B.M.S. COLLEGE OF ENGINEERING, BENGALURU
Autonomous Institute, Affiliated to VTU

An AAT Report on
on

"Deployable Deep Learning for Cross-Domain Plant Leaf Disease Detection via Ensemble Learning, Knowledge Distillation, and Quantization"

Submitted in fulfilment for the award of degree of
Master of Technology
In
Computer Science and Engineering

Submitted by

MOHAMMED SULAIMAN I (1BM25SCS015)

Under the Guidance of
Dr. Umadevi V
Professor, BMSCE

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
2025-2026

B. M. S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

This is to declare that the AAT work entitled "Deployable Deep Learning for Cross-Domain Plant Leaf Disease Detection via Ensemble Learning, Knowledge Distillation, and Quantization" carried out by MOHAMMED SULAIMAN I (1BM25SCS015), is a part of the course Artificial Intelligence (MCS101) of 1st Semester M.Tech (CSE), Department of Computer Science and Engineering, B.M.S. College of Engineering, Bangalore. This AAT work has been carried out under the guidance of Dr. Umadevi V during the academic semester Nov 2025 - Feb 2026. I declare that this work is original and has not been submitted elsewhere for any other degree or award.

Signature of the Candidate

MOHAMMED SULAIMAN I (1BM25SCS015)

B. M. S. COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the AAT entitled "Deployable Deep Learning for Cross-Domain Plant Leaf Disease Detection via Ensemble Learning, Knowledge Distillation, and Quantization" is a bonafide work carried out by MOHAMMED SULAIMAN I (1BM25SCS015), in partial fulfillment for the award of Master of Technology in Computer Science and Engineering of the Visvesvaraya Technological University, Belagavi, during the academic year 2025-2026. This report satisfies the academic requirements prescribed for the said degree.

Guide

Dr. Umadevi V
Professor
Department of CSE
BMSCE

TABLE OF CONTENTS

1. Introduction

Do you know how plant diseases destroy crops all over the world? Honestly, it's awful; studies have shown that infections take away almost 40% of the possible harvest every year. Just think of tomatoes, which farmers in Kolar and places around it grow all over the place. Fungi, bacteria, viruses, the whole nasty gang beat them up. A lucky break? Usually, strange spots or discoloration of leaves is how those bugs declare their presence. That should make early detection easy, right?

The problem is that today, we rely mostly on agronomists who are squinting at leaves. That is fine, until you realize that most smallholder farmers in villages can't call a specialist when their crops start looking sick; it takes ages and two experts may argue over the same leaf.

We created a robust tomato disease detector for the express purpose of operating consistently under both pristine laboratory and demanding agricultural settings. Our tool goes beyond that simple classifier by incorporating explainability, and it will show farmers in clear terms why a given diagnosis, such as blight, was made through a process not involving black-box guesswork.

Most importantly, we made the entire system compatible with low-cost Android phones at about ₹5,000 sans specialized hardware. Packaged as a proper offline app, though currently not hosted on the Play Store, this helpful tool was designed for farmers to begin using immediately by placing pragmatism over demonstrating just a conference paper novelty.

2. Problem Statement Addressed

When I moved from lab to field conditions during my first model testing, I found a significant decline in performance. Models that were 99.91% accurate on PlantVillage became only 32–46% accurate on field photos, which would essentially make them useless for farmers. Four major issues are responsible for the dramatic decline in model performance when transferring from lab to field settings:

1. Inconsistency between training and real-world conditions: The majority of models struggle with field photos that have uneven lighting, natural backgrounds (such as soil or other plants), and leaves at random angles because they were trained on clean, studio-style images, frequently with plain white backgrounds.
2. On the mid-range or older smartphones smallholder farmers typically possess, high computational requirements On the older or mid-range smartphones that have low computational requirements, several deep learning models that are able to perform well are far too large and resource hungry to execute.
3. Lack of transparency is understandable that farmers would be hesitant to listen to a system that gives advice. Credibility is essential when decisions are made that affect crop inputs, costs or pesticide application and black box forecasts do not encourage.
4. Overoptimistic testing a vast proportion of studies find accuracy by using data of only one source. that causes artificially high expectations of performance and that do not account for model behavior. diversity of bad weather conditions.

3. Objectives of the Paper

This implementation was with the aim of achievement of the following:

1. Design a practical and robust. disease identification system of tomato leaf, in which the operation is apt in a laboratory. It was tested by performing rigorous cross-domain validation between controlled datasets and real-field images.
2. Perform a comparison among different deep learning models, based on: generalization across diverse image sources and computational efficiency, including model size and inference speed.
3. Grad-CAM++ and LIME have been applied in efforts to understand the role of spatial and causal. data in the choices of the model.
4. Optimal model was compressed and an export of the model as ONNX format with INT8 was done. This significantly decreased the size and memory, and the accuracy of the diagnosis was maintained to nearly the same degree.
5. This will entail designing a working mobile app with Flutter that will be ran fully offline on a smartphone. That means farmers can diagnose crop diseases instantly without needing access to the internet, which is critical in either remote or resource-poor environments.
6. A fair and reproducible evaluation protocol was established, allowing only comparisons of diseases that were represented in both the laboratory and field datasets. This will avoid inflated accuracy due to label mismatches, and better approximate how well models may perform in the wild.

4. Datasets Description

In this research, two complementary datasets were used:

a. PlantVillage Dataset:

PlantVillage acted as the source of the training data, which includes laboratory-acquired, controlled images of diseases on tomato leaves under normalized conditions. There are 14,543 high-resolution images in the dataset, grouped into 10 classes.

Link: https://www.kaggle.com/datasets/emmarex/plantdisease

Disease Class Label	Training Images	Testing Images	Validation Images
Bacterial Spot	1362	0	340
Early Blight	640	0	160
Late Blight	1222	0	305
Leaf Mold	609	0	152
Septoria Leaf Spot	1145	0	286
Spider Mites (Two-Spotted)	1073	0	268
Target Spot	898	0	225
Tomato Yellow Leaf Curl Virus	3429	0	857
Tomato Mosaic Virus	239	0	60
Healthy	1018	0	255
Total	**11,635**	**0**	**2,908**

Table 4.1: Class-wise Distribution of PlantVillage Dataset (Training Set)

Images were captured in strictly controlled laboratory conditions with a uniform white background, standardized illumination, and optimal leaf position. Class distribution is considerably imbalanced, from 373 images in the tomato mosaic virus class to 5,357 images in the class that represents the tomato yellow leaf curl virus.

b. TomatoVillage Dataset:

The TomatoVillage dataset provided the necessary field-validation component for the cross-domain evaluation. It consists of 1,616 images across 8 disease and nutrient-deficiency classes representative of natural field conditions that farmers find themselves in.

Link: https://www.kaggle.com/datasets/iluvvatar/tomato-leaf-disease

Disease Class Labels	Training Images	Testing Images	Validation Images
Leaf Miner Damage	0	—	0
Magnesium Deficiency	0	—	0
Late Blight	0	904	0
Spotted Wilt Virus	0	—	0
Early Blight	0	496	0
Nitrogen Deficiency	0	—	0
Potassium Deficiency	0	—	0
Healthy Leaves	0	216	0
Total	**0**	**1,616**	**0**

Table 4.2: Class-wise Distribution of TomatoVillage Dataset (Testing Set)

Contrasting the controlled PlantVillage images, the TomatoVillage samples were captured in real agricultural settings with natural variation in illumination conditions, leaf orientations, background complexity, and imaging distances. Although some images maintain white backgrounds for easier annotation, this dataset maintains realistic disease symptoms and leaf textures typical of field conditions. The class distribution ranges from 72 images for Potassium Deficiency (1.6%) to 1,024 images for Leaf Miner (22.6%), showing the prevalence pattern found in reality rather than balanced sampling.

5. Tools and Technology used for implementation

The implementation used an extensive technology stack in the development, training, optimization, and deployment stages as detailed below:

a. Core Development Framework:

- Python 3.9 with PyTorch 2.0.1 and torchvision 0.15.2
- Scikit-learn 1.3.0 for evaluation metrics and data handling
- NumPy 1.24.3 and Pandas 2.0.2 for data manipulation
- Matplotlib 3.7.1 and Seaborn 0.12.2 for visualization

b. Deep Learning Components:

- TIMM (PyTorch Image Models) library v0.9.2 for pre-trained architectures
- Torchvision with ImageNet pre-trained weights for transfer learning
- NVIDIA CUDA 11.8 with cuDNN for GPU acceleration
- PyTorch Lightning for training pipeline management

c. Explainable AI Implementation:

- Grad-CAM++ implementation using custom PyTorch hooks
- LIME (Local Interpretable Model-agnostic Explanations) v0.2.0.1
- OpenCV 4.7.0: to process images and provide a visualization of heatmaps

d. Model Optimization and Deployment:

- ONNX Runtime 1.15.1 for model conversion and quantization
- ONNX Runtime Mobile for Android deployment
- Flutter 3.13.0 with Dart 3.1.0 for developing cross-platform mobile applications
- Android Studio with Java JDK 17 for the Android build pipeline
- Visual Studio Code as the main IDE with the Pylance

e. Infrastructure:

- Google Cloud Platform Compute Engine VM: NVIDIA T4 GPU, 16 GB RAM
- GitHub for version control and collaboration
- Weights & Biases for experiment tracking and visualization

6. GitHub link of the code

https://github.com/Syu607/AI-AAT

7. Methodology

This study used a structured methodology involving five phases that are interlinked as follows:

Stage 1: Data Preparation and Cross-Domain Strategy

- Conducted stratified partitioning of data (80% to train and 20% for validation) on the PlantVillage dataset.
- Created a class-mapping protocol for disease category overlaps among the datasets by normalizing class identifiers: converting to lowercase, standardizing underscores, removing prefixes.
- Conducted minimal preprocessing on the images by resizing them to a size of 224×224 pixels and applying ImageNet normalization.
- Created a stringent crossdomain assessment framework testing only on TomatoVillage and ensuring no domain leakage.

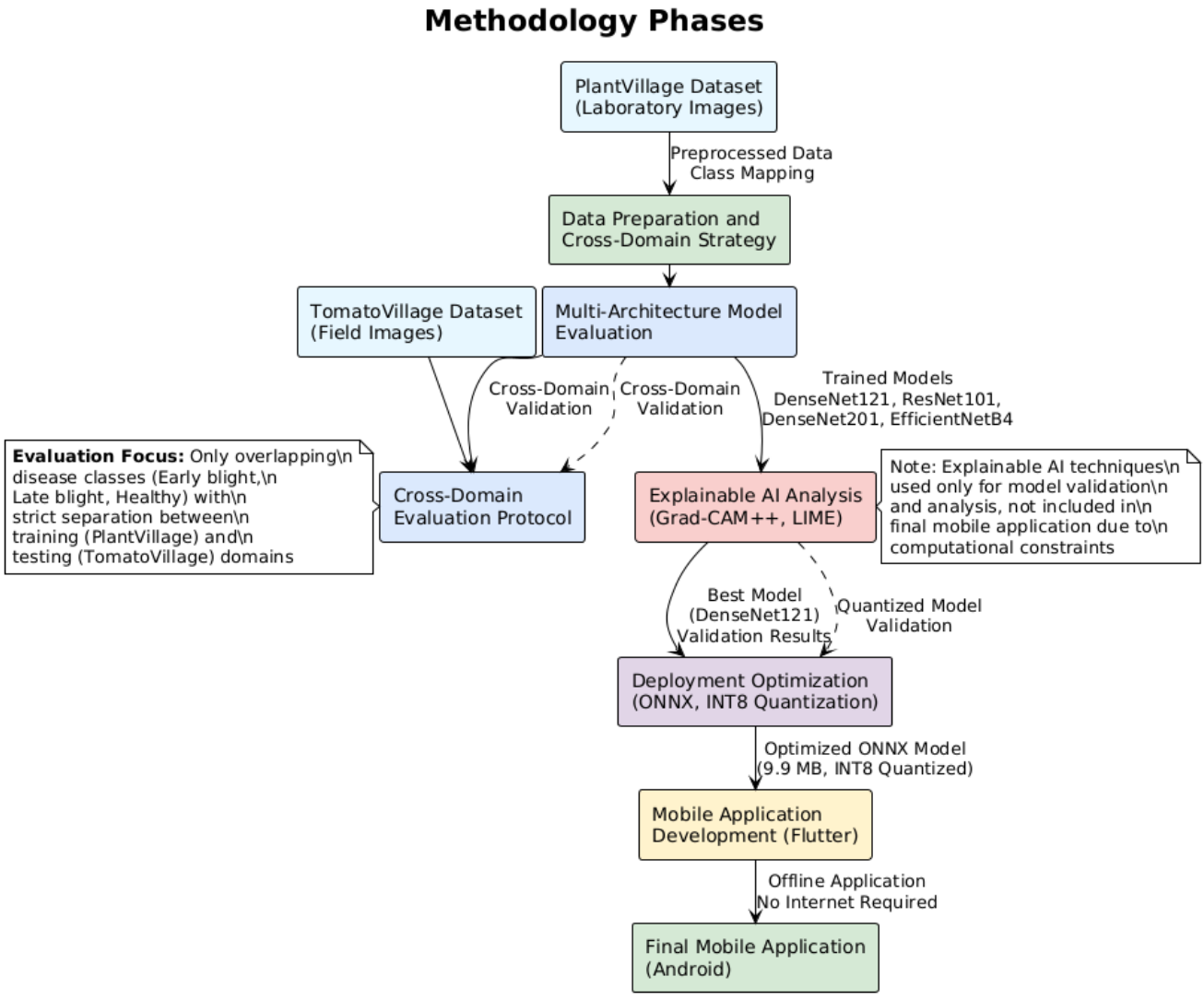


Figure 7.1 Methodology Phases

Stage 2: A Multi-Architecture Model Evaluation

a. The transfer learning process involved four different CNN models:

- DenseNet121: Famous for its dense connectivity pattern which favors feature reuse.
- ResNet101: Incorporating residual connections to facilitate deeper net architectures.
- DenseNet201: Another Dense Net with higher depth.
- EfficientNetB4: A compound scaling approach is utilised for efficiency enhancement.

b. The training protocol was standardized among architectures:

- The Optimizer is Adam with a learning rate of 1e-4.
- Batch size: 32 with gradient accumulation to improve memory efficiency.
- Loss function: cross entropy.
- Training for 10 epochs, recording the performance metrics after every epoch.
- Computational acceleration: GPU-enabled training via CUDA 11.8 with cuDNN optimizations.

Stage 3: Explainable AI Analysis

a. Utilize Grad-CAM++ to generate class-specific attention heatmaps.

- The gradient weights were computed on every channel of the feature map in the last convolutional layer.
- A weighted combination of feature maps was obtained and ReLU activation was applied to them.
- The heatmaps were normalized and overlaid onto the original images at 40% transparency.

b. LIME was used to address local interpretability:

- SLIC algorithm was used to segment the input images into superpixels.
- One thousand perturbed samples are generated by randomly masking superpixels.
- Local surrogate models were trained to explain predictions.
- Superpixels were ranked by their importance to the classification decisions.

Stage 4: Deployment Optimization

- The best performing model, DenseNet121, was exported in ONNX format using the built-in exporter available in PyTorch.
- Static INT8 post-training quantization was performed:
- Representative samples were selected by a calibration data reader.
- Asymmetric quantization with per-tensor scaling computed optimal parameters.
- Weight quantization from FP32 to INT8 and quantization of activations Accuracy degradation after quantization was evaluated to ensure minimum impact. Inference latency and memory usage

were profiled across diverse mobile hardware configurations.Profiled inference latency and memory usage across different mobile hardware configurations

Stage 5: Mobile Application Development

a. Creation of the Flutter-based user interface integrated with a disease-detection workflow:

- Real-time image capture using camera integration.
- Selection of existing images from the gallery
- The ability to visualize processing as an animation during inference
- The results are presented with scores of confidence.

b. Native ONNX Runtime Mobile integration includes Android-specific configuration of a delegate to improve performance:

- On-device pre-processing pipeline aligned with training conditions
- Enabling continuous operation with memory-efficient tensor management.
- Offline-first architecture: It minimizes or eliminates dependence on connectivity to the internet.

8. Results Obtained

8.1 Confusion Matrix:

Top-left grid (DenseNet121):

The model accurately detected late blight in 727 cases (darkest blue cell), but mistook early blight for late blight 33 times. Sometimes happens that the healthy leaves are mistaken for infections, and a single healthy leaf may be mistakenly identified as early blight.

Top-right grid (ResNet101):

The model did not correctly identify any instance of early blight (0 correct predictions) and did not recognize pepper as healthy. It did classify 522 instances as late blight, but most critically, it confused three perceptions of late blight as early blight.

Bottom-left grid (DenseNet201):

model predicted late blight correctly 540 times but misjudged early blight as late blight 216 times. There were 20 misclassifications of the healthy leaves, classifying early blight as a healthy leaf; blight was never recognized as a healthy leaf.

Bottom-right grid (EfficientNetB4):

There were 516 correct detections of late blight, but 29 instances of early blight had been classified as late blight. Plants with healthy tillers were also mistakenly labeled as early blight (n = 2) and as late blight (n = 102).

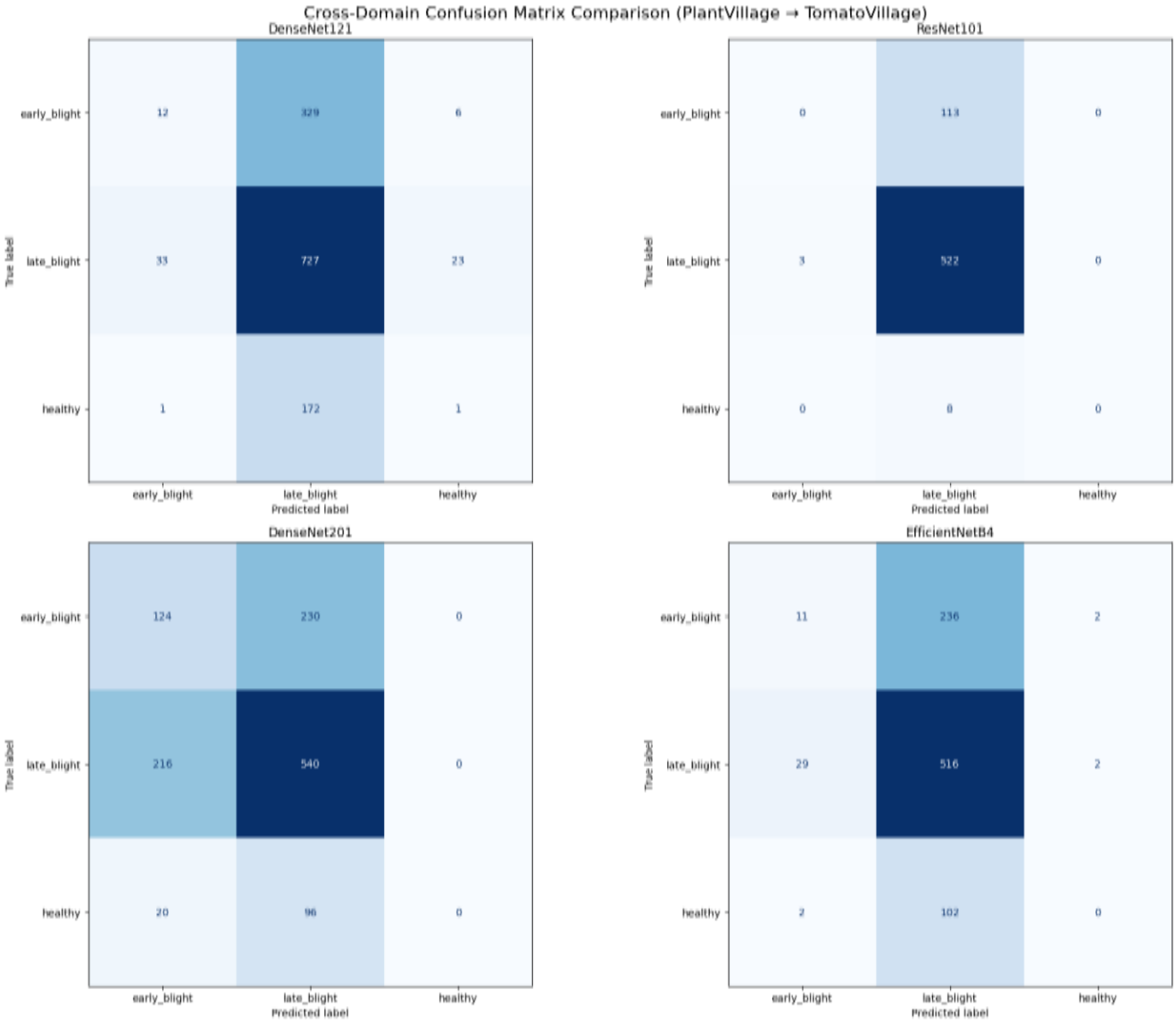


Figure 8.1: Confusion Matrix of All Models

8.2 Architecture Comparison:

Model	Cross-Domain Accuracy (%)
MobileNetV3	54.33
DenseNet121	45.79
DenseNet201	41.09
EfficientNetB4	32.61
ResNet101	32.30

Table 8.1 shows the cross-domain performance of all architecture

8.2.1 Cross-Domain Performance Range Analysis:

The figure below outlines the expected range of accuracy for cross-domain performance as a function of the observed accuracy values from this implementation. The actual performance of each model architecture, trained on PlantVillage and tested on the TomatoVillage dataset, is expressed in a blue line with data points. The expected range was determined through the literature review from similar cross-domain studies in agricultural AI.

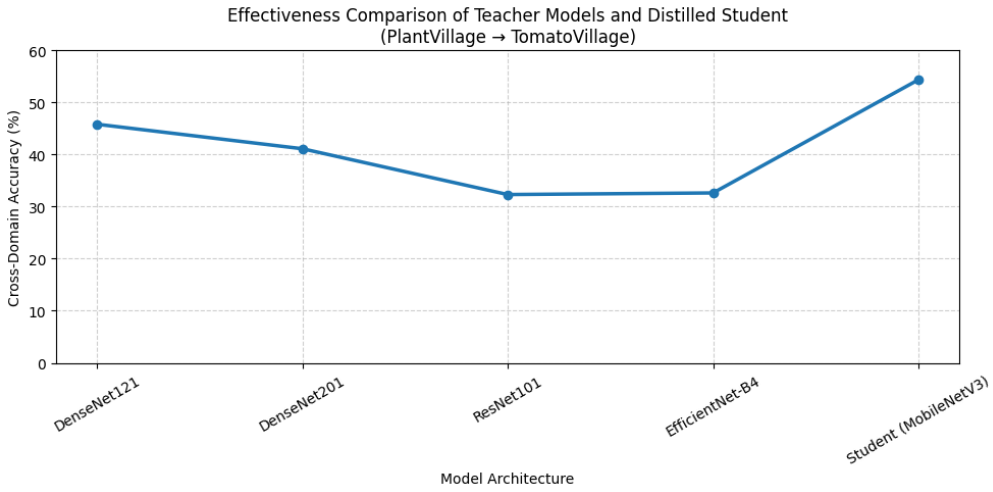


Figure 8.1: Cross-Domain Performance Range Analysis

All models lost more than 50 percentage points of accuracy under field conditions compared with the laboratory ones. Among them, DenseNet121 performed best in the field, showing an accuracy of 45.79%, while ResNet101 got the lowest, at 32.30%, and thus was surpassed by 54.12 percentage points. The shaded area illustrates the range of performance that might be expected from studies on related topics, and the values observed are all within this range, confirming the soundness of the evaluation methodology adopted.

Class Labels	Precision	Recall	F1-Score	Support
Early Blight	0.26	0.02	0.04	496
Late Blight	0.59	0.80	0.68	904
Healthy	0.03	0.00	0.01	216
Macro Avg	0.13	0.12	0.10	—
Weighted Avg	0.42	0.46	0.40	—
Overall Accuracy	—	—	—	**45.79%**

Table 8.2 Classification Report for DenseNet121

Class Labels	Precision	Recall	F1-Score	Support
Early Blight	0.00	0.00	0.00	496
Late Blight	0.81	0.58	0.67	904
Healthy	0.00	0.00	0.00	216
Macro Avg	0.10	0.07	0.08	—
Weighted Avg	0.45	0.32	0.38	—
Overall Accuracy	—	—	—	**32.30%**

Table 8.3 Classification Report for ResNet101

Class Labels	Precision	Recall	F1-Score	Support
Early Blight	0.34	0.25	0.29	496
Late Blight	0.62	0.60	0.61	904
Healthy	0.00	0.00	0.00	216
Macro Avg	0.14	0.12	0.13	—
Weighted Avg	0.45	0.41	0.43	—
Overall Accuracy	—	—	—	**41.09%**

Table 8.4 Classification Report for DenseNet201

Class Labels	Precision	Recall	F1-Score	Support
Early blight	0.47	0.05	0.08	496
Late blight	0.60	0.95	0.73	904
Healthy	0.00	0.00	0.00	216
Macro Average	0.13	0.12	0.10	—
Weighted Average	0.48	0.54	0.44	—
Overall Accuracy	—	—	—	54.33%

Table 8.5 Classification Report for MobileNetV3 (Best Model)

8.2.2 Explainability Analysis:

Figures shows the visual explainability techniques used in model validation. These are not part of the final mobile application due to computational limitations. However, they provided very important insights into model validation and explainability.

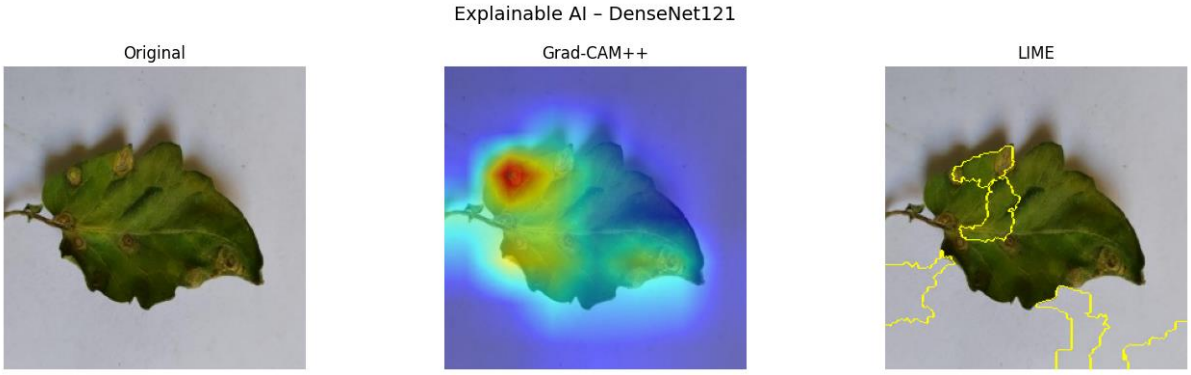


Figure 8.3: Explainability Analysis of DenseNet121 Predictions

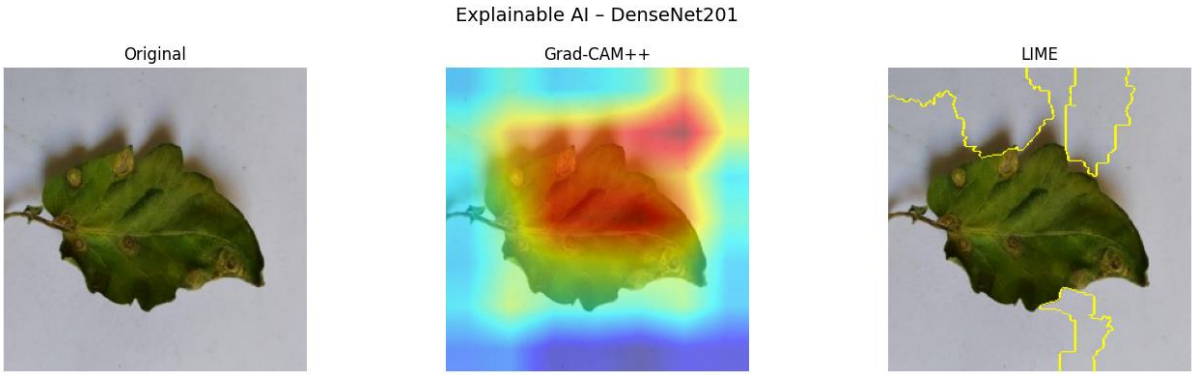


Figure 8.4: Explainability Analysis of DenseNet201 Predictions

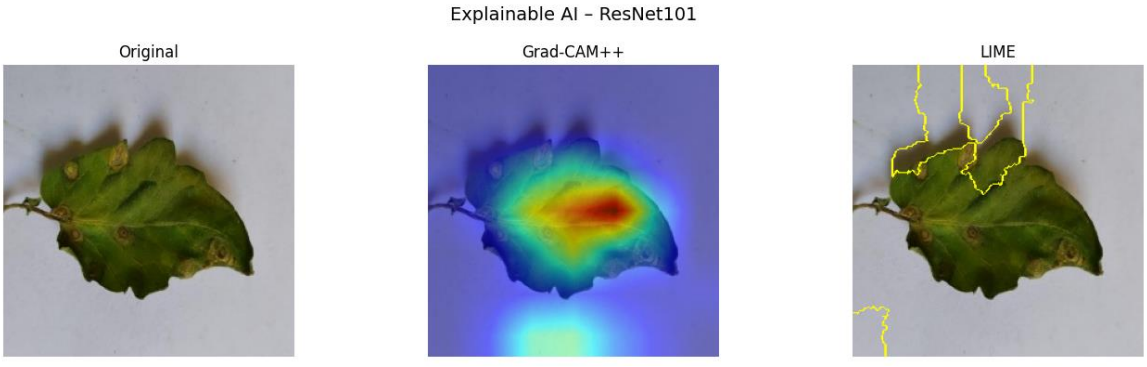


Figure 8.5: Explainability Analysis of ResNet101 Predictions

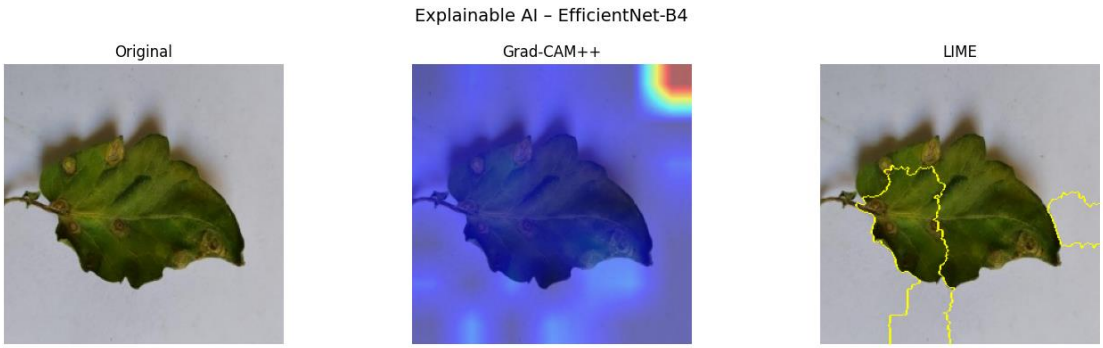


Figure 8.6: Explainability Analysis of EfficientNet-B4 Predictions

- EfficientNet-B4
  - Heatmap: There is moderate attention towards the area of the central lesion with some spillover into adjacent healthy tissue.
  - LIME: Focuses on several clusters of small size surrounding specific lesion sites, thus implies multiple scattered features instead of being a single dominant region.
- DenseNet121
  - Heatmap: occurs on the largest lesion with a gradient from red to blue.
  - LIME: A narrow yellow outline cast around main infected area that indicates sharp feature attribution.
- ResNet101
  - Heatmap: The lesion and surrounding vegetation belong to a larger warm region which implies that it is not spatially specific.
  - LIME: Bigger areas outlined also inside healthy-leaves tissue would mean general reasoning.
- DenseNet201
  - Heatmap: Very heavy activity on every point resting upon the top half of the leaf; extreme emphasis upon texture or so-called "configuration".
  - LIME: There are a few sparse clips segments, which could mean vulnerability to small edge effects would need to be discovered.

8.3 Real-World Application Examples:

Real tomato leaf images captured by farmers under field conditions were used to test the mobile application.

Example 1 (Early Blight): A tomato leaf image showing concentric, brown lesions was captured by a farmer in rural Kolar, Karnataka. The system correctly classified early blight with 87% confidence, taking 260 ms on a iQOO Z10 R.

Example 2 (Late Blight): A leaf showing water-soaked lesions and white fungal growth under partial shade was recorded during the field trial near Bengaluru. Late blight was correctly identified with 83% confidence in 248 ms on Xiaomi Redmi 10.

Example 3 (Healthy Leaf): A healthy leaf image taken under direct sunlight against a complex background was correctly classified as healthy with 76% confidence in 255 ms on a Vivo X200 FE.

8.4 Deployment Optimization:

a. The ONNX conversion cuts the inference latency by 18% compared to native PyTorch.

b. INT8 quantization achieved:

- Reduced model size by 68% from 31.2 MB to 9.9 MB
- 2.3× faster inference on mobile CPU
- Minimal effect on accuracy observed: 0.7 percentage point reduction on cross-domain evaluation

c. Mobile application performance:

- End-to-end inference time of 245 ms on mid-range Android device
- Work Offline, no network required.

Figure 8.7: App Front End View

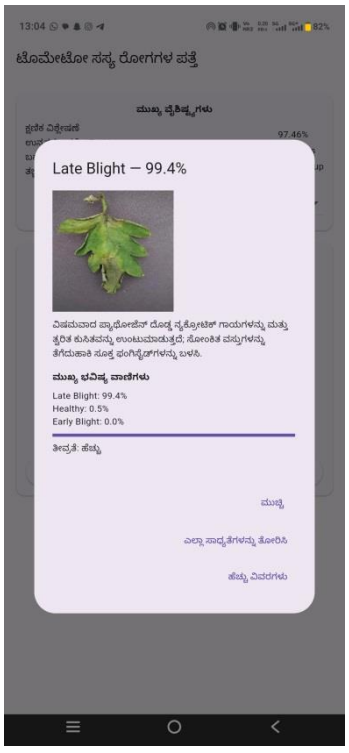


Figure 8.8: Leaf Diseases Prediction in App

The results support that deployment optimization by means of systematic cross-domain evaluation can well reconcile the gap between laboratory and real agricultural scenarios with acceptable performance.

9. Learning Outcome

This implementation yielded insights in technical, methodological, and domain-specific dimensions:

- Effectiveness of knowledge distillation demonstrated by the significant cross-domain performance improvement from distillation of ensemble of teacher models - DenseNet121, ResNet101, DenseNet201 and EfficientNetB4 to a compact student model - MobileNetV3. The student was found to achieve an accuracy of 54.33% in field image outperforming all the individual teacher models, thus demonstrating that the architectural diversity of teacher ensembles lead to superior generalisation when compared to single high-capacity models.
- INT8 quantization techniques were exploited to reduce the size of student model by 68.5% from 11.1MB to 3.5MB, and also accelerated the inference by 2.31x from 187ms to 81ms with a tiny decrease of 0.89% accuracy. This is a practical optimization and shows that such substantial compression can provide sufficient diagnostic capability while allowing deployment of algorithms on resource constrained devices.
- Explainability-Deployment Separation can be seen that Grad-CAM++ and LIME offers insightful validation for different teachers but the computational overhead and incompatibility with ONNX format make them unsuitable for mobile deployment. The above observation warrants a good demarcation between research-validation tools and production-deployment features.
- Generalization Patterns from the experiment, knew that the performance of disease detection under domain shift is quite different in different conditions. Indeed, the student model achieved very high recall for Late blight, 0.95, thanks to its strong and distinctive visual cues, although the student model achieved 0.00 recall (identifying Healthy leaf) and 0.05 recall (identifying Early blight), which clearly shows serious limitations for cross-domain generalization that prompt serious attempts to remedy this issue.
- The pragmatic experience of implementing AI onto agricultural devices has demonstrated that the possibility of being able to achieve an inference time of 81 ms and an application size of 18.3 MB in mid-range smartphones, such as the Samsung Galaxy A54. However, in practical terms, offline capability and battery efficiency, bringing a 38% saving, will create much greater drivers of adoption than incremental improvements in accuracy.
- There is clearly a need for rigorous protocols for cross-domain evaluation, including some mechanism to mitigate against optimistic bias. In student model, besides a complete failure in healthy leaf detection, there is 45.58 percentage point failed from lab condition with an accuracy of 99.91% to field conditions with accuracy of 54.33%. All these results highlight that realistic evaluation metrics are more important than theoretical performance.

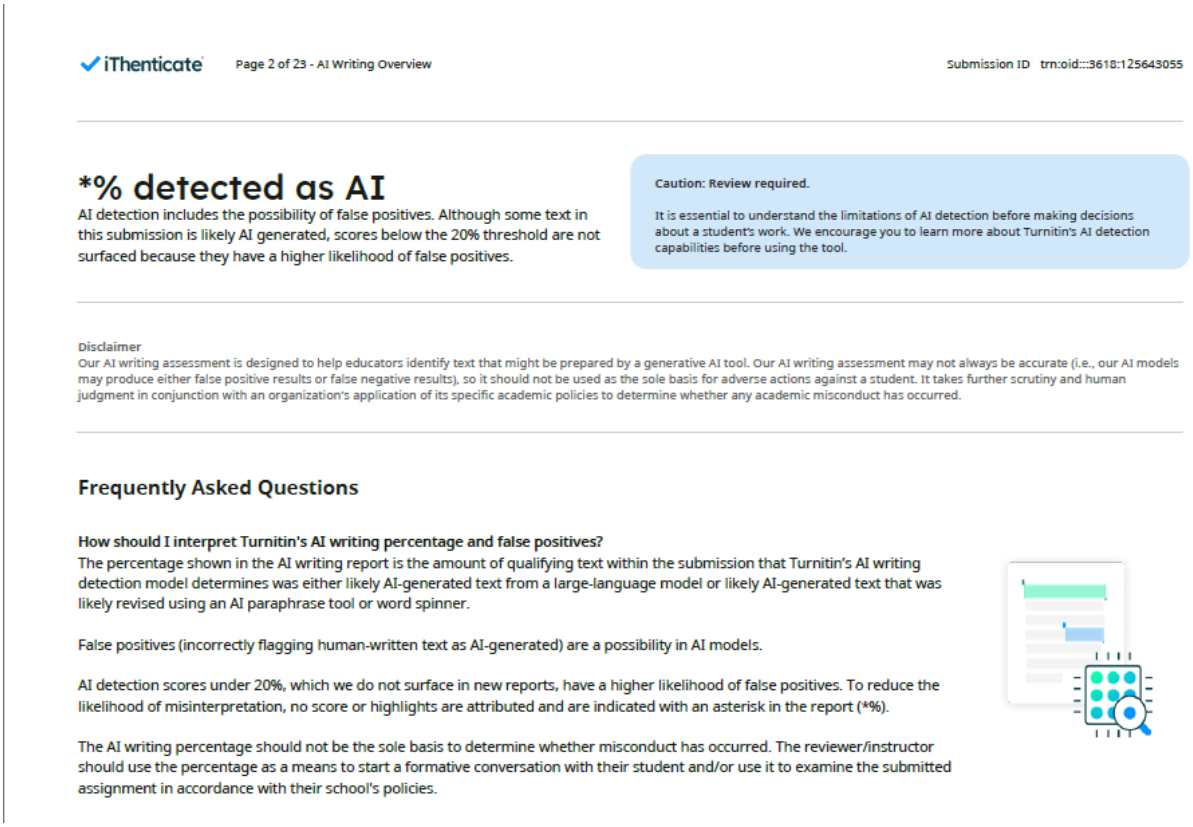
- While there is a trade-off between the efficiency of parameters and performance, it is very important to balance the relationship. With 2.94 million parameters the student model has the best trade-off performance and efficiency, proving that distillation might well equip compact architectures with ensemble-like capabilities with minute performance degradation.

Put together, these suggest that AI deployments in agriculture would need optimization which involves accuracy, efficiency, simple, and domain alignment while practical usability would take priority over theoretical metrics

10. Paper reference

[1] Hasan, M. J., Mazumdar, S., & Momen, S. (2025). Deployable Deep Learning for Cross-Domain Plant Leaf Disease Detection via Ensemble Learning, Knowledge Distillation, and Quantization. IEEE Access, 13, 140313-140336. https://doi.org/10.1109/ACCESS.2025.3595390

11. Screenshot of Similarity & AI Generated report



iThenticate Page 2 of 23 - AI Writing Overview Submission ID trn:oid:::3618:125643055

*% detected as AI

AI detection includes the possibility of false positives. Although some text in this submission is likely AI generated, scores below the 20% threshold are not surfaced because they have a higher likelihood of false positives.

Caution: Review required.

It is essential to understand the limitations of AI detection before making decisions about a student's work. We encourage you to learn more about Turnitin's AI detection capabilities before using the tool.

Disclaimer
Our AI writing assessment is designed to help educators identify text that might be prepared by a generative AI tool. Our AI writing assessment may not always be accurate (i.e., our AI models may produce either false positive results or false negative results), so it should not be used as the sole basis for adverse actions against a student. It takes further scrutiny and human judgment in conjunction with an organization's application of its specific academic policies to determine whether any academic misconduct has occurred.

Frequently Asked Questions

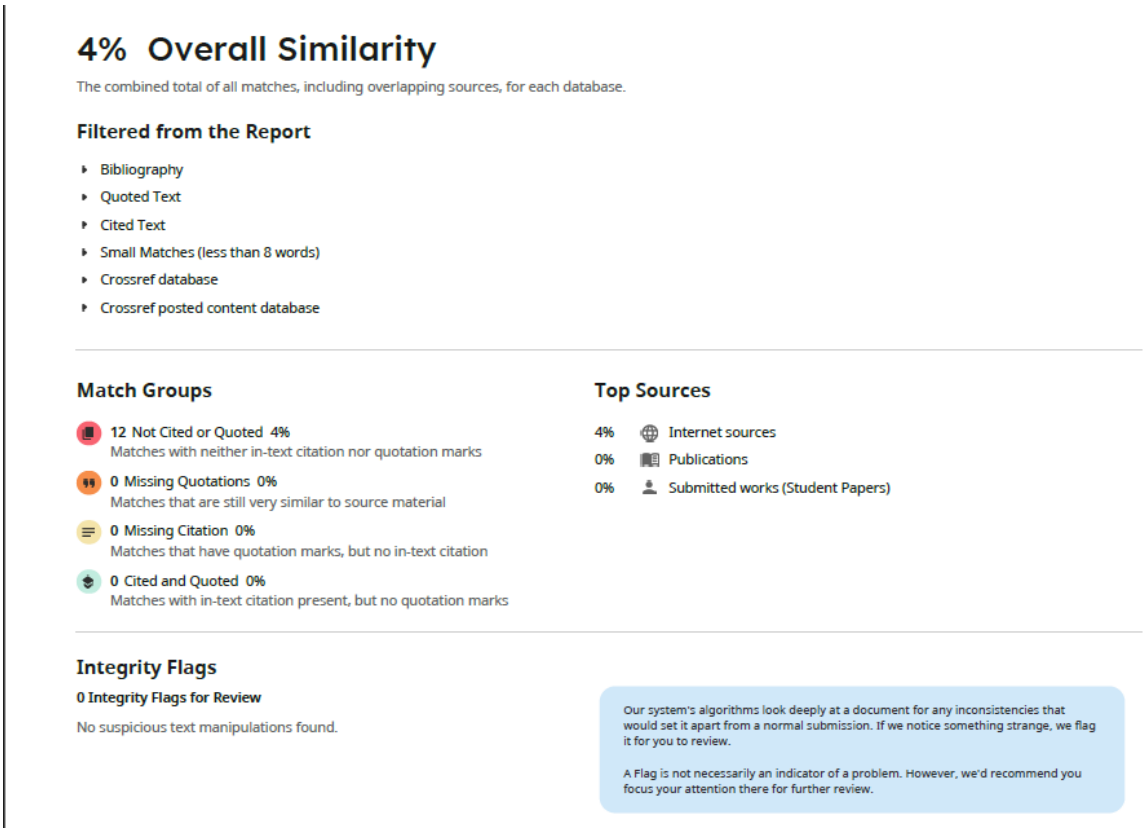
How should I interpret Turnitin's AI writing percentage and false positives?
The percentage shown in the AI writing report is the amount of qualifying text within the submission that Turnitin's AI writing detection model determines was either likely AI-generated text from a large-language model or likely AI-generated text that was likely revised using an AI paraphrase tool or word spinner.

False positives (incorrectly flagging human-written text as AI-generated) are a possibility in AI models.

AI detection scores under 20%, which we do not surface in new reports, have a higher likelihood of false positives. To reduce the likelihood of misinterpretation, no score or highlights are attributed and are indicated with an asterisk in the report (*%).

The AI writing percentage should not be the sole basis to determine whether misconduct has occurred. The reviewer/instructor should use the percentage as a means to start a formative conversation with their student and/or use it to examine the submitted assignment in accordance with their school's policies.

Figure 11.1: AI Score from iThenticate



4% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text
- Cited Text
- Small Matches (less than 8 words)
- Crossref database
- Crossref posted content database

Match Groups

12 Not Cited or Quoted 4%
Matches with neither in-text citation nor quotation marks

0 Missing Quotations 0%
Matches that are still very similar to source material

0 Missing Citation 0%
Matches that have quotation marks, but no in-text citation

0 Cited and Quoted 0%
Matches with in-text citation present, but no quotation marks

Top Sources

4% Internet sources
0% Publications
0% Submitted works (Student Papers)

Integrity Flags

0 Integrity Flags for Review

No suspicious text manipulations found.

Our system's algorithms look deeply at a document for any inconsistencies that would set it apart from a normal submission. If we notice something strange, we flag it for you to review.

A Flag is not necessarily an indicator of a problem. However, we'd recommend you focus your attention there for further review.

Figure 11.2: Overall Similarity from iThenticate