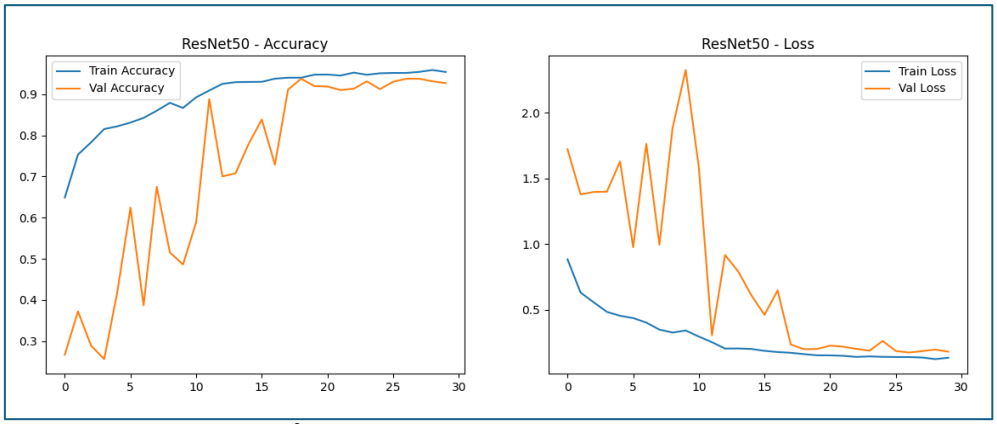
Hình 2.1: Đồ thị Accuracy – Loss của model CNN



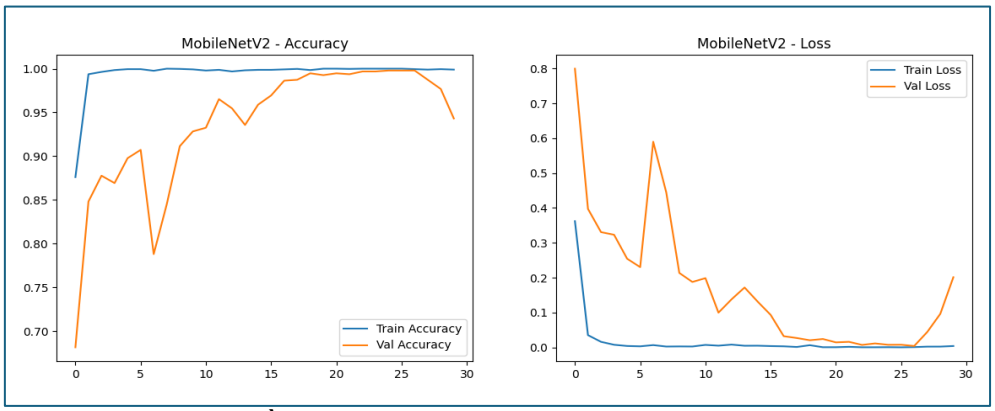
Hình 2.2: Đồ thị Accuracy – Loss của model DenseNet201



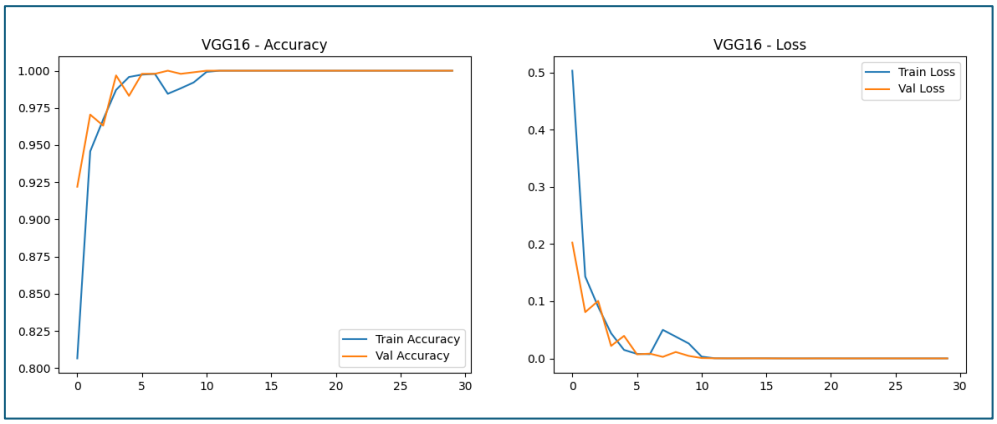
Hình 2.3: Đồ thị Accuracy – Loss của model EfficientNetBo



Hình 2.4: Đồ thị Accuracy – Loss của model ResNet50

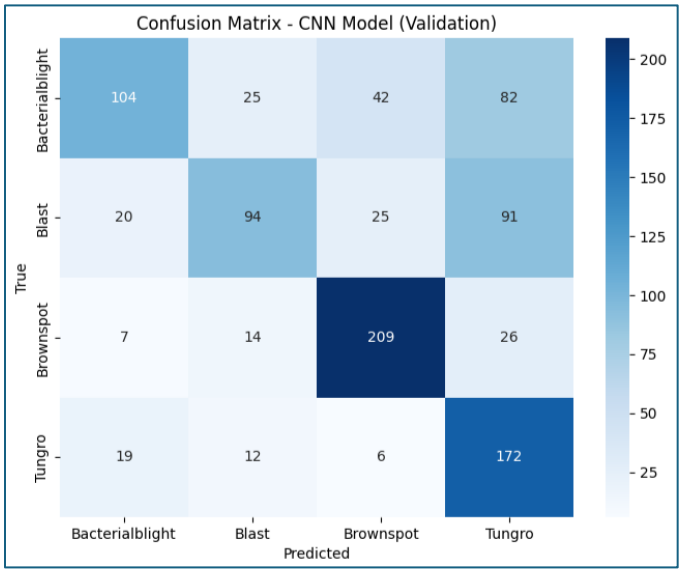


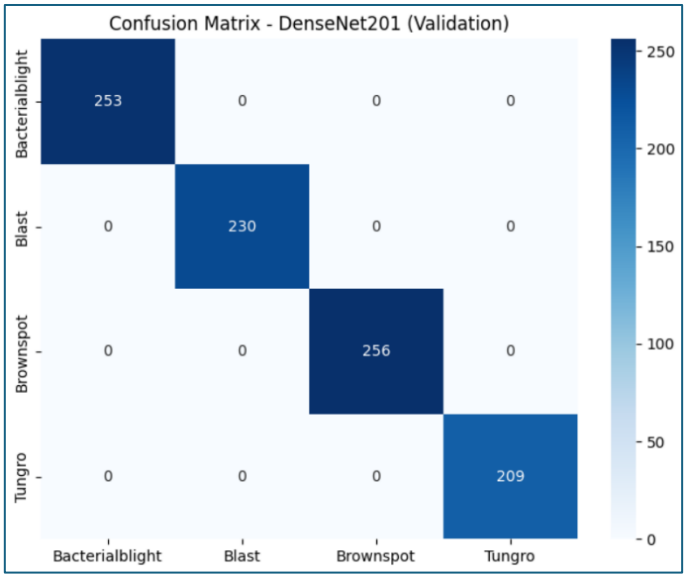
Hình 2.5: Đồ thị Accuracy – Loss của model MobileNetV2

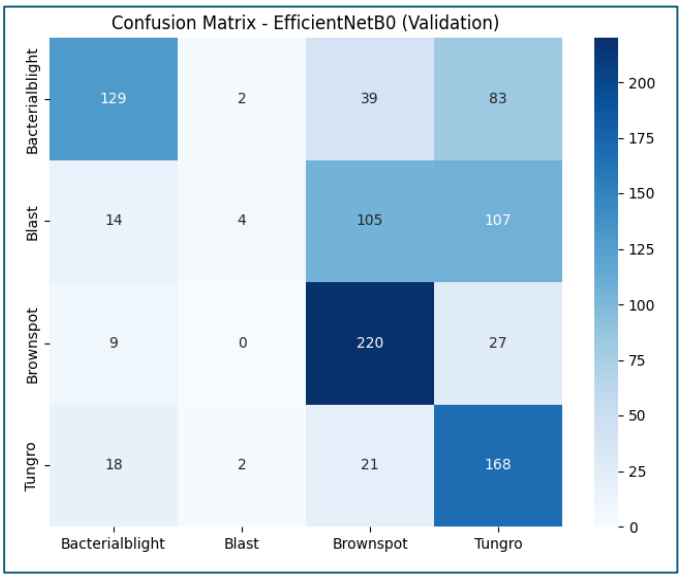


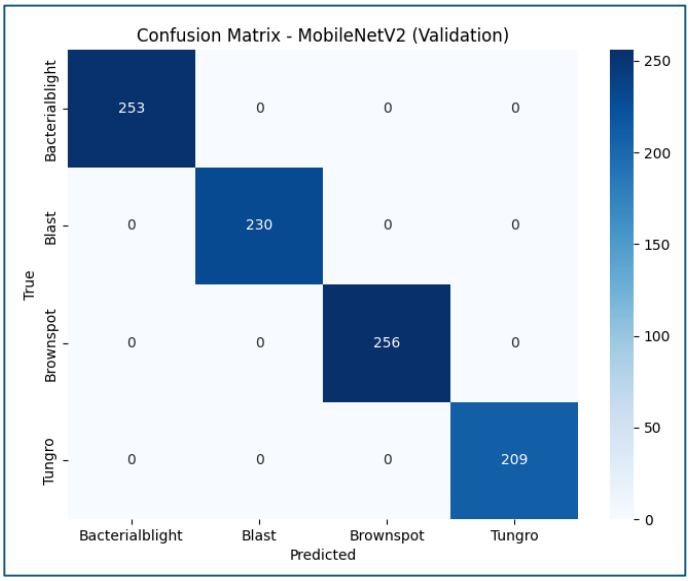
Hình 2.6: Đồ thị Accuracy – Loss của model VGG16

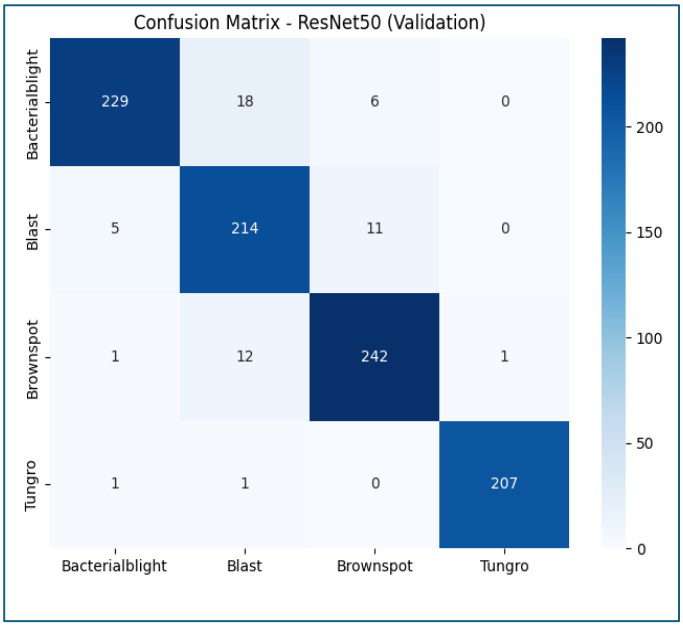
B. Confusion Matrix (Validation)

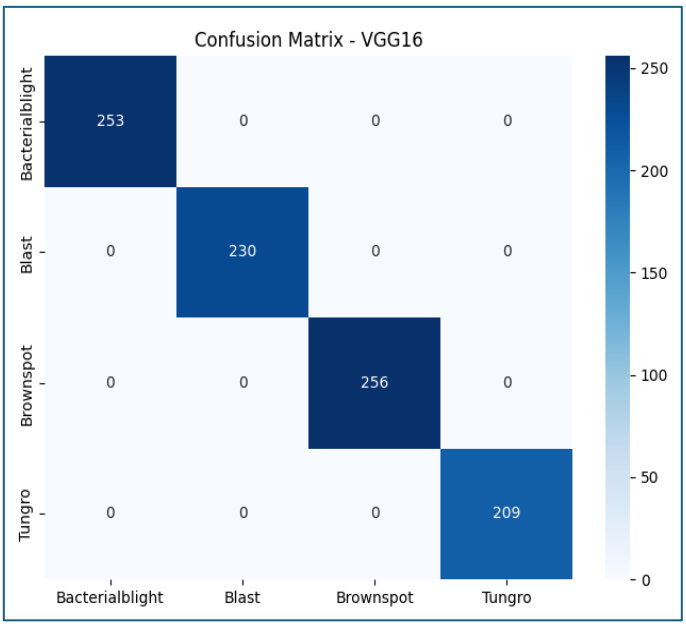




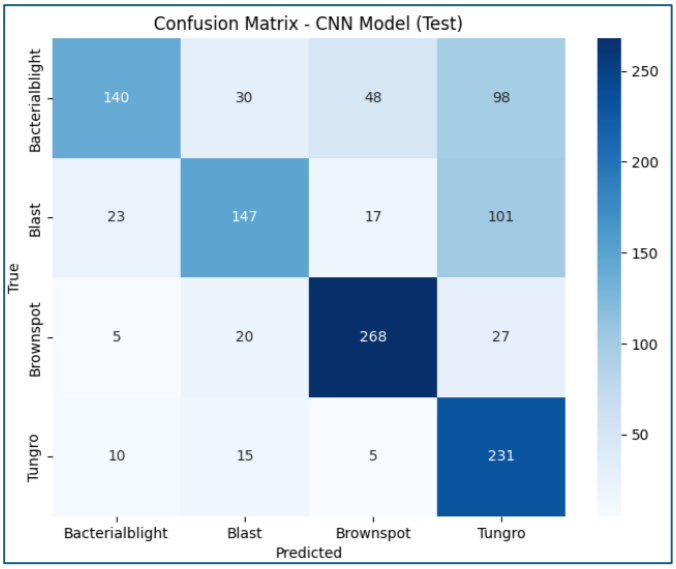


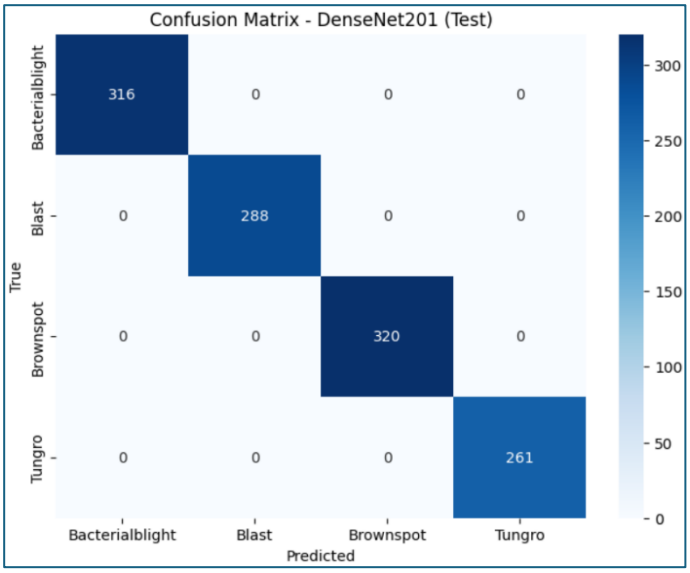


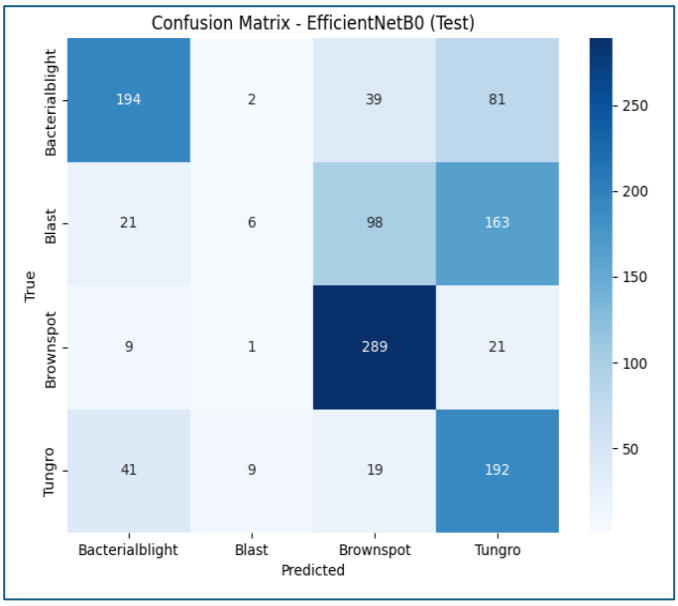


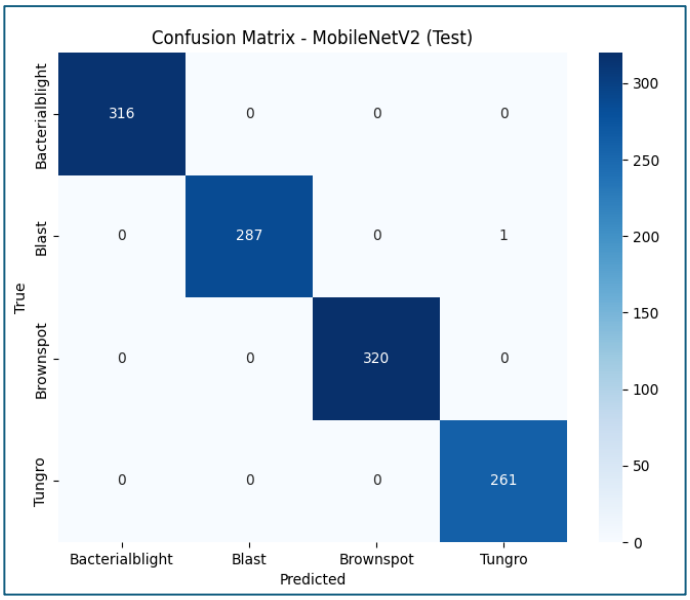


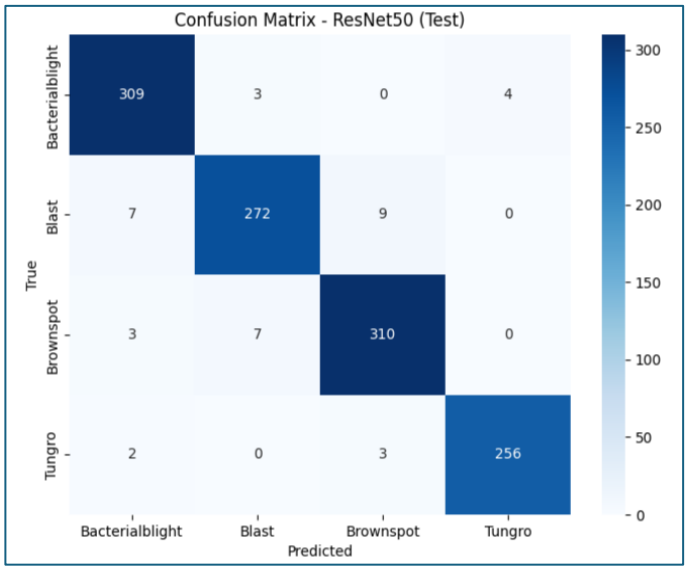
C. Confusion Matrix (Test)

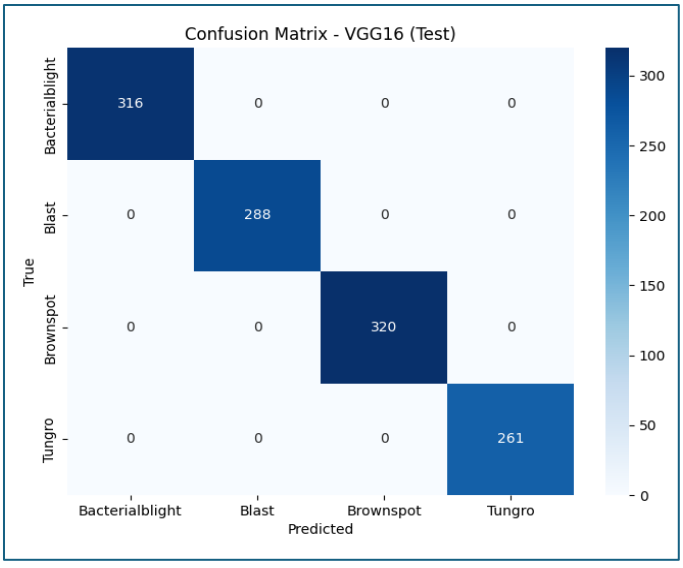








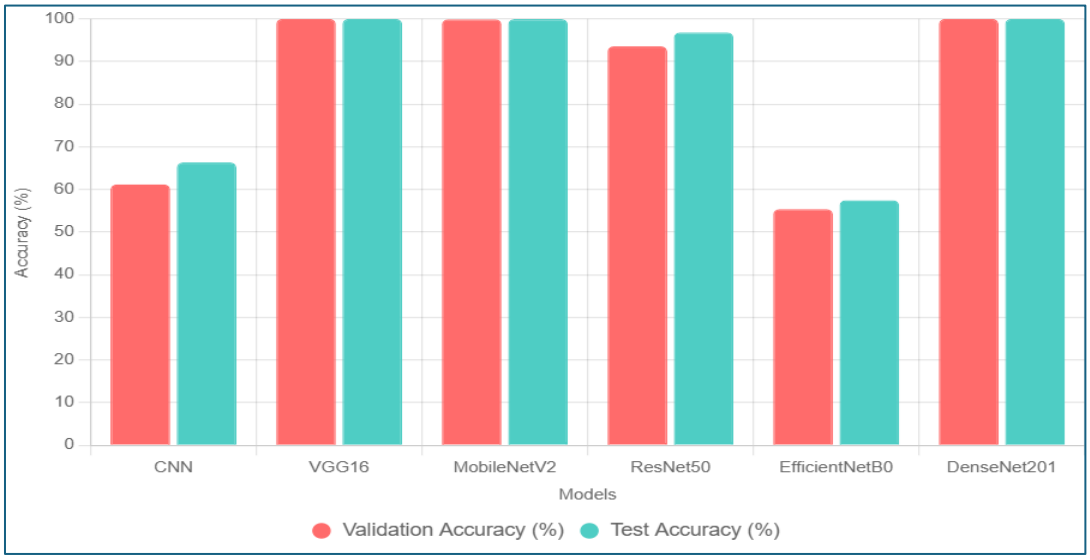




5. Đánh giá mô hình

Bảng thống kê kết quả Validation Loss, Validation Accuracy, Test Accuracy, Test Loss khi huấn luyện EPOCHS = 30 với BATCH_SIZE = 64

Mô hình	Validation Accuracy (%)	Validation Loss	Test Accuracy(%)	Test Loss
CNN	61.18	1.0269	66.33	1.0160
VGG16	100.00	0.0002	100.00	0.0002
MobileNetV2	0.9989	0.0036	99.92	0.0041
ResNet50	93.57	0.1646	96.79	0.1065
EfficientNetB0	55.38	1.0577	57.47	0.9913
DenseNet201	100.00	0.0002	100.00	0.0002



Hình 2.7: Biểu đồ so sánh

- Các mô hình có hiệu suất cao :
  - ✓ VGG16 và DenseNet201 đạt được độ chính xác xác thực và kiểm tra hoàn hảo (100%) với giá trị mất mát rất thấp (0,0002). Điều này cho thấy khả năng khái quát hóa và phù hợp tuyệt vời với tập dữ liệu, cho thấy các mô hình này có hiệu quả cao đối với nhiệm vụ phân loại này.
  - ✓ MobileNetV2 cũng có hiệu suất cực kỳ tốt với độ chính xác xác thực là 99,89% và độ chính xác thử nghiệm là 99,92%, với mức tổn thất tối thiểu (0,0036 và 0,0041), cho thấy hiệu suất mạnh mẽ với kiến trúc nhẹ.
- Các mô hình tốt:
  - ✓ ResNet50 mang lại hiệu suất vững chắc với độ chính xác xác thực 93,57% và độ chính xác thử nghiệm 96,79%, với mức mất mát cao hơn một chút (0,1646 và 0,1065). Điều này cho thấy khả năng khái quát hóa tốt nhưng không hoàn hảo.

- Các mô hình chưa hiệu quả :
  - ✓ CNN cho thấy hiệu suất thấp nhất với độ chính xác xác thực 61,18% và độ chính xác thử nghiệm 66,33%, cùng với các giá trị mất mát cao hơn (1,0269 và 1,0160). Điều này cho thấy kiến trúc CNN cơ bản có thể không đủ cho nhiệm vụ này.
  - ✓ EfficientNetB0 có kết quả kém nhất với độ chính xác xác thực là 55,38% và độ chính xác thử nghiệm là 57,47%, cùng mức mất mát cao hơn (1,0577 và 0,9913), cho thấy phương pháp này gặp khó khăn trong việc học tập dữ liệu một cách hiệu quả.

Trong quá trình khi huấn luyện EPOCHS = 15 với BATCH_SIZE = 32 và huấn luyện đồng thời các mô hình thì thu được kết quả như sau:

Mô hình	**Validation Accuracy (%)**	**Validation Loss**
CNN	57.52	1.0526
VGG16	65.32	0.6532
MobileNetV2	86.73	0.3833
ResNet50	42.95	1.2524
EfficientNetB0	27.00	1.3827
DenseNet201	89.11	0.3430

- Hiệu suất nhìn chung giảm đáng kể ở hầu hết các mô hình, đặc biệt là các mô hình sâu như ResNet50 và VGG16. Điều này là dễ hiểu vì:
  - ✓ Số epoch giảm một nửa → mô hình chưa được học đủ → chưa hội tụ.
  - ✓ Batch size giảm cũng khiến gradient cập nhật nhiễu hơn → quá trình tối ưu không ổn định.
- Mô hình ổn định hơn khi giảm epochs:
  - ✓ DenseNet201 và MobileNetV2 vẫn giữ hiệu suất tốt → có khả năng học nhanh hơn, hội tụ tốt hơn trong số lượng bước huấn luyện ít hơn.
  - ✓ MobileNetV2 và DenseNet201 đều có Validation Loss thấp nhất, cho thấy không chỉ chính xác mà còn tổng quát tốt.
- VGG16, ResNet50 và EfficientNetB0 bị ảnh hưởng nặng:
  - ✓ Các mô hình này có kiến trúc sâu và phức tạp hơn, yêu cầu nhiều epoch để hội tụ.
  - ✓ Có thể cần batch size đủ lớn để tận dụng ưu thế học sâu.