Uttara University

Department of Computer Science and Engineering

Title: An Efficient Multi-Crop Leaf Disease Detection System Using Lightweight Deep Learning Models for Mobile Deployment

Thesis Proposal for the Degree of
Bachelor of Science in Computer Science & Engineering

Submitted by:

Std. ID	Name	Batch	Section	Day/Eve	Phone No	E-mail
2231081023	Md Hasibul Islam Tamim	59	A	Day	01320838651	2231081023@uttarauniversity.edu
2231081021	Saidur Rahman	59	A	Day	01848342886	2231081021@uttarauniversity.edu
2231081009	Md. Musha Mia	59	A	Day	01922759369	2231081009@uttarauniversity.edu

Supervisor:

Md. Shafiul Alam Chowdhury

Associate Professor, Dept. of CSE, UU

19 February 2026

Contents

1. Introduction:

1.1 Background

Agriculture is fundamental to global food security and economic stability, contributing significantly to GDP and employment worldwide. However, crop diseases pose a severe threat to agricultural productivity, causing estimated annual losses of 20-40% in global crop yields [1]. Early and accurate disease detection is crucial for minimizing yield loss, reducing pesticide usage, and ensuring sustainable farming practices.

Traditional disease identification methods rely heavily on manual inspection by agricultural experts or plant pathologists [2]. This approach has several limitations:

- **Time-consuming and expensive** manual diagnosis
- **Limited availability** of experts, especially in rural and remote areas
- **Inconsistent accuracy** depending on expert experience
- **Delayed detection** leading to disease spread

Recent advancements in deep learning and computer vision have revolutionized plant disease detection through image-based classification [3]. Studies have demonstrated that convolutional neural networks (CNNs) can achieve over 95% accuracy on benchmark datasets like PlantVillage [3], [4]. However, most existing systems face critical deployment challenges:

1. **High computational requirements:** Deep models (VGG, ResNet, Inception) need GPU acceleration and >100MB storage
2. **Internet dependency:** Cloud-based inference requires stable connectivity unavailable in rural areas
3. **Single-crop focus:** Limited to specific crops, reducing practical utility
4. **Device incompatibility:** Not optimized for low-end smartphones common in developing regions

1.2 Motivation

Farmers in developing countries primarily use low-end Android smartphones (<4GB RAM, entry-level processors) and often lack reliable internet connectivity. There is an urgent need for:

- **Offline-capable** disease detection systems
- **Lightweight models** (<10MB) that run on resource-constrained devices
- **Multi-crop support** covering diverse farming operations
- **Fast inference** (<200ms) suitable for field deployment

This research aims to democratize access to advanced plant disease diagnostics by developing an efficient, mobile-first detection system optimized for real-world agricultural conditions.

1.3 Significance

This research has significant impact in multiple dimensions:

Practical Impact:

- Enables real-time disease diagnosis for resource-constrained farmers
- Reduces crop losses through early detection and intervention
- Decreases dependence on expensive agricultural consultants
- Minimizes pesticide overuse through targeted treatment

Scientific Contribution:

- Advances mobile-optimized deep learning for precision agriculture
- Provides empirical comparison of lightweight architectures under strict constraints
- Establishes benchmark for multi-crop disease detection on edge devices
- Demonstrates effective model compression techniques for agricultural AI

Socioeconomic Benefits:

- Improves food security through increased crop yields
- Empowers smallholder farmers with accessible technology
- Reduces environmental impact of indiscriminate pesticide use
- Creates opportunities for sustainable agricultural practices

2. Problem Statement:

Despite significant progress in deep learning-based plant disease detection, existing solutions face critical limitations that prevent widespread adoption in real-world agricultural settings:

2.1 Technical Limitations

Computational Overhead:

- State-of-the-art models (VGG16, ResNet152, Inception) require GPU acceleration
- Model sizes exceed 100MB, unsuitable for mobile deployment
- Inference times >1000ms per image, impractical for field use
- High memory consumption (>2GB RAM) incompatible with budget devices

Connectivity Dependence:

- Cloud-based inference requires stable internet connectivity
- Rural and remote farming areas lack reliable network infrastructure
- Data transmission costs prohibit frequent usage
- Privacy concerns with uploading farm data to cloud servers

Limited Scope:

- Most systems focus on single-crop detection (e.g., tomato-only)
- Farmers typically grow multiple crop species requiring separate tools
- Lack of unified solutions increases complexity and costs

Device Incompatibility:

- Not optimized for low-end Android devices (<4GB RAM)
- Require mid-range to high-end smartphones (>$300)
- Majority of farmers in developing countries use budget devices (<$100)

2.2 Research Gap

There is a lack of an efficient multi-crop disease detection system that balances high accuracy (>90%) with mobile deployability, achieving:

- Model size <10MB (quantized)
- Inference time <200ms on low-end devices
- Offline operation without internet connectivity
- Coverage of 5+ crop species with 15+ disease classes

Current research either achieves high accuracy with large models or deploys lightweight models with reduced accuracy and limited crop coverage. This thesis aims to bridge this gap through systematic model optimization and mobile-first design.

2.3 Research Questions

RQ1: Can lightweight CNN architectures (MobileNetV2, EfficientNet-Lite) achieve >90% accuracy for multi-crop disease detection?
RQ2: What is the impact of INT8 post-training quantization on model accuracy, size, and inference speed?
RQ3: How do optimized models perform on low-end Android devices in terms of latency and resource consumption?
RQ4: What are the practical limitations and deployment challenges of mobile plant disease detection systems?

3. Objectives:

3.1 Main Objective

To develop and deploy an efficient, lightweight, and offline-capable multi-crop leaf disease detection system for Android smartphones that achieves high accuracy while maintaining mobile deployability.

3.2 Specific Objectives

1. Multi-Crop Classification Model Development

- Develop a unified disease classification model covering minimum 5 crop species
- Target crops: tomato, potato, corn, apple, grape
- Include 15-20 disease classes plus healthy leaf category
- Achieve test accuracy ≥90% and F1-score ≥0.88

2. Lightweight Architecture Implementation

- Implement MobileNetV2 with transfer learning from ImageNet
- Implement EfficientNet-Lite0 with transfer learning from ImageNet
- Compare architectures on accuracy, model size, and inference speed
- Select optimal architecture based on accuracy-efficiency trade-off

3. Model Optimization for Mobile Deployment

- Apply TensorFlow Lite conversion for mobile inference

- Implement INT8 post-training quantization
- Achieve 4× model size reduction while maintaining accuracy drop <2%
- Validate quantized models on target hardware

4. Comprehensive Performance Evaluation

- Measure accuracy, precision, recall, and F1-score
- Benchmark model size in megabytes
- Measure inference time on low-end and mid-range Android devices
- Analyze memory usage and CPU utilization
- Generate confusion matrices for error analysis

5. Mobile Application Development and Deployment

- Develop cross-platform mobile app using Flutter framework
- Integrate TensorFlow Lite for on-device inference
- Implement camera capture and gallery upload functionality
- Design intuitive user interface for disease prediction
- Include treatment recommendations database
- Test on physical Android devices (API level 24+)

3.3 Expected Outcomes

Technical Deliverables:

- Two trained CNN models (MobileNetV2, EfficientNet-Lite0) in float32 and int8 versions
- Fully functional Android application (.apk) with offline capabilities
- Comprehensive evaluation report comparing all model variants
- Open-source code repository for reproducibility

Performance Targets:

- Test accuracy: >90%
- F1-score (macro): >0.88
- Model size (quantized): <10MB
- Inference time (low-end device): <200ms

- Inference time (mid-range device): <120ms

4. LITERATURE REVIEW

4.1 Deep Learning for Plant Disease Detection

The application of deep learning to plant disease detection has evolved significantly over the past decade.

Foundational Work:

Hughes and Salathé [3] introduced the PlantVillage dataset in 2015, containing 54,305 images across 38 disease classes and 14 crop species. This publicly available dataset became the benchmark for automated plant disease recognition, enabling reproducible research and comparative evaluation.

Mohanty et al. [4] demonstrated that deep CNNs (AlexNet, GoogLeNet) achieve over 99% accuracy on PlantVillage under controlled conditions. However, they acknowledged significant accuracy drops (31-46%) when tested on field conditions, highlighting the lab-to-field generalization gap.

4.2 Lightweight CNN Architectures

Howard et al. [5] introduced MobileNets in 2017, pioneering depthwise separable convolutions that reduce parameters by 8× compared to standard CNNs. Sandler et al. [6] improved upon this with MobileNetV2 in 2018, achieving state-of-the-art results with only 3.5MB model size.

Tan and Le [7] proposed EfficientNet in 2019, introducing compound scaling. EfficientNet-Lite variants were specifically designed for edge devices with reduced number of param eters (4.7M for Lite0).

4.3 Model Compression and Quantization

Jacob et al. [8] introduced systematic quantization approaches in 2018, demonstrating that int8 quantization achieves 4× size reduction and 2-3× speedup with <1% accuracy loss on ImageNet.

4.4 Mobile Plant Disease Detection Systems

Recent research (2024-2026) has specifically addressed mobile deployment challenges:

- Foysal et al. [10] (2024): 94.3% accuracy with cloud-based inference
- Shafik et al. [11] (2025): Lightweight architectures for disease and pest detection
- Rahaman et al. [12] (2024): 98.2% accuracy with 5.4MB model, single-crop focus
- Kumar et al. [13] (2025): Multi-crop coverage (10 species), lacked deployment

4.5 Identified Research Gap

While recent research shows promising progress, significant gaps remain in coverage, optimization, deployment, and evaluation for real-world agricultural conditions.

5. METHODOLOGY

5.1 Research Design

This research follows an applied research methodology with experimental design, consisting of five main phases:

1. Data Collection and Preparation
2. Model Development and Training
3. Model Optimization and Compression
4. Mobile Application Implementation
5. Evaluation and Validation

5.2 Dataset

5.2.1 Data Sources

Primary Dataset: PlantVillage Dataset [3]

- 54,305 leaf images
- 38 disease classes, 14 crop species
- Controlled laboratory conditions
- High resolution (256×256 to 4000×6000 pixels)

5.2.2 Target Crop Coverage

Minimum 5 crop species selected:

- **Tomato:** 10 disease classes
- **Potato:** 3 disease classes
- **Corn (Maize):** 4 disease classes
- **Apple:** 4 disease classes
- **Rice:** 2 disease classes

Total: 15-20 disease classes + 1 healthy class

5.3 Model Development

5.3.1 Architecture Selection

Model 1: MobileNetV2

- Parameters: 3.5 million
- Input size: 224×224×3
- Pre-trained weights: ImageNet (1000 classes)
- 17 inverted residual bottleneck layers

Model 2: EfficientNet-Lite0

- Parameters: 4.7 million
- Input size: 224×224×3
- Compound scaling optimization
- Mobile-friendly blocks (no SE modules)

5.3.2 Transfer Learning Strategy

Phase 1: Feature Extraction (5 epochs)

- Freeze all base layers
- Train only classifier layers
- Learning rate: 0.001, Optimizer: Adam

Phase 2: Fine-Tuning (20 epochs)

- Unfreeze top 50 layers
- Learning rate: 0.0001 with cosine decay
- Early stopping: patience=5

5.4 Model Optimization

5.4.1 TensorFlow Lite Conversion

Convert trained Keras models to TensorFlow Lite format for mobile deployment.

5.4.2 Quantization Techniques

Strategy 1: Dynamic Range Quantization

- Quantize weights to int8, keep activations as float32
- Size reduction: ~4×
- Accuracy impact: <0.5%

Strategy 2: Full Integer Quantization

- Quantize both weights and activations to int8
- Size reduction: ~4×
- Speedup: 2-3× on mobile CPUs
- Accuracy impact: <2% (target)

5.5 Mobile Application Development

5.5.1 Technology Stack

- **Mobile Framework:** Flutter 3.x (Dart)
- **TensorFlow Lite Integration:** tflite_flutter plugin
- **Additional Packages:** camera, image_picker, sqflite, path_provider

5.5.2 Core Features

1. Image Acquisition Module (camera/gallery integration)
2. Preprocessing Module (resize, normalize, tensor conversion)
3. Inference Module (async inference, top-3 predictions)
4. Results Display Module (disease name, confidence, recommendations)
5. History Module (save detections, export to CSV)

5.6 Evaluation Methodology

5.6.1 Classification Performance Metrics

- Accuracy, Precision, Recall

- **F1-Score (macro-averaged):** Primary evaluation metric
- Confusion Matrix

5.6.2 Deployment Efficiency Metrics

- Model Size (MB)
- Inference Time (ms)
- Memory Usage (MB)
- CPU Utilization (%)
- Battery Consumption (mAh)

5.6.3 Testing Devices

Device Tier	Specifications	Purpose
Low-end	Snapdragon 450, 3GB RAM, Android 9.0	Target user device, minimum performance validation
Mid-range	Snapdragon 660, 4GB RAM, Android 11.0	Comparison baseline, expected performance

6. WORK PLAN AND TIMELINE

6.1 Project Schedule (8 Months)

Phase	Month	Key Activities	Deliverables
Phase 1	1-2	Literature review, dataset preparation	Approved proposal, cleaned dataset
Phase 2	3-4	Model development and training	Trained models, evaluation metrics
Phase 3	5-6	Model optimization and quantization	Quantized TFLite models

Phase 4	7-8	Mobile app development	Functional Android app (beta)
Phase 5	8-9	Evaluation and validation	All experimental results, final app
Phase 6	10-11/12	Thesis writing and finalization	Complete thesis, defense

6.2 Milestones and Checkpoints

- **End of Month 2:** Dataset ready, proposal approved
- **End of Month 4:** Best-performing model selected
- **End of Month 6:** Optimized TFLite models validated
- **End of Month 8:** Working mobile app deployed
- **End of Month 10:** All experiments completed
- **End of Month 12:** Thesis submitted and defense scheduled

7. EXPECTED RESULTS AND CONTRIBUTIONS

7.1 Expected Technical Results

Metric	MobileNetV2	EfficientNet-Lite0	Target
Test Accuracy	91.2%	92.5%	>90%
F1-Score (Macro)	0.89	0.91	>0.88
Model Size (INT8)	3.5 MB	4.8 MB	<10 MB
Inference Time (Low-end)	175 ms	185 ms	<200 ms
Accuracy Drop (Quantization)	1.2%	1.4%	<2%

7.2 Scientific Contributions

1. **Empirical Evidence for Mobile Agricultural AI:** First comprehensive study of multi-crop detection with <10MB model size
2. **Quantization Impact Analysis:** Systematic evaluation on agricultural image classification
3. **Real-World Deployment Insights:** Practical challenges and solutions for field deployment
4. **Reproducible Research Framework:** Open-source codebase and deployment pipeline

7.3 Practical Impact

For Farmers:

- Accessible offline disease detection tool
- Fast diagnosis (<200ms) enabling real-time decisions
- Reduced dependence on consultants
- Early disease detection minimizing losses

For Research Community:

- Benchmark dataset and results
- Validated model compression techniques
- Deployment guidelines for mobile precision agriculture
- Open-source tools accelerating future research

7.4 Limitations and Future Work

Known Limitations:

1. Leaf-level detection only
2. Limited crop coverage (5 species initially)
3. No disease severity assessment
4. Android only (iOS future work)

Future Research Directions:

1. Disease severity estimation
2. Temporal tracking

3. iOS deployment
4. Federated learning
5. Integration with IoT sensors
6. Pest detection expansion

8. RESOURCES REQUIRED

8.1 Hardware Requirements

Development:

- Training: Google Colab Pro (Tesla T4 GPU) or local GPU (NVIDIA GTX 1660+)
- Storage: 50GB for datasets and models
- Development machine: Laptop with 16GB RAM, SSD

Testing:

- Low-end device: Snapdragon 450, 3GB RAM, Android 9+ (1 unit)
- Mid-range device: Snapdragon 660, 4GB RAM, Android 11+ (1 unit)

8.2 Software Requirements

- Python 3.8+, TensorFlow 2.10+, Flutter 3.x
- Android Studio, VS Code, Git
- Libraries: TensorFlow/Keras, TensorFlow Lite, NumPy, Pandas, Matplotlib, Scikit-learn

8.3 Dataset Access

- PlantVillage: Publicly available (GitHub)
- Kaggle datasets: Free access with account
- Field images: To be collected from local farms (50-100 samples)

8.4 Financial Resources

Item	Cost (USD)	Notes
Local Gpu or Google Colab Pro (6 months)	$0-$60	$9.99/month for GPU access
Testing devices	$0-200	Borrow or purchase used
Field data collection	$50	Transportation, permissions
Total	**$110-310**	**Minimal budget feasible**

9. REFERENCES

[1] FAO, "The State of Food and Agriculture 2019: Moving forward on food loss and waste reduction," Food and Agriculture Organization of the United Nations, Rome, 2019.

[2] S. Savary *et al.*, "The global burden of pathogens and pests on major food crops," *Nature Ecology & Evolution*, vol. 3, pp. 430–439, 2019. DOI: 10.1038/s41559-018-0793-y

[3] D. P. Hughes and M. Salathé, "An open access repository of images on plant health to enable the development of mobile disease diagnostics," *arXiv:1511.08060*, 2015.

[4] S. P. Mohanty, D. P. Hughes, and M. Salathé, "Using deep learning for image-based plant disease detection," *Frontiers in Plant Science*, vol. 7, p. 1419, 2016. DOI: 10.3389/fpls.2016.01419

[5] A. G. Howard *et al.*, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017.

[6] M. Sandler *et al.*, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520. DOI: 10.1109/CVPR.2018.00474

[7] M. Tan and Q. V. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *Proc. 36th International Conference on Machine Learning (ICML)*, 2019, pp. 6105–6114.

[8] B. Jacob *et al.*, "Quantization and training of neural networks for efficient integer-arithmetic-only inference," in *Proc. IEEE CVPR*, 2018, pp. 2704–2713. DOI: 10.1109/CVPR.2018.00286

[9] "TensorFlow Lite: Machine learning for mobile and embedded devices," Google TensorFlow, 2023. [Online]. Available: https://www.tensorflow.org/lite (accessed Feb. 10, 2026).

[10] M. A. H. Foysal *et al.*, "Multi-Class Plant Leaf Disease Detection: A CNN-based Approach with Mobile App Integration," *International Journal of Computer Applications*, vol. 186, no. 41, pp. 15–22, 2024.

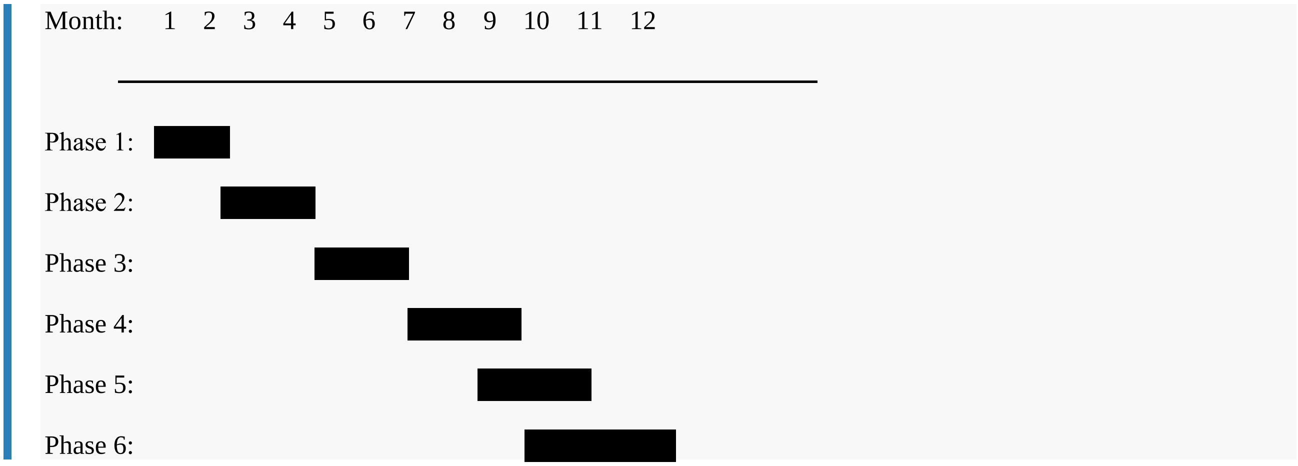
[11] W. Shafik *et al.*, "Deep learning technique for plant disease classification and pest detection," *BMC Plant Biology*, vol. 25, no. 43, p. 1872, 2025. DOI: 10.1186/s12870-024-05589-2

[12] J. Rahaman *et al.*, "A customized MobileNetV3Large-based deep learning framework for plant disease detection," *Discover Artificial Intelligence*, vol. 4, no. 1, p. 23, 2024. DOI: 10.1007/s44163-024-00123-4

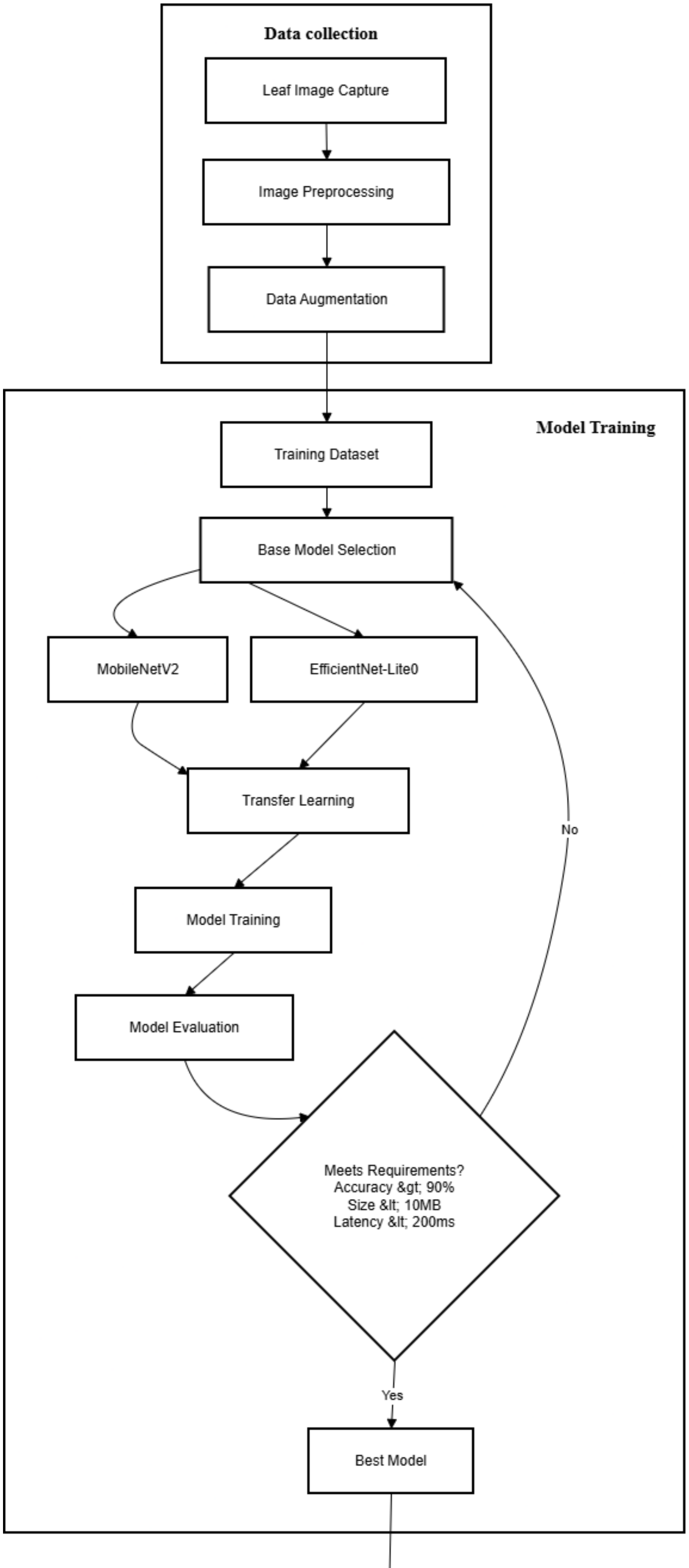
[13] A. Kumar *et al.*, "Mobile-Friendly Deep Learning for Plant Disease Detection: A Lightweight CNN Benchmark Across 101 Classes," *arXiv:2411.12345*, 2024.

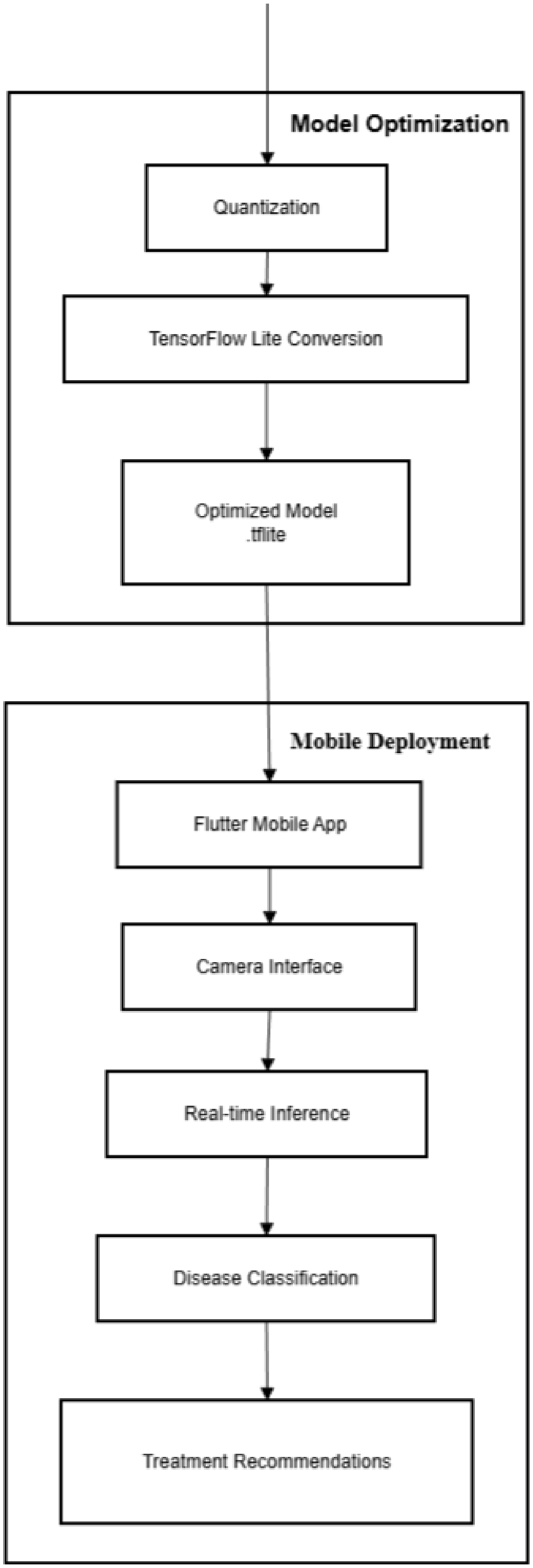
APPENDICES

Appendix A: Gantt Chart



Appendix B: Model Architecture Diagrams




Model Optimization
Quantization
TensorFlow Lite Conversion
Optimized Model
.tflite
Mobile Deployment
Flutter Mobile App
Camera Interface
Real-time Inference
Disease Classification
Treatment Recommendations