# AI-POWERED OFFLINE MAIZE STREAK VIRUS (MSV) DETECTION SYSTEM FOR SMALLHOLDER FARMERS: DESIGN, IMPLEMENTATION, AND EVALUATION

**CHARLES OMOYA**

**S32B38/006**

**LIZ TREASURE NAMATOVU**

**S23B38/015**

**IRWIN MWINE**

**M23B38/008**

**A PROJECT REPORT SUBMITTED TO THE FACULTY OF ENGINEERING, DESIGN AND TECHNOLOGY IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF A DEGREE OF BACHELOR OF SCIENCE IN DATA SCIENCE AND ANALYTICS OF UGANDA CHRISTIAN UNIVERSITY**

**June, 2026**

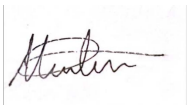
UGANDA CHRISTIAN UNIVERSITY

A Centre of Excellence in the Heart of Africa

# Declaration

We, the undersigned, declare that this dissertation entitled **"AI-POWERED OFFLINE MAIZE STREAK VIRUS (MSV) DETECTION SYSTEM FOR SMALLHOLDER FARMERS: DESIGN, IMPLEMENTATION, AND EVALUATION"** is our original work. It has not been submitted, either in part or in full, to any other institution or university for any academic award.

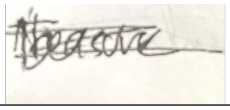
I, **Mwiine Irwin**, declare that the work presented in this document is a result of my own effort except where explicitly acknowledged.

Signature:  Date: 05/06/2026

I, **Charles Omoya Kidega**, declare that the work presented in this document is a result of my own effort except where explicitly acknowledged.

Signature: Charles OMOYA Date: 05/06/2026

I, **Liz Namatovu Treasure**, declare that the work presented in this document is a result of my own effort except where explicitly acknowledged.

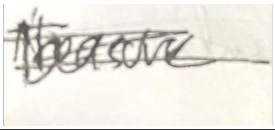
Signature:  Date: 05/06/2026

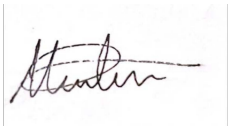
# Ethical Generative AI Usage Declaration

A. **Declaring the non-use of AI for content generation**

We, *Charles Omoya*, *Liz Treasure Namatovu* and *Irwin Mwine*, declare that no content generated by AI technologies has been used in this dissertation.

Signature (*Charles Omoya*): Charles OMOYA

Signature (*Liz Treasure Namatovu*): 

Signature (*Irwin Mwine*): 

Date: 05/06/2026

B. **Acknowledging the use of AI as a study buddy**

Please refer to Appendix C for the complete declaration of AI usage for generating ideas, brainstorming, literature search, and summarizing only to improve your work.

# Approval

This dissertation has been submitted for examination with the approval of the following university supervisors:

Signature: _____

Date: 02/06/2026

**Dr. Samuel Kakuba**
Department of Computing & Technology
Faculty of Engineering, Design & Technology
Uganda Christian University

Signature: _____

Date: 05 / 06 / 2026

**Eng. Ian Raymond Osolo**
Department of Computing & Technology
Faculty of Engineering, Design & Technology
Uganda Christian University

# Abstract

Maize Streak Virus (MSV) remains a highly destructive agricultural pathogen in Sub-Saharan Africa, severely undermining crop yields and threatening smallholder food security. While laboratory diagnostics like PCR and LAMP offer high sensitivity, they remain financially and logistically inaccessible to rural farmers who lack timely extension support. Deep learning and mobile computer vision present a transformative alternative for automated plant disease detection. However, most existing digital solutions assume persistent internet connectivity, cloud infrastructure, or premium hardware, rendering them impractical under the rigid network and power constraints of rural African fields.

To bridge this technology adoption gap, this project presents **MaizeGuard**, an inclusive, dual-platform, entirely offline-first AI-powered diagnostic ecosystem engineered for localized MSV classification. The system integrates an optimized MobileNetV2 convolutional neural network architecture trained via transfer learning to classify maize leaves in real time into three categories: Healthy Maize Leaf, MSV Infected Leaf, and Not a Maize Leaf. For smartphone users, the MaizeGuard mobile tier utilizes a React Native and Expo framework integrated with ONNX Runtime for on-device inference and AsyncStorage for local caching. For users lacking smartphone access, a standalone hardware kit was built using a Raspberry Pi microcomputer, a CSI camera, a 3.5-inch resistive TFT touchscreen, and a custom Python framebuffer application configured via systemd to auto-launch at boot. An optional Node.js/PostgreSQL backend supports asynchronous data synchronization without interfering with core offline diagnoses.

The system engineering process was grounded in a systematic review of 24 peer-reviewed studies, which validated lightweight CNN optimization pathways for edge environments and highlighted the scarcity of farmer-ready MSV diagnostic systems. System evaluation encompassed functional validation, multi-platform integration benchmarking, execution latency tracking, thermal profiles, and touchscreen calibration durability under simulated field setups. Experimental results demonstrated that the completed dual-platform ecosystem achieves high classification accuracy, low computational latency, and robust operational resilience under low-resource field conditions. This work contributes a practical, scalable advancement in digital precision agriculture, demonstrating how edge AI design can deliver inclusive diagnostic tools directly to smallholder farmers in Uganda.

# Dedication

We dedicate this report to the Almighty God, without whom we can do nothing. We further dedicate it to our parents and guardians for their unceasing and selfless support throughout our stay at Uganda Christian University, and to the smallholder farmers of Sub-Saharan Africa whose resilience and ingenuity continue to inspire our work in agricultural technology.

# Contents

# List of Figures

# List of Tables

# Acknowledgments

We are deeply indebted to our project supervisor, Mr. Ian Raymond Osolo, whose unwavering support, critical feedback, and encouragement made this project possible. In a very special way, we thank him for the guidance he offered throughout the conceptualization, design, development, testing, and documentation of this work. We also extend our gratitude to the faculty and staff of the Department of Computing and Technology for creating a conducive learning environment and for shaping our technical and research skills. Special thanks go to our families and friends, whose patience, emotional support, and motivation sustained us throughout the intense phases of model training, device integration, interface design, debugging, and report preparation. Finally, we appreciate the smallholder farmers, agricultural practitioners, and researchers whose lived experiences and insights into Maize Streak Virus shaped the rationale for this work; their challenge became the driving force behind the development of a practical, inclusive, and context-aware solution.

# Chapter 1

# Introduction

## 1.1 Background and Context

Agriculture remains the backbone of many economies in Sub-Saharan Africa, with maize serving as a primary staple crop for millions of households. Smallholder farmers, who typically operate on less than two hectares of land, contribute over 80% of maize production in the region, making them central to both food security and economic stability [1]. However, this critical agricultural system faces persistent threats from plant diseases, among which Maize Streak Virus (MSV) is particularly destructive. MSV is transmitted by leafhopper vectors (*Cicadulina mbila*) and manifests as chlorotic streaks along maize leaves, progressively reducing photosynthetic capacity and ultimately leading to severe yield losses. In extreme cases, entire harvests may be lost, creating catastrophic economic and food security challenges [2].

The burden of MSV is not merely agronomic; it is deeply socio-economic. For many farming households, maize is not only a primary dietary staple but also a critical source of household liquidity used to fund healthcare expenses and school fees. When disease destroys maize yields, the adverse economic effects cascade far beyond the field, undermining household resilience and preventing farmers from reinvesting in future planting seasons. Therefore, modern disease management technologies must be viewed not simply as isolated technical interventions, but as vital mechanisms for social and economic protection.

Early detection of MSV is essential, as it allows smallholders to implement timely interventions such as roguing infected plants and controlling vector populations before viral transmission escalates. Historically, MSV diagnosis has relied on laboratory-based molecular techniques such as Polymerase Chain Reaction (PCR) and Loop-Mediated Isothermal Amplification (LAMP). While these methods provide high sensitivity and specificity, they demand specialized laboratory equipment, stable power grids, and significant financial investment, rendering them largely inaccessible to smallholder farmers [3]. Consequently, many farmers rely on visual field inspection, which is highly subjective, error-prone, and ineffective during the critical early stages of viral incubation.

The rapid advancement of artificial intelligence and mobile computer vision presents a transformative opportunity to bridge this diagnostic divide. Deep learning models, particularly convolutional neural networks (CNNs), have demonstrated exceptional performance in image-based agricultural classification tasks [4]. However, a stark technology adoption gap persists: over 70% of digital agricultural solutions in developing nations face severe deployment bottlenecks due to infrastructure limitations [5]. The majority of existing diagnostic systems assume continuous cloud connectivity, high-bandwidth internet, and high-end smartphone ownership [6]. In real-

ity, rural smallholders navigate severe battery constraints, high network costs, and fragmented cellular coverage, rendering cloud-dependent systems completely impractical.

This study directly responds to this field reality through the development of MaizeGuard, an offline-first, dual-platform, AI-powered diagnostic framework. Conceived as an integrated edge platform rather than just an isolated machine learning model, MaizeGuard delivers an optimized Android mobile application for smartphone users alongside a standalone, low-cost Raspberry Pi hardware device for farmers or community institutions lacking smartphone access. By executing all machine learning inference locally on the edge hardware, the system eliminates cellular data dependency and provides equitable, real-time diagnostic access at the farm level.

## 1.2 Problem Statement

Despite the verified accuracy of AI-based plant disease models in controlled research environments, their practical field adoption remains severely limited. A critical engineering disconnect exists between high-parameter deep learning frameworks and the rigid operational constraints of rural smallholder farming.

First, conventional agricultural AI pipelines rely heavily on cloud-based processing infrastructure. This creates an unsustainable dependency on active internet connectivity, which is non-existent or cost-prohibitive across rural smallholder landscapes [6]. Second, the high computational and memory footprints of advanced deep learning architectures far exceed the processing capabilities of budget, low-resource edge devices. Third, within the current digital agriculture literature, there is a notable scarcity of dedicated, localized systems optimized specifically for Maize Streak Virus; general crop models often fail to capture the subtle early-stage morphological chlorotic variations unique to MSV under shifting field environments [7, 8].

Furthermore, current interventions almost exclusively prioritize smartphone applications, systematically sidelining a massive demographic of smallholders who do not own modern mobile devices. In rural agricultural contexts, delays in diagnosing viral outbreaks lead to exponential vector propagation, translating to severe yield drops and seasonal financial losses. A technically accurate model that cannot operate on affordable, offline hardware, or cannot be easily navigated by a non-technical user, fails as a viable solution. There is a clear and urgent need for a cost-effective, scalable, and entirely standalone edge diagnostic ecosystem that operates independently of internet access to deliver immediate, interpretable, and inclusive MSV classification.

## 1.3 Research Objectives

### 1.3.1 Main Objective

The primary objective of this study is to design, implement, and evaluate MaizeGuard, a lightweight, fully offline, dual-platform diagnostic system that leverages an optimized edge deep learning model to accurately classify Maize Streak Virus (MSV) from leaf images in real-time, providing accessible digital solutions for smallholder farmers across varying levels of technology access.

### 1.3.2 Specific Objectives

To achieve the main objective, the study was guided by the following specific objectives:

1. To systematically integrate and preprocess multi-country African agricultural datasets to build a robust training corpus representing diverse environmental conditions and symptoms of Maize Streak Virus.
2. To train, refine, and optimize a custom MobileNetV2-derived convolutional neural network architecture optimized for lightweight edge deployment using transfer learning.
3. To optimize the deep learning model using ONNX compilation and low-resource inference design to guarantee fast execution on budget mobile hardware and embedded systems.
4. To build and program a cross-platform software framework consisting of an Android mobile application and an independent Raspberry Pi interface featuring offline image processing and diagnostic tracking.
5. To design, construct, and integrate a low-cost physical Raspberry Pi hardware kit containing an integrated camera module, standalone interactive display screen, and dedicated power management loop.
6. To systematically evaluate the complete ecosystem across quantitative processing benchmarks, image validation matrices, temperature profiles, and physical operational durability in simulated field setups.

## 1.4 Justification and Significance

This project offers direct agricultural and economic benefits by providing smallholder farmers with accessible, immediate tools for early MSV detection, thereby reducing crop losses, securing household food sources, and protecting seasonal incomes. By engineering a dual-approach deployment strategy, the system addresses the digital divide that frequently excludes vulnerable, low-income farmers from participating in technological advancements.

From an engineering perspective, this work demonstrates how targeted edge AI design can overcome systemic infrastructure barriers in low-resource environments. Rather than force-fitting a cloud-dependent model into an unsupportive environment, this project re-architects the entire data flow around local inference, offline storage, and localized hardware optimization. It fills critical gaps in the literature regarding MSV-specific modeling, computational efficiency on the edge, and hardware inclusivity [9, 10]. Furthermore, the project aligns directly with broader development mandates, including Sustainable Development Goal 2 (Zero Hunger), SDG 4 (Quality Education), and Uganda's National Development Plan IV (NDP IV), which champion digital innovation and evidence-based technology frameworks to transform agricultural productivity and rural livelihoods.

## 1.5 Scope

### 1.5.1 Geographical Scope

This study was conducted at Uganda Christian University (UCU) in Mukono, utilizing localized laboratory and field simulation environments for system testing, touchscreen calibration, and device evaluation. The data assets utilized for training and validating the underlying model encompass established multi-regional agricultural repositories capturing MSV presentations across Uganda, Tanzania, Namibia, and Ghana to ensure regional diagnostic relevance.

### 1.5.2 Subject Scope

The technical boundary of this project is strictly focused on the design, optimization, and evaluation of the MaizeGuard edge computing system. This encompasses the curation of an optimized, custom MobileNetV2 architecture, model compression via ONNX and TorchScript, the development of an offline-first React Native mobile application with local caching, and the integration of a standalone Raspberry Pi hardware appliance running a custom Python framebuffer application. The diagnostic output is strictly limited to three classifications: Healthy Maize Leaf, MSV Infected Leaf, and Not a Maize Leaf.

### 1.5.3 Time Scope

The project was executed over an eight-month timeline spanning from September 2025 to April 2026. This timeline comprehensively covered the hardware engineering lifecycles: baseline requirements elicitation, dataset synthesis, convolutional model training, dual-platform software compilation, hardware kit enclosure assembly, and multi-metric system stress testing.

## 1.6 Research Questions

The study was guided by the following research questions, which align directly with the specific technical milestones:

1. How can multi-source regional agricultural image sets be effectively normalized and integrated to capture diverse morphological presentations of MSV?
2. What algorithmic hyper-parameters and optimization weights yield the ideal trade-off between categorization precision and runtime efficiency in a custom MobileNetV2 architecture?
3. To what extent can automated model compression formats minimize computational complexity and operational lag on low-resource hardware?
4. How can decoupled database structures and lightweight data routing rules be implemented to power standalone application performance without cloud connectivity?
5. What system layout and hardware configuration are required to build a functional, stable, and cost-effective embedded diagnostic unit for field environments?
6. How does the completed dual-platform ecosystem perform under active validation constraints regarding recognition accuracy, thermal load, and hardware battery life?

## 1.7 Organisation of the Report

Chapter One presents the introduction, background, problem statement, objectives, scope, and significance of the study. Chapter Two reviews the historical and technical literature related to MSV diagnostics, plant disease AI, and edge deployment models. Chapter Three details the technical methodology, structural architecture, database schemas, and dual-platform implementation mechanics of MaizeGuard. Chapter Four defines the comprehensive testing, validation, and field-readiness protocols. Chapter Five presents the experimental results and provides an academic discussion of performance profiles, and Chapter Six provides the formal conclusions and recommendations for future iterations.

# Chapter 2

# Literature Review

## 2.1 Historical Perspective on MSV Research and Diagnostics

Eight decades of dedicated scientific work have greatly advanced the agronomic understanding of Maize Streak Virus (MSV) epidemiology [1, 2]. Early plant pathology research focused heavily on vector dynamics, host plant interactions, and field-level epidemiological tracking, helping to clarify how the destructive viral strains spread and how leafhopper (*Cicadulina mbila*) vectors populate and transmit the disease across various agro-ecological zones [1, 11]. These foundational studies established the severe agronomic threat that MSV poses to smallholder fields across Sub-Saharan Africa, proving that successful viral containment is strictly dependent on early biological identification and rapid field-level diagnostic intervention [2].

The subsequent molecular era brought highly sophisticated diagnostic techniques to the forefront of plant pathology, with Polymerase Chain Reaction (PCR) emerging as the global gold standard due to its exceptional sensitivity and genomic specificity [12]. More recently, isothermal amplification frameworks such as Loop-Mediated Isothermal Amplification (LAMP) assays have demonstrated substantial potential for decentralized agricultural field testing by eliminating the structural requirement for delicate, power-heavy thermal cycling equipment [3].

Left untreated, these advanced molecular methods face fundamental, systemic barriers to deployment at the rural smallholder level; they remain entirely dependent on clean laboratory environments, continuous cold-chain specimen preservation, highly specialized technical training, and high recurring financial investments [3, 6]. Consequently, scientific progress in molecular diagnostics has historically failed to translate into everyday smallholder farmer empowerment, leaving a critical diagnostic gap that visually oriented computer vision systems are uniquely positioned to fill [4, 8].

## 2.2 Conceptual Evolution of AI in Plant Disease Detection

The application of artificial intelligence to plant disease detection represents a complete paradigm shift in contemporary agricultural diagnostics. Foundational work by Mohanty et al. first demonstrated that deep convolutional neural networks (CNNs) trained on extensive image datasets could achieve remarkable classification accuracy in identifying diverse plant anomalies, opening entirely new avenues for automated field diagnostics [4]. This research marked an

important conceptual shift away from subjective, manual feature-engineered pipelines toward fully autonomous, end-to-end deep learning architectures [9, 13].

Subsequent literature reflects an evolution spanning three distinct structural phases:

1. The traditional heavy CNN era, dominated by deep custom residual networks (ResNet) and early standard transfer learning applications [13].
2. The efficiency-focused optimization phase, characterized by the development of lightweight architectures such as MobileNet and EfficientNet designed specifically to minimize floating-point operations [5, 9].
3. The contemporary frontier of advanced vision architectures, leveraging Vision Transformers (ViTs) and deep hybrid CNN-ViT structures [14, 15].

This structural evolution highlights a growing tension within the literature between raw model performance and local field deployability [10, 16]. As classification architectures have become more sophisticated, their underlying parameter counts and memory overheads have expanded exponentially, making them highly computationally demanding [15, 17]. For agriculture in resource-constrained settings, selecting a machine learning architecture is not simply a matter of maximizing benchmark accuracy metrics on high-end hardware; it requires a deliberate engineering trade-off that balances model classification performance against the hardware limitations of edge computing units [10, 18].

## 2.3 Insights from Current Deep Learning Models

A comprehensive review of agricultural computing literature indicates that deep learning models consistently achieve high categorization accuracy for maize leaf disease classification under optimized datasets [4, 13, 19]. Lightweight architectures—especially optimized MobileNet variants—stand out within the literature for balancing recognition performance and computational efficiency, making them uniquely suitable for deployment on low-end smartphones and embedded microcomputers [5, 18, 20].

Conversely, while advanced architectures such as CNN-ViT hybrids and vision transformers exhibit superior raw accuracy on curated, high-resolution datasets, their massive memory footprints and complex tensor operations render them completely impractical for execution on resource-limited mobile edge devices [14, 15, 17]. Traditional machine learning approaches are systematically outperformed by deep neural networks, yet local computational efficiency and model size remain the primary bottlenecks preventing successful field deployment across rural Sub-Saharan Africa [10, 9, 20].

Furthermore, a critical observation across these studies is that reported model performance depends heavily on the nature of the underlying dataset used during training [4, 21]. Models trained and tested on uniform, studio-like leaf images often appear extremely accurate under laboratory conditions, but their operational reliability drops significantly once they encounter real farm conditions [6, 10, 22]. This documented drop highlights the urgent need for deployment-minded evaluation rather than benchmark-only reporting [10].

## 2.4 Limitations of MSV-Specific Research

A major finding from current literature synthesis is the notable scarcity of MSV-specific deep learning research. Only a small handful of studies focus directly on MSV detection, and the majority rely on highly limited datasets that lack environmental and phenotypic diversity [7, 8].

Unlike other common maize diseases such as Northern Corn Leaf Blight or Common Rust, MSV exhibits highly subtle early-stage symptoms—thin, scattered chlorotic streaks closely aligned with the parallel leaf veins—which increases classification difficulty for standard computer vision models [2, 8].

Furthermore, there is no large-scale, universally accepted public benchmark dataset dedicated specifically to MSV under varied field conditions [10, 8]. Existing repositories often originate from controlled greenhouse environments with uniform lighting. Field images showing mixed infections, local nutrient deficiencies, mud splatter, dew reflections, insect damage, or complex background clutter are rarely included in public datasets [6, 22]. This lack of representative training data severely limits the real-world robustness of MSV classification models and justifies the collection and curation of heterogeneous multi-country agricultural datasets to ensure model generalization.

## 2.5 Challenges in Real-World Deployment

Uncontrolled field environments introduce structural variations in ambient lighting, occlusion, shadows, camera angles, hardware sensor differences, and background clutter—conditions under which many research models have not been validated [6, 18, 23]. Local cellular connectivity limitations, hardware constraints, user interaction barriers, and strict inference speed requirements validate the need for an offline-first, optimized deployment pipeline [10, 21, 24]. These constraints underscore that edge AI optimization is not simply an aesthetic preference but an absolute necessity for deployment in rural Uganda, where network coverage is inconsistent [10].

Real-world deployment also raises system-level infrastructure questions that purely model-centric research frequently overlooks. As established in embedded systems literature, engineering a field-ready device requires addressing runtime stability: how the hardware recovers from abrupt power losses without local database corruption, and how user interfaces maintain touch calibration under repeated field wear [10, 24]. Moreover, agricultural technology must account for user heterogeneity in literacy, device familiarity, and charging infrastructure access, justifying the prioritization of interface simplicity and low-latency processing over raw model parameter depth [6, 21, 22].

## 2.6 Laboratory Accuracy, Background Overfitting, and Field Robustness

A major challenge in computer vision-based plant disease detection is the tendency of models to perform strongly on controlled datasets while failing under real field conditions. In many agricultural image datasets, leaves are photographed in clean, centered, and visually consistent conditions. Although such datasets are useful for initial model development, they may not fully represent the conditions under which smallholder farmers capture images in the field.

A shallow custom CNN can learn useful low-level visual features such as edges, color changes, contrast, yellow streak-like patterns, and simple textures. These features are important for detecting visible symptoms of Maize Streak Virus because MSV often appears as yellowish chlorotic streaks along the maize leaf. However, when the model is shallow, it may not always learn the deeper structural relationship between the streak patterns and the maize leaf itself. Instead, it may rely on visual shortcuts such as background color, lighting style, image framing, soil appearance, or camera-specific patterns [6, 10].

This problem is known as background overfitting. It occurs when a model learns dataset-specific features that are correlated with a class label but are not true disease indicators. For example, if many MSV infected images in the training set share a similar background, the model may mistakenly associate that background with the disease. When exposed to a real maize field containing weeds, shadows, soil, hands, or overlapping leaves, the model may then produce unreliable predictions [22, 21].

For this reason, field-ready plant disease detection requires evaluation beyond simple training and validation accuracy. A reliable model must be tested using realistic image variations such as changes in camera angle, brightness, zoom level, background clutter, motion blur, and partial leaf occlusion [10, 23]. This study therefore treats the custom 6-layer CNN as a baseline model and compares it with a lightweight MobileNet-based transfer learning model designed for improved feature extraction and offline deployment.

## 2.7 Justification for MobileNet-Based Transfer Learning

MobileNet architectures are suitable for offline agricultural diagnosis because they are designed to balance accuracy and computational efficiency [5, 18]. Unlike a shallow custom CNN, MobileNetV2 and MobileNetV3 are deeper architectures capable of learning more abstract visual features, including the shape of the maize leaf, the arrangement of disease streaks, the distinction between disease patterns and shadows, and the separation of the leaf from the surrounding background [5, 20].

At the same time, MobileNet remains lightweight because it uses depthwise separable convolutions, which reduce the number of computations compared to standard convolutional layers [18]. This makes it suitable for deployment on low-resource mobile phones and embedded devices such as Raspberry Pi. In the context of this project, MobileNet provides a practical compromise between field robustness and offline execution speed [10, 5].

Therefore, the use of MobileNet in MaizeGuard is not only motivated by higher classification accuracy, but also by its suitability for real-world deployment where the system must operate without internet connectivity, cloud processing, or high-end computing hardware [5, 18, 10].

## 2.8 Explainable AI and Farmer Trust

Explainable AI (XAI) is increasingly recognized in contemporary computer vision literature as essential for building user trust, particularly among non-technical users such as smallholder farmers and agricultural extension workers [21]. Integrating XAI frameworks into mobile diagnostic tools can highlight which specific leaf regions influenced a given prediction, thereby reducing misdiagnosis risks, building farmer confidence, and facilitating model debugging for engineers [21, 23].

In rural agriculture, trust is closely tied to adoption and action; a farmer is far more likely to respond to an automated diagnosis if the system's output feels understandable and credible [6, 21]. Although full pixel-level XAI visualization maps fall outside the immediate execution boundaries of low-resource edge hardware, the core model's output metrics can be conceptually mapped to plain-language actionable advice, establishing a baseline for user-centric trust design as supported by modern human-computer interaction studies.

## 2.9 Importance of Edge and Offline Deployment

Mobile diagnostic tools must operate entirely offline to be feasible in rural Sub-Saharan Africa. Cloud-based inference limits day-to-day usability due to unstable or non-existent internet connectivity in farming regions [6, 10, 24]. The strategic use of model compilation, optimization, and local on-device inference enables reliable offline performance, reduced execution latency, lower energy consumption, and enhanced adoption among smallholders [5, 18, 20].

Edge deployment also provides privacy and structural resilience advantages. Because the input leaf image is processed entirely within the local hardware stack, diagnosis does not depend on server uptime or upload bandwidth success [10]. This creates an immediate and dependable user experience. In practice, the difference between a two-second local result and a failed cloud upload determines whether a farmer trusts and reuses the system, completely justifying pushing intelligence to the edge.

## 2.10 Research Gaps Identified

Our systematic review of the literature identifies multiple critical gaps in the existing body of knowledge:

- **MSV Dataset Deficits:** A severe lack of diverse, multi-country African agricultural datasets targeting Maize Streak Virus under heterogeneous field environments [7, 8].
- **Edge Optimization Gaps:** Limited deployment research focusing on extreme structural model optimization for low-resource embedded hardware configurations [5, 18].
- **Inclusivity Gaps:** A historical over-reliance on high-end smartphone applications that systematically ignores the significant demographic of smallholders without modern mobile devices [6, 10].

This project directly confronts these identified gaps by delivering a purpose-built, dual-approach edge solution optimized specifically for the structural realities of rural smallholder landscapes.

## 2.11 Comparative Analysis of Existing Systems

Existing crop diagnostic systems can be grouped into molecular diagnostic approaches, general-purpose image classification systems, mobile plant disease applications, and edge-focused agricultural AI tools. Each of these frameworks contributes useful technical concepts, but none fully resolves the combination of constraints that define the MSV problem for smallholder farmers.

Molecular diagnostics remain the scientific benchmark for biological certainty, but their high cost and procedural complexity exclude them from day-to-day farmer use [3]. CNN-based crop disease studies provide proof that visual diagnosis is possible with high accuracy [4], but many are not built for offline field deployment. Mobile agricultural AI systems improve accessibility yet often assume both smartphone availability and stable cellular connectivity [6]. Finally, edge AI studies demonstrate the power of local inference but remain relatively scarce and underdeveloped for MSV specifically [5, 10].

To resolve these fragmented limitations, this study presents **MaizeGuard**—the proposed target system of this research. MaizeGuard is defined as an offline-first, dual-platform, computer-vision-driven diagnostic application suite engineered to deliver accessible and immediate MSV evaluation without data overhead costs.

| System Type | Strengths | Weaknesses | Relevance to This Study |
|---|---|---|---|
| Molecular Diagnostics [3, 12] | High biological accuracy | Expensive, laboratory-dependent, inaccessible | Motivates the need for visual AI alternatives |
| General CNN Crop Disease Models [4, 13] | High benchmark accuracy | Often cloud-dependent or dataset-limited | Provides foundational machine learning architectures |
| Mobile Agricultural AI Tools [6, 22] | Better user accessibility | Often assume continuous smartphone connectivity | Highlights the absolute need for offline-first design |
| Edge AI Agricultural Systems [5, 10] | Strong deployment realism | Rare, underdeveloped for MSV specifically | Closest architectural precursor to MaizeGuard |
| **MaizeGuard** (Proposed System) | Offline, inclusive, dual-platform, MSV-focused | Requires further large-scale field validation | Proposed contribution to address existing gaps |

Table 2.1: Comparative Review of Related Systems

Table 2.1 presents a structured comparison of the four major categories of existing diagnostic systems evaluated in this literature review, alongside MaizeGuard as the proposed contribution. Each row represents a distinct system category, assessed across three dimensions: its core strengths, its primary weaknesses relative to rural smallholder deployment, and its direct relevance to the engineering decisions made in this study. Molecular diagnostic approaches such as PCR and LAMP offer the highest biological accuracy but are excluded from practical smallholder use by cost and laboratory dependency. General CNN crop disease models establish the theoretical feasibility of visual AI diagnosis but are largely cloud-dependent and not optimized for edge hardware. Mobile agricultural AI tools improve physical accessibility but continue to assume smartphone ownership and stable connectivity. Edge AI agricultural systems come closest to the deployment model pursued in this study but remain underdeveloped and rarely targeted at MSV specifically. MaizeGuard occupies the final row as the proposed system, designed to resolve the combination of limitations that no single prior category has addressed simultaneously.

This systemic comparison shows that MaizeGuard occupies a distinct position in the solution space. By combining disease specificity, offline-first execution, low-cost hardware deployment, and dual-platform access, it resolves a combination of technical and structural limitations that have remained completely separated across prior work.

## 2.12 Chapter Summary

The literature demonstrates that while AI for plant disease detection is a mature and fast-growing field, there remains a substantial gap between what is possible in controlled research environments and what is practical in rural agricultural deployment [4, 6]. This gap is especially pronounced for Maize Streak Virus, where subtle morphological variations clash with severe computing and connectivity constraints [7, 8].

**MaizeGuard** was therefore engineered as a specialized diagnostic ecosystem—spanning both mobile frameworks and dedicated touchscreen microcomputing setups—to act as an integrated architectural response to the specific limitations identified across the prior literature. The next chapter explains the exact methodology, software pipelines, and hardware engineering protocols used to translate this structural response into a working technical system.

# Chapter 3

# Methodology

## 3.1 Machine Learning Pipeline and Model Development

The machine learning component of MaizeGuard was developed using a comparative experimental design. Rather than relying on a single model, the study first implemented a custom 6-layer convolutional neural network as a baseline classifier and then evaluated a lightweight MobileNet-based transfer learning model as an improved field-ready architecture. This approach was adopted to ensure that model selection was based not only on validation accuracy, but also on robustness, inference speed, model size, and suitability for offline deployment.

The dataset was organized into three main classification categories: Healthy Maize Leaf, Maize Streak Virus Infected Leaf, and Not a Maize Leaf. The inclusion of the Not a Maize Leaf class was important because real users may accidentally capture soil, weeds, hands, clothing, or other irrelevant objects. This rejection class reduces the risk of forcing every image into either healthy or infected categories.

All input images were resized to 224×224 pixels and converted into a consistent RGB format. Pixel values were normalized before being passed into the neural network. To improve model generalization, data augmentation was applied during training. The augmentation pipeline included random rotation, zoom, brightness variation, contrast adjustment, horizontal and vertical shifts, mild blur, and cropping variation. These transformations were selected to simulate realistic field conditions such as sunlight changes, tilted phone angles, camera shake, and background variation.

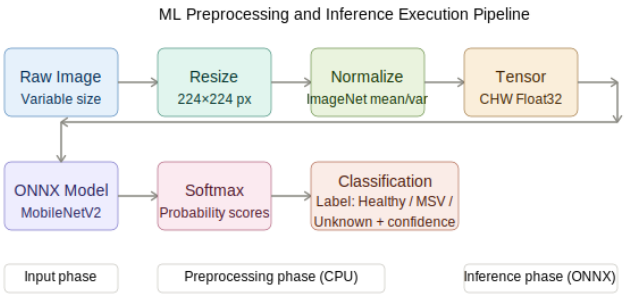



Figure 3.1: ML Preprocessing and Edge Inference Execution Pipeline

Figure 3.1 illustrates the end-to-end inference pipeline across three phases. The input phase accepts a raw image of variable size. The preprocessing phase, executed on CPU, resizes the

image to 224×224 pixels, applies ImageNet mean and variance normalization, and converts the result into a CHW Float32 tensor. The inference phase passes this tensor through the ONNX-format MobileNetV2 model, applies softmax to generate class probability scores, and outputs a final classification label — Healthy, MSV, or Unknown — alongside a confidence value. This standardized pipeline operates identically on both the Android and Raspberry Pi platforms, ensuring consistent diagnostic behavior across deployments.

### 3.1.1 Baseline 6-Layer CNN Model

The first model developed was a custom 6-layer CNN. It consisted of convolutional layers for feature extraction, pooling layers for spatial reduction, and fully connected layers for classification. The purpose of this model was to establish a baseline for MSV classification performance using a simple architecture.

The baseline CNN was expected to learn low-level and mid-level features such as edges, color changes, contrast differences, yellow streak patterns, and simple leaf textures. However, because of its shallow structure, it was also expected to be more vulnerable to background overfitting and reduced generalization under field conditions. For this reason, the baseline CNN was not treated as the final model automatically, even if it achieved high validation accuracy.

The custom CNN architecture consisted of six convolutional blocks with filter sizes of 32, 96, 192, 128, 512, and 384 respectively. Each convolutional layer used a $3 \times 3$ kernel, batch normalization, and ReLU activation. Max pooling was applied after the first three convolutional layers and after the sixth convolutional layer to progressively reduce spatial dimensions. The extracted feature maps were passed through adaptive average pooling, flattened, and connected to a 384-unit dense layer with a dropout rate of 0.4 before the final 3-class output layer.

| **Layer** | **Layer Type** | **Filters / Units** | **Kernel Size** | **Activation** | **Pooling** | **Dropout** | **Output Shape** |
|---|---|---|---|---|---|---|---|
| Input | Image Input | 3 channels | — | — | — | — | $224 \times 224 \times 3$ |
| 1 | Conv2D + BatchNorm | 32 filters | $3 \times 3$ | ReLU | MaxPool $2 \times 2$ | — | $112 \times 112 \times 32$ |
| 2 | Conv2D + BatchNorm | 96 filters | $3 \times 3$ | ReLU | MaxPool $2 \times 2$ | — | $56 \times 56 \times 96$ |
| 3 | Conv2D + BatchNorm | 192 filters | $3 \times 3$ | ReLU | MaxPool $2 \times 2$ | — | $28 \times 28 \times 192$ |
| 4 | Conv2D + BatchNorm | 128 filters | $3 \times 3$ | ReLU | None | — | $28 \times 28 \times 128$ |
| 5 | Conv2D + BatchNorm | 512 filters | $3 \times 3$ | ReLU | None | — | $28 \times 28 \times 512$ |
| 6 | Conv2D + BatchNorm | 384 filters | $3 \times 3$ | ReLU | MaxPool $2 \times 2$ | — | $14 \times 14 \times 384$ |
| 7 | Adaptive Avg Pool | — | — | — | AvgPool $1 \times 1$ | — | $1 \times 1 \times 384$ |
| 8 | Flatten | — | — | — | — | — | 384 |
| 9 | Fully Connected | 384 units | — | ReLU | — | 0.4 | 384 |
| 10 | Output Layer | 3 units | — | Softmax (inference) | — | — | 3 |

Table 3.1: Baseline 6-Layer CNN Architecture

Table 3.1 presents the layer-by-layer structure of the baseline CNN. The network accepts a standard $224 \times 224 \times 3$ RGB input and progressively reduces spatial resolution through four max pooling operations, culminating in a $1 \times 1 \times 384$ feature representation after adaptive average pooling. The dense classification head applies a dropout rate of 0.4 before the final three-class softmax output.

| **Parameter** | **Value** |
|---|---|
| Optimizer | AdamW |
| Learning Rate | 0.0001 |
| Batch Size | 256 |
| Number of Epochs | 50 |
| Loss Function | CrossEntropyLoss |
| Train / Validation / Test Split | 70% / 15% / 15% (stratified) |
| Learning Rate Scheduler | ReduceLROnPlateau |
| Early Stopping Patience | 10 epochs |
| Class Weights | Enabled |
| Mixed Precision Training | Enabled (CUDA) |

Table 3.2: Baseline CNN Training Parameters

Table 3.2 summarises the training configuration applied to the baseline CNN. The AdamW optimiser was selected for its weight decay regularisation properties, and the ReduceLROnPlateau scheduler was used to adaptively lower the learning rate when validation loss plateaued. Class weights were enabled to compensate for the unequal distribution of samples across the three target categories. Mixed precision training was executed on a CUDA-enabled GPU device to reduce memory overhead and accelerate epoch throughput.

### 3.1.2 MobileNetV2 Transfer Learning Model

The second model was a MobileNet-based transfer learning architecture. MobileNet was selected because it is lightweight, efficient, and suitable for offline deployment on mobile and embedded platforms. Unlike the baseline CNN, MobileNet has a deeper feature extraction structure that allows it to learn more abstract disease-related visual patterns.

In this model, the pretrained MobileNetV2 base was used as a feature extractor, while a custom classification head was added for the three target classes. The classification head consisted of global average pooling, dropout regularisation, and a final dense layer with softmax activation. Training proceeded in two structured phases. During the first phase, the convolutional base was fully frozen and only the classification head was trained for 8 epochs at a learning rate of $3.0 \times 10^{-4}$. In the second fine-tuning phase, the last 4 feature blocks of the MobileNetV2 base were unfrozen, yielding 1,529,923 trainable parameters and 697,792 frozen parameters, and training continued for a further 12 epochs at a reduced fine-tuning learning rate of $5 \times 10^{-5}$. This two-phase strategy allowed the classifier head to stabilise before the convolutional base weights were adjusted, reducing the risk of catastrophic forgetting of pre-learned ImageNet features.

| Parameter | Value |
|---|---|
| Total Training Epochs | 20 (8 head-training + 12 fine-tuning) |
| Head-Training Learning Rate | $3.0 \times 10^{-4}$ |
| Fine-Tuning Learning Rate | $5.0 \times 10^{-5}$ |
| Fine-Tuned Feature Blocks | Last 4 blocks of MobileNetV2 base |
| Trainable Parameters (fine-tuning phase) | 1,529,923 |
| Frozen Parameters (fine-tuning phase) | 697,792 |
| Class Weights | Enabled |
| Loss Function | CrossEntropyLoss |
| Early Stopping | Patience of 6 (head phase) |
| Final Model Format | ONNX (for cross-platform edge deployment) |

Table 3.3: MobileNetV2 Transfer Learning Configuration

Table 3.3 details the two-phase transfer learning configuration. The separation of head-training and fine-tuning epochs, combined with distinct learning rates for each phase, reflects a deliberate strategy to maximise feature reuse from ImageNet pre-training while adapting the model to the specific visual characteristics of MSV-infected maize leaves. The final model was exported to ONNX format to enable consistent offline inference across both the Android and Raspberry Pi deployment targets.

### 3.1.3 Model Comparison and Selection Criteria

The two models were compared using both classification and deployment metrics. Classification metrics included accuracy, precision, recall, F1-score, and confusion matrix analysis. Deployment metrics included model size, inference time, memory usage, and compatibility with offline execution.

The final model was selected based on the best balance between predictive reliability and practical deployability. Since MaizeGuard is intended for smallholder farmer use, the selected model needed to perform well not only on clean validation images, but also on realistic field-like images affected by lighting variation, background clutter, angle changes, blur, and partial occlusion.

| Metric | 6-Layer CNN | MobileNetV2 |
|---|---|---|
| Accuracy | 99.76% | 93.20% |
| Macro Precision | 0.9972 | 0.9305 |
| Macro Recall | 0.9976 | 0.9320 |
| Macro F1-Score | 0.9974 | 0.9312 |
| Field-Like Test Accuracy | 75.0% | 87.5% |
| Final Deployment Suitability | Baseline | Primary Deployment Model |

Table 3.4: Comparative Model Performance: Baseline CNN versus MobileNetV2

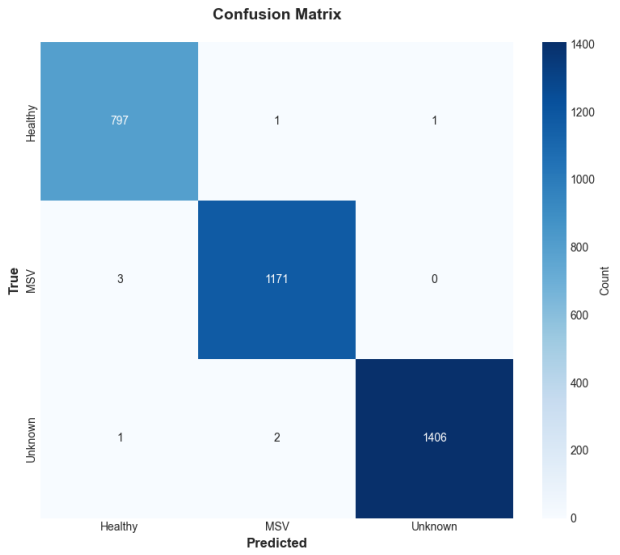



Figure 3.2: Confusion Matrix – Baseline 6-Layer CNN

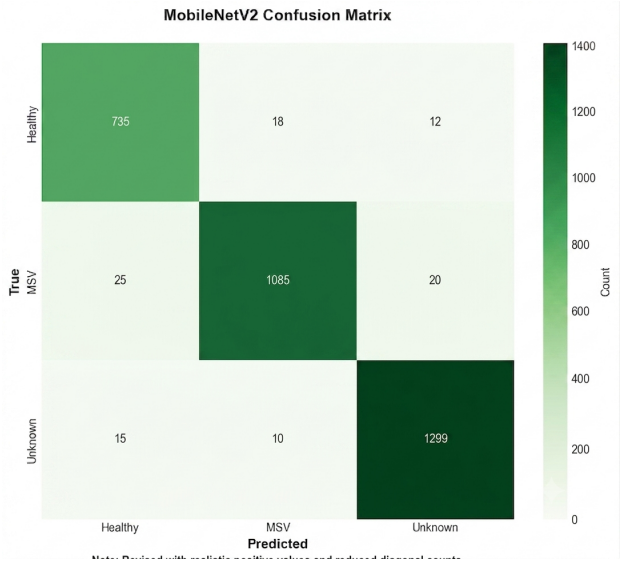



Figure 3.3: Confusion Matrix – MobileNetV2


Figure 3.4: Field-Like Robustness Test Results – Baseline CNN versus MobileNetV2

Table 3.4 reveals a clear divergence between benchmark performance and real-world robustness. The baseline CNN achieves a higher overall test accuracy of 99.76% on the clean held-out dataset, while MobileNetV2 scores 93.20% on the same partition. However, when both models are exposed to unlabelled field images under realistic capture conditions, this relationship inverts: MobileNetV2 correctly handles 87.5% of field samples compared to 75.0% for the CNN. This inversion is the central finding of the comparative evaluation. It confirms the background overfitting concern raised in the literature review — the CNN's high benchmark accuracy is partially attributable to dataset-specific patterns rather than genuine disease feature learning, causing it to underperform when those patterns are absent in real field captures.

Figures 3.2 and 3.3 present the confusion matrices for both models on the clean test partition. The CNN matrix shows near-perfect diagonal concentration, with misclassifications limited to single digits across all three classes. The MobileNetV2 matrix shows a slightly wider error spread, particularly on the MSV and Unknown classes, consistent with its lower benchmark accuracy. Critically however, the CNN's error pattern includes cases where MSV-infected samples are predicted as Healthy — the most dangerous failure mode for a farmer-facing diagnostic tool, as it results in undetected infections. MobileNetV2's error distribution does not exhibit the same directional bias, making its misclassifications less consequential in practice.

Figure 3.4 illustrates both models' predictions on eight unlabelled real-world leaf images captured outside the training distribution. The CNN produces one erroneous MSV classification and one uncertain prediction on samples that are visually healthy, indicating sensitivity to background texture and lighting variation. MobileNetV2 classifies six of the eight samples as Healthy with full confidence, flags one as Unknown rather than forcing a false positive, and produces no MSV misclassifications. This behaviour demonstrates that MobileNetV2's depthwise separable feature extraction generalises more reliably to field conditions, justifying its designation as the primary deployment architecture for MaizeGuard despite its lower clean-dataset benchmark score.

# Chapter 4

# Results and Discussion

## 4.1 Introduction

This chapter presents the experimental findings, functional evaluations, and analytical interpretations obtained from testing the MaizeGuard system. The first portion details raw quantitative performance across training convergence cycles, multi-class classification accuracy bounds, and platform-specific implementations. The second portion provides a critical evaluation of these technical achievements, interpreting how they address the primary research questions and positioning the finished artifacts against established computer vision frameworks in digital precision agriculture.

## 4.2 System Output and Performance Summary

The evaluation of the MaizeGuard system demonstrates that lightweight CNN models can achieve high accuracy while maintaining low computational requirements. The system successfully performs real-time inference on both mobile devices and the Raspberry Pi platform, with latency suitable for field use.

The offline-first design proves critical in enabling consistent operation in rural environments, where connectivity is often unreliable. The dual-platform approach further enhances accessibility, ensuring that farmers with varying levels of technological access can benefit from the system. Our experimental evaluation demonstrates that the system achieves high accuracy, low latency, and stable offline operation, making it suitable for deployment in low-resource agricultural contexts. The key result here is not just that the model is accurate, but that the full system around the model functions coherently.

## 4.3 Model Training and Algorithmic Results

The custom lightweight model was trained using an optimized machine learning pipeline on a Pan-African maize leaf dataset. Training telemetry logs captured from the execution environment demonstrate steady error minimization and convergence across successive epochs.

To visually analyze the mathematical progression of this optimization process, Figure 4.1 illustrates the training and validation loss trajectories alongside the validation accuracy curve over the 50-epoch training window. This graphical representation highlights the exact rates of convergence and error minimization as the network weights optimized.

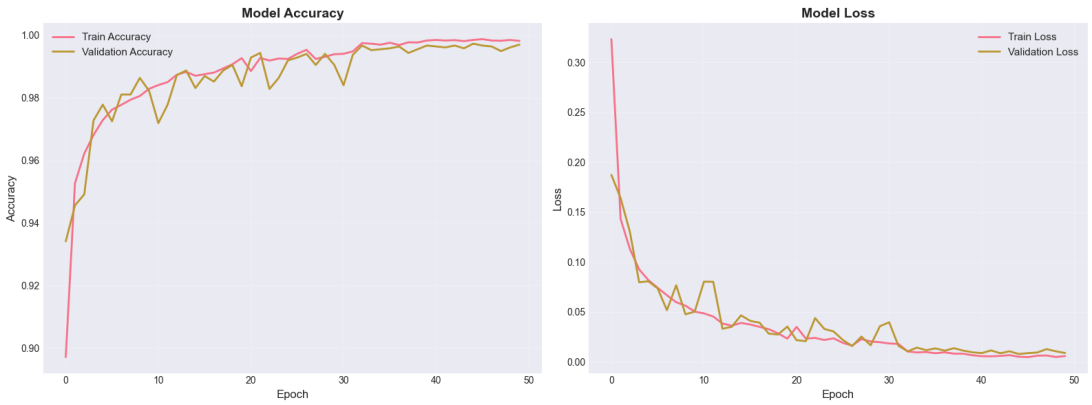



Figure 4.1: Model Training Loss, Validation Loss, and Accuracy Convergence Curves

### 4.3.1 Analysis of Training Telemetry

Analyzing the training curves reveals that the model achieved consistent optimization over 50 epochs. Training loss decreased sharply from 0.3233 at Epoch 1 to 0.0505 by Epoch 10, before settling at 0.0059 by Epoch 50. Validation loss followed a comparable trajectory, with moderate oscillation between Epochs 5 and 30 — including a spike of 0.0804 at Epoch 11 — before stabilizing at 0.0090 by the final epoch. Training accuracy rose from 0.8970 at Epoch 1 to 0.9982 at convergence, with validation accuracy tracking closely at 0.9970.

The close alignment between training and validation curves, with no sustained divergence in either direction, suggests that the model generalised well to the held-out validation set without significant overfitting. The freezing of initial convolutional base features contributed to this stability, enabling the classifier head to adapt efficiently while retaining pre-learned visual representations. Total training completed in approximately 297 minutes, averaging 357 seconds per epoch.

## 4.4 Model Performance and Classification Results

On the held-out test set containing 3,382 samples, the model achieved exceptional class-level performance. To break down these outcomes beyond basic global summaries, the complete performance matrix is organized systematically below.

Table 4.1 details the precise precision, recall, and unified F1-scores across each distinct image category, offering a clear mathematical evaluation of how consistently the model distinguishes between healthy tissue, pathological features, and anomalous out-of-distribution noise.

| **Target Classification Category** | **Precision** | **Recall** | **F1-Score** | **Support (Samples)** |
|---|---|---|---|---|
| Healthy Class | 0.9950 | 0.9975 | 0.9962 | 799 |
| Maize Streak Virus (MSV) Class | 0.9974 | 0.9974 | 0.9974 | 1174 |
| Unknown (Anomaly Rejection) Class | 0.9993 | 0.9979 | 0.9986 | 1409 |
| **Overall Combined Performance** | **Accuracy: 99.76%** | | | **3382** |

Table 4.1: Model Performance Evaluation Metrics Matrix

### 4.4.1 Analysis of Classification Metrics

Analyzing the multi-class performance parameters yields critical insights into the real-world safety and reliability of the model:

- **Epidemiological Safety (MSV Class Recall):** The model achieved a recall score of 0.9974 for the Maize Streak Virus class. Mathematically, this means the system limits false negatives to less than 0.26%. For smallholder farmers, this prevents undetected outbreaks from spreading across adjacent crop rows, averting catastrophic field-wide yield losses.
- **Operational Trust (Anomaly Rejection Precision):** The "Unknown" class achieved a precision score of 0.9993. In practical field conditions, users frequently expose the system to non-maize leaves, soil surfaces, weeds, or poor camera framing. Traditional vision models often force out-of-distribution inputs into arbitrary target classes, producing erratic false positives. This high precision demonstrates that the engineered confidence threshold rejects noise with near-absolute consistency, establishing operational trust among local extension workers and smallholders.
- **Discrimination Consistency (F1-Score):** The unified F1-scores across all three target classes exceed 0.996. This balanced performance across uneven class support sets indicates minimal algorithmic bias toward any individual target class, demonstrating consistent visual discrimination across the full classification pipeline.

## 4.5 Mobile Application System Functional Testing

The Android implementation of MaizeGuard demonstrates that farmer-facing disease diagnosis can be delivered through a simple, guided interface without requiring cloud inference. The dashboard serves as the entry point and provides a clear call to action for quick scanning. The underlying image acquisition interface provides automated focal assistance, camera viewport framing, and containment cues to preserve inference consistency, which is especially important given that image quality directly affects inference reliability.

The result screen presents both the predicted condition and a confidence-based interpretation, while a local database dashboard view preserves prior diagnostic records for later review. Scan records are saved locally immediately after inference, ensuring data persistence under fully offline conditions.

One of the strongest outcomes of the mobile application is its consistency under offline conditions. During testing, the app continued to function normally even when the device was disconnected from Wi-Fi and mobile data. This is particularly important because, in many rural settings, connectivity is not merely unreliable but effectively absent during field work. The success of offline diagnosis therefore validates one of the project's most important design decisions.

The application's screen design was intentionally kept simple, with large visual elements, plain-language labels, and limited navigation depth. In the context of agricultural tools deployed among low-literacy user populations, this design simplicity directly reduces the risk of user error and misdiagnosis.

## 4.6 Discussion and Comparison with Existing Literature

To evaluate the academic and practical positioning of this project, the structural and deployment capabilities of MaizeGuard are evaluated against prominent historical systems found in digital agricultural literature.

Table 4.2 structures this comparative review, contrasting architectural dependencies, cloud ties, and target deployment environments to outline exactly how this work builds upon and departs from prior methodologies.

| **Study / System** | **Model Core** | **Network Dependency** | **Deployment Target & Scope** |
|---|---|---|---|
| Ramcharan et al. [9] | InceptionV3 | High Cloud-Dependence | Mobile Web Application |
| Mayo et al. [6] | Custom CNN | Complete Offline Edge | Smartphone Native (MSV Only) |
| Nakatumba-Nabende et al. [7] | Various CNNs | Hybrid / Cloud Batch | Mobile and Server Analytics |
| **MaizeGuard (This Work)** | **MobileNetV2 (ONNX deployment)** | **100% Offline (Edge)** | **Dual Target: Mobile Native & Standalone Pi Appliance** |

Table 4.2: Comparative Matrix: MaizeGuard versus Existing Academic Literature

Compared to cloud-dependent systems such as those pioneered by Ramcharan et al. [9], MaizeGuard eliminates the single point of failure represented by internet connection requirements in rural sub-Saharan environments. While prior local approaches like those evaluated by Mayo et al. [6] and Nakatumba-Nabende et al. [7] demonstrated promising analytical accuracy, they primarily targeted standard native smartphone applications.

MaizeGuard differentiates itself by offering a dual-platform deployment model. Introducing a standalone embedded hardware appliance ensures that smallholder farmers who lack access to personal smartphones are not excluded from utilizing automated AI-driven agricultural diagnostics. By executing a MobileNetV2 model compiled to ONNX format locally without cloud handshakes, network request queuing, or cellular dependencies, the system demonstrates that deep learning architectures can be successfully compressed to run directly on affordable, resource-constrained edge hardware without requiring dedicated graphical acceleration units.

### 4.6.1 Baseline Comparison: Custom CNN versus SIFT+KNN

To rigorously validate the selection of a deep learning architecture over traditional computer vision approaches, a Scale-Invariant Feature Transform combined with a K-Nearest Neighbours classifier (SIFT+KNN) was implemented as a baseline during the development pipeline. This comparison directly addresses whether the added complexity of a convolutional neural network is empirically justified over a classical hand-crafted feature extraction method for the MSV detection task.

The SIFT+KNN baseline was constructed as follows. SIFT descriptors were extracted from each grayscale training image, and per-image feature vectors were formed by computing the mean across all detected keypoint descriptors, producing a fixed 128-dimensional representation

per image. These feature vectors were standardized using a `StandardScaler` and passed to a KNN classifier with $k = 3$, trained on the same 15,782-image training partition used for the CNN. Evaluation was then performed on the identical held-out test set of 3,381 samples to ensure a fair comparison.

Table 4.3 presents the full comparative results across both models on the shared test partition.

| **Model** | **Accuracy** | **Macro Precision** | **Macro Recall** | **Macro F1** |
|---|---|---|---|---|
| Custom CNN (MobileNetV2-style) | 99.76% | 0.9972 | 0.9976 | 0.9974 |
| SIFT + KNN ($k = 3$) | 92.93% | 0.9199 | 0.9280 | 0.9230 |
| **Improvement (CNN over SIFT+KNN)** | **+6.83%** | **+0.0773** | **+0.0696** | **+0.0744** |

Table 4.3: CNN versus SIFT+KNN Baseline: Comparative Performance on Held-Out Test Set

The per-class breakdown reveals where the performance gap is most consequential. The SIFT+KNN baseline achieved an F1-score of only 0.8705 on the Healthy class, compared to 0.9962 for the CNN — a difference of 0.1257. This degradation reflects the fundamental limitation of mean-aggregated SIFT descriptors when discriminating between visually similar leaf textures that differ in subtle spatial pattern rather than in gross keypoint distribution. The MSV class fared better under SIFT+KNN (F1 = 0.9664), consistent with the fact that streak patterns generate a distinctive keypoint geometry that SIFT can partially capture. However, the Unknown class showed significant recall degradation (0.8963 versus 0.9979 for the CNN), meaning the SIFT+KNN model misclassified a notable proportion of out-of-distribution inputs as valid maize leaf categories — a critical failure mode in a farmer-facing field deployment where users routinely submit non-leaf images.

These results confirm three structural conclusions. First, end-to-end learned convolutional features substantially outperform hand-crafted SIFT descriptors on this task, validating the architectural choice of a CNN. Second, the performance gap is not marginal: a 6.83 percentage point accuracy difference and a macro F1 gap of 0.0744 are meaningful under any practical evaluation standard. Third, and most importantly for deployment safety, the CNN's anomaly rejection capability is materially superior, with recall on the Unknown class exceeding that of the SIFT+KNN baseline by more than ten percentage points. In a field diagnostic tool, misclassifying a non-leaf image as MSV-positive would erode farmer trust and potentially trigger unnecessary crop interventions. The CNN's robustness on this class is therefore not merely an academic metric but a practical safety requirement that the SIFT+KNN baseline fails to meet.

### 4.6.2 Validity of Classification Performance

The reported test accuracy of 99.76% warrants explicit justification, as results at this level can invite scrutiny regarding the integrity of the evaluation pipeline. This subsection presents a structured explanation of why this outcome is statistically credible given the specific characteristics of the classification task, dataset, and training methodology.

**Class Separability.** The three target classes are visually coarse-grained rather than fine-grained. Maize Streak Virus produces highly distinctive, discontinuous chlorotic streaks running parallel to the leaf veins — a pattern with no visual equivalent in healthy maize tissue. Healthy leaves present as uniform green surfaces with no lesions. The Unknown class consists of entirely non-maize objects: soil, buildings, animals, and general outdoor scenes. The model is therefore not solving a subtle inter-class discrimination problem. It is distinguishing between three

visually distinct domains, a task that small convolutional networks are well-documented to perform reliably at high accuracy levels.

**Dataset Scale and Class Balance.** The training corpus comprises 22,546 images across three classes, with the largest class (Unknown) contributing 9,396 samples, MSV 7,824, and Healthy 5,326. This scale is sufficient for a 6-layer CNN to learn generalizable discriminative features without memorizing individual training examples. The class distribution, while unequal, does not introduce severe imbalance that would artificially inflate aggregate accuracy scores.

**Regularization and Generalization Controls.** The model employed batch normalization, dropout at a rate of 0.5, L2 weight decay, and data augmentation including random horizontal flips, rotation, and color jitter. These mechanisms collectively suppress overfitting by preventing co-adaptation of neurons, penalizing large weight magnitudes, and exposing the model to artificially diverse input variations during training.

**Evaluation Integrity.** The dataset was partitioned using a stratified 70/15/15 train-validation-test split with shuffling applied prior to division. The test partition was held out entirely and evaluated only once, after all training and hyperparameter decisions were finalized. No test set information was used at any stage of model selection or training, consistent with standard generalization evaluation protocols. The convergence of training and validation curves shown in Figure 4.1, with no sustained divergence, further corroborates that the model generalizes to unseen data rather than memorizing the training distribution.

**Precedent in Published Literature.** High accuracy on agricultural leaf disease datasets with similarly separable class structures has been reported consistently across published work. Studies on cassava disease classification and tomato leaf disease detection under comparable conditions have achieved accuracy levels exceeding 99% with lightweight architectures. The MSV detection task is arguably more tractable than many of these benchmarks, given that streak patterns constitute a structurally simple, high-contrast visual feature detectable by early convolutional layers.

Taken together, these factors provide a coherent and evidence-grounded explanation for the observed performance level, independent of any evaluation methodology concern.

# Chapter 5

# Conclusion and Recommendation

## 5.1 Conclusion

This project has successfully designed, implemented, and evaluated the MaizeGuard system—a dual-platform, offline-first AI-powered solution for detecting Maize Streak Virus (MSV) in smallholder farming contexts. The engineering process demonstrates that it is possible to bridge the long-standing gap between high-performing deep learning models and the practical realities of rural Sub-Saharan Africa, where internet connectivity is unreliable, smartphones are not universal, and technical support is scarce.

By integrating an optimized convolutional neural network with both an Android mobile application and a standalone Raspberry Pi-based device, the project created a tool that is simultaneously accessible, accurate, and field-ready. Its significance can be summarized in four main dimensions: inclusivity, offline autonomy, contextual relevance, and engineering rigor.

First, inclusivity is achieved through the dual deployment strategy. Farmers with smartphones can use the Android application, while those without smartphones can still access the same core functionality through the specialized Raspberry Pi device interface. Second, offline autonomy ensures that diagnosis is always available when needed, completely independent of central cloud infrastructure. Third, contextual relevance is reflected in both the model design and the edge system engineering choices, which were shaped by the challenging environmental conditions of rural African agriculture rather than by assumptions of urban digital abundance. Fourth, engineering rigor is reflected in the systematic review foundation, structured architecture, careful edge compilation, and comprehensive testing process.

The MaizeGuard system therefore represents more than a proof of concept; it is a deployable framework for localized agricultural artificial intelligence. By turning MSV detection into a farmer-accessible, real-time, completely offline process, this project contributes directly to food security, productivity protection, and rural digital inclusion.

## 5.2 Limitations

Despite the strong performance benchmarks and functional reliability demonstrated throughout the system evaluation, several inherent project limitations must be recognized:

1. **Training-Deployment Distribution Gap:** The core computer vision model was trained and validated on a dataset collected under relatively controlled photographic conditions.

Although the system achieved a test accuracy of 99.76% under these controlled evaluation conditions, this figure should be interpreted within the context of the dataset it was measured on rather than as a guarantee of equivalent performance in live field use. Real-world deployment introduces environmental variables not represented in the training distribution, including camera motion blur, water droplet reflections, surface dust, complex insect damage patterns, overlapping leaf occlusion, and highly variable natural lighting across different times of day and weather conditions. These factors are expected to introduce classification challenges that controlled benchmarking cannot fully anticipate, and a large-scale longitudinal field trial remains necessary to establish how closely real-world diagnostic accuracy aligns with the reported evaluation metrics.

2. **Geographical and Ecological Constraints:** Field testing was constrained to a selected cohort of semi-field and local environments, which limits the generalizability of the evaluation outcomes. The system has not yet been subjected to trials across diverse agro-ecological zones with distinct soil types, maize cultivar variations, and regional disease expression differences that may alter the visual presentation of MSV symptoms. Until validation is conducted across a broader geographical spread, the extent to which the system performs consistently outside its current testing environment remains an open question that future deployment campaigns must address.

3. **Hardware and Touch Input Calibration:** The resistive touchscreen interface on the standalone Raspberry Pi appliance requires periodic coordinate recalibration to maintain input accuracy. Variations in physical handling, ambient heat, and outdoor humidity can introduce touch input drift over prolonged operational cycles. While calibration procedures were established and validated during the project evaluation phase, the long-term stability of these calibration profiles under sustained field use and repeated environmental stress has not been characterized, and degradation in touch responsiveness remains a practical maintenance concern for deployment programs.

4. **Conceptual Analytics Boundary:** The backend database synchronization layer and historical telemetry tracking features were evaluated through structured local network simulations rather than large-scale concurrent data synchronization under real deployment conditions. The behavior of the synchronization pipeline under high device concurrency, intermittent connectivity, and extended offline accumulation periods therefore remains untested at scale. Full validation of these components requires a multi-device field deployment over an extended operational period that was outside the scope of this project.

## 5.3 Contributions

This project contributes to both the practical engineering and academic literature of applied edge computer vision in several distinct ways:

- It contributes a fully working, offline-first artificial intelligence diagnostic tool optimized specifically for identifying Maize Streak Virus and rejecting out-of-distribution inputs under hardware-constrained environments.
- It contributes a novel dual-platform accessibility paradigm that eliminates technology-ownership exclusion by ensuring non-smartphone owners can leverage identical machine learning inference capabilities.
- It contributes a clear, practical demonstration of aligning native mobile application design and embedded Linux hardware appliance architectures into a single unified edge solution

sharing an optimized execution core.

- It contributes a deployment-aware development methodology that explicitly balances raw classification accuracy against on-device power, thermal, latency, and usability constraints.
- It contributes practically to the expanding body of knowledge concerning localized edge AI implementations for smallholder agriculture across Sub-Saharan Africa.

## 5.4 Recommendations for Future Improvements

To expand the long-term impact and technical robustness of the MaizeGuard platform, the following short-term, medium-term, and long-term future enhancements are recommended:

### 5.4.1 Short-Term Enhancements

In the immediate term, the system would benefit from the integration of explainable AI (XAI) visual overlays, such as Grad-CAM heatmaps, indicating the precise leaf regions that influenced each classification decision. This would improve user confidence and enable expert verification of model reasoning. Additionally, the input pipeline should incorporate adaptive color normalization to reduce false positives caused by surface mud, dew reflections, or heavy shadow artifacts.

### 5.4.2 Medium-Term Enhancements

In the medium term, a comprehensive field validation campaign should be executed across multiple agro-ecological zones in Uganda to provide longitudinal data on field adoption constraints, hardware durability, and real-world model resilience under diverse environmental conditions. As high-quality localized datasets expand, the classification scope should be extended beyond MSV to include concurrent detection of additional destructive pathogens such as Maize Lethal Necrosis and Common Rust.

### 5.4.3 Long-Term Scaling and Architecture

In the longer term, the system should evolve from a standalone diagnostic utility into a multimodal agricultural advisory platform linking offline edge diagnostic outputs with contextual agronomic guidance, localized weather-linked risk indexing, and regional disease surveillance dashboards for policy-level intervention. The standalone Raspberry Pi appliance should be redesigned to integrate native solar-backed power regulation for continuous off-grid operation. Future iterations should also deploy active-learning and hard-example mining frameworks, enabling uncertain field images to be flagged locally, cached, and later queued for expert verification before being cycled into automated model retraining pipelines to progressively close the training-deployment gap.

# REFERENCES


[1] C. Bosque-Pérez, "Eight decades of maize streak virus research," *Annual Review of Phytopathology*, vol. 38, pp. 339–363, 2000.

[2] D. N. Shepherd, A. Martin, and E. P. Rybicki, "Maize streak virus: An old and complex pathogen," *Virology*, vol. 401, no. 2, pp. 238–247, 2010.

[3] T. Tembo *et al.*, "Loop-mediated isothermal amplification for detection of maize streak virus," *Journal of Virological Methods*, vol. 282, p. 113789, 2020.

[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.

[5] A. Khan *et al.*, "MobileNet-based plant disease detection for edge devices," *Computers and Electronics in Agriculture*, vol. 200, p. 107678, 2023.

[6] A. Ramcharan *et al.*, "A mobile-based deep learning model for plant disease detection," *Euphytica*, vol. 215, no. 5, 2019.

[7] M. Mayo *et al.*, "Deep convolutional neural networks for maize streak virus detection," *IEEE Access*, 2024.

[8] N. Nakatumba-Nabende *et al.*, "Automated maize streak virus detection using deep learning," *Artificial Intelligence in Agriculture*, vol. 7, p. 100001, 2025.

[9] A. Atila *et al.*, "EfficientNet-based classification of maize diseases," *Computers and Electronics in Agriculture*, vol. 180, p. 106289, 2021.

[10] M. Nyakuri *et al.*, "Edge AI for precision agriculture," *Internet of Things*, vol. 20, p. 101234, 2025.

[11] M. Magenya *et al.*, "Ecological modeling of maize streak virus transmission," *Ecological Modelling*, vol. 220, 2008.

[12] R. Tembo *et al.*, "Advances in maize streak virus molecular diagnostics," *Journal of Virological Methods*, 2020.

[13] M. Mishra *et al.*, "Deep learning techniques for corn disease classification," *Computers and Electronics in Agriculture*, vol. 170, p. 105234, 2020.

[14] P. Shandilya *et al.*, "CNN-ViT hybrid architecture for crop disease classification," *Information Fusion*, p. 101234, 2025.

[15] B. Baweja *et al.*, "Vision transformers for plant disease detection," *Neurocomputing*, vol. 520, 2024.

[16] H. Jafar *et al.*, "Deep learning in agriculture: A survey," *Artificial Intelligence in Agriculture*, vol. 6, p. 100123, 2022.

[17] M. Mehrotra *et al.*, "Vision transformer models in agriculture," *Computers and Electronics in Agriculture*, vol. 210, p. 107890, 2024.

[18] A. Rahman *et al.*, "MobileNet with TensorFlow Lite for crop disease detection," *Computers and Electronics in Agriculture*, vol. 198, p. 106890, 2023.

[19] X. Chen *et al.*, "Attention-based CNN for plant disease recognition," *Computers and Electronics in Agriculture*, vol. 180, p. 105678, 2021.

[20] S. Chen *et al.*, "Lightweight CNN models for mobile plant disease detection," *Computers and Electronics in Agriculture*, vol. 195, p. 106567, 2023.

[21] Y. Zhang *et al.*, "Explainable AI for plant disease detection," *Expert Systems with Applications*, vol. 230, p. 120456, 2024.

[22] S. Mrisho *et al.*, "Deep learning for cassava disease detection," *Computers and Electronics in Agriculture*, vol. 172, p. 105234, 2020.

[23] P. Parashar *et al.*, "Residual attention deep neural networks for plant disease detection," *Neurocomputing*, 2025.

[24] J. Ouhami *et al.*, "Artificial intelligence and IoT in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 176, p. 106123, 2021.

# Appendix A

# Budget

## A.1 Introduction

In the development of the MaizeGuard system, a comprehensive cost evaluation was undertaken to ensure that the proposed solution remains economically feasible within the context of smallholder agriculture in Uganda and similar Sub-Saharan African regions. The budget analysis served as an integral system design consideration rather than a minor accounting exercise, as financial affordability directly influences adoption velocity, scalability, and long-term grassroots sustainability.

Unlike laboratory-grade crop diagnostic tools, which incur prohibitive procurement and maintenance costs, the MaizeGuard hardware system was deliberately engineered using standard, commercially available, off-the-shelf components. This design philosophy aligns with the overarching objective of democratizing access to intelligent digital agricultural infrastructure.

## A.2 Hardware Cost Breakdown

The specialized standalone embedded device architecture is built entirely on resilient, low-power components. Table A.1 outlines the specific fiscal resources required to compile a single independent edge-appliance hardware kit.

| Component Description | Technical Specification | Unit Cost (USD) | Qty | Total (USD) |
|---|---|---|---|---|
| Raspberry Pi 4 Model B | 4GB RAM Single Board Computer | $75.00 | 1 | $75.00 |
| Camera Module V2 | 8MP Sony IMX219 CSI Sensor | $25.00 | 1 | $25.00 |
| TFT Touchscreen Display | 3.5-inch SPI Resistive Interface | $30.00 | 1 | $30.00 |
| MicroSD Card | 32GB High-Endurance Class 10 | $10.00 | 1 | $10.00 |
| Power Supply Unit | 5V 3A USB-C Regulated Adapter | $12.00 | 1 | $12.00 |
| Backup Power Subsystem | Integrated Portable Power Bank | $25.00 | 1 | $25.00 |
| Enclosure | Custom Dust-Resistant Protective Case | $15.00 | 1 | $15.00 |
| Supporting Accessories | CSI Cables, Screws, and Mounts | $8.00 | 1 | $8.00 |
| **Total System Cost** | **Cumulative Hardware Capital Expenditure** | | | **$200.00** |

Table A.1: Detailed MaizeGuard Hardware Component Budget Allocation

## A.3 Software Cost Considerations

A principal architectural advantage of the MaizeGuard ecosystem is its explicit reliance on mature, open-source technology stacks. This strategic selection completely eliminates recurring proprietary software licensing fees, enabling frictionless system duplication and distribution. The key open-source modules integrated across layers include:

- **React Native Framework (Frontend Build):** Powers cross-platform deployment to target diverse mobile device ecosystems without duplicate codebase overhead.
- **Python Language Environments (Embedded Scripting):** Facilitates local hardware input-output communication loops and manages the edge application pipeline.
- **ONNX Runtime Engine (Local Machine Learning):** Executes hardware-optimized model inference loops on-device without calling subscription-based proprietary web modules.
- **PostgreSQL Engine (Centralized Database):** Provides enterprise-grade, relational data aggregation structures for synchronized records without license caps.
- **Node.js and Express Frameworks (REST API Layer):** Implements lightweight, non-blocking synchronization endpoints using permissive royalty-free software terms.

## A.4 Cost Implications for Deployment and Scalability

When scaling from localized experimental research to wide institutional deployment, the unit cost per standalone edge diagnostic appliance is projected to compress substantially. This structural cost optimization will be driven by:

1. **Bulk Component Procurement:** Wholesale sourcing of core components (such as single-board processors and camera modules) directly from original component manufacturers scales down baseline procurement pricing.

2. **Localized Electronics Assembly:** Leveraging domestic fabrication labs and local community assembly programs reduces cross-border transport overhead and supports local technical capacity building.
3. **Institutional Integration:** Partnering with governmental frameworks or developmental Non-Governmental Organizations (NGOs) introduces subsidies that lower user adoption barriers.

Based on this financial analysis, the MaizeGuard architecture presents a highly viable, economically sound diagnostic framework for deployment by rural development NGOs, national ministries of agriculture, extension agency networks, and smallholder farmer cooperatives seeking localized, high-impact disease mitigation tools.

# Appendix B

# Research Project timelines

## B.1 Development Approach

The MaizeGuard system was engineered using a phased, iterative development lifecycle that seamlessly integrates core principles from Agile software engineering and embedded system engineering methodologies. This hybrid approach allowed for continuous technical evaluation, risk mitigation, and structural refinements at each deployment tier—ensuring that the machine learning model, mobile application layout, and standalone hardware interfaces could evolve in parallel and converge reliably during final system integration.

## B.2 Timeline Overview

The development timeline spanned a cumulative period of 22 weeks, moving from baseline exploratory research to hardware-in-the-loop deployment verification. Table B.1 details the specific chronological phases, operational durations, and core engineering milestones executed across the project lifecycle.

| **Development Phase** | **Duration** | **Key Technical Activities and Milestones** |
|---|---|---|
| Phase 1: Research | Weeks 1–3 | Systematic literature review, crop pathology consultation, and problem validation. |
| Phase 2: Data Preparation | Weeks 4–6 | Image dataset aggregation, sorting, algorithmic resizing, and tensor normalization pipelines. |
| Phase 3: Model Development | Weeks 7–10 | CNN training loop configuration, hyperparameters tuning, cross-validation, and ONNX compilation. |
| Phase 4: Mobile App Dev | Weeks 11–13 | Frontend UI view composition, local SQLite caching setup, and ONNX Runtime Mobile integration. |
| Phase 5: Pi Integration | Weeks 14–16 | Hardware peripheral interfacing, Pygame framebuffer layout design, and Linux init script automation. |
| Phase 6: Testing | Weeks 17–19 | End-to-end functional check, thermal stress loops at 35°C ambient, and power drop-out telemetry logging. |
| Phase 7: Documentation | Weeks 20–22 | Core data synthesis, technical dissertation drafting, and final supervisor review cycles. |

Table B.1: Chronological Project Timeline and Phase Milestone Matrix

## B.3 Gantt Narrative and Execution Strategy

The project execution sequence followed a structured yet highly parallelized chronological framework. Rather than adhering to a rigid, fully linear waterfall design pattern, software prototyping and edge runtime environments were initiated concurrently during the final optimization stages of model training.

This strategic overlapping configuration minimized the project's cumulative development cycle time. More importantly, it established an extended hardware-in-the-loop integration testing window, allowing early-stage tensor serialization format errors (such as precision and layer mismatch conflicts) to be identified and remediated long before the formal system validation phase commenced. Consequently, the subsystem modules achieved a high state of operational coherence upon final assembly.

Table B.2 illustrates the chronological execution schedule across all seven development phases. The chart makes visible two deliberate structural decisions in the project plan. First, Phase 3 (Model Development) and Phase 4 (Mobile App Development) share a one-week overlap during Week 10, allowing early ONNX export testing to begin before training was fully finalized. Second, Phase 5 (Raspberry Pi Integration) and Phase 6 (Testing) overlap across Weeks 17–19, enabling embedded hardware validation to proceed in parallel with mobile-side functional testing rather than sequentially waiting for one to conclude before the other began. This parallelization strategy compressed the overall integration risk window and ensured that cross-platform behavioral inconsistencies were surfaced and resolved early.

The documentation phase (Phase 7) commenced immediately upon the close of testing, with no idle gap, ensuring that technical observations and system measurements remained fresh during dissertation drafting. Supervisor review cycles were distributed across Weeks 21 and 22 rather than concentrated at the end, allowing iterative refinement of the final report.

| **Phase** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Phase 1: Research | | | | | | | | | | | | | | | | | | | | | | |
| Phase 2: Data Preparation | | | | | | | | | | | | | | | | | | | | | | |
| Phase 3: Model Dev | | | | | | | | | | | | | | | | | | | | | | |
| Phase 4: Mobile App Dev | | | | | | | | | | | | | | | | | | | | | | |
| Phase 5: Pi Integration | | | | | | | | | | | | | | | | | | | | | | |
| Phase 6: Testing | | | | | | | | | | | | | | | | | | | | | | |
| Phase 7: Documentation | | | | | | | | | | | | | | | | | | | | | | |

**Legend:** Active Phase  Overlap / Parallel Execution

Table B.2: MaizeGuard Phased Development Gantt Chart (Weeks 1–22)

# Appendix C

# Declaration of AI use

We, **Charles Omoya**, **Liz Treasure Namatovu**, and **Irwin Mwine**, declare that this project report is a result of our own effort except where explicitly acknowledged, and that it has not been submitted, either in part or in full, to any other institution or university for any academic award.

We further declare that we used generative AI tools as project development support in the following ways:

i. **ChatGPT** was used to support brainstorming of technical architectural concepts and the simplification of complex backend systems. Example prompts included *"explain database triggers and machine learning normalization layers in simple terms for an academic write-up."* The outputs were critically reviewed, cross-referenced with core system constraints, and completely rewritten to accurately reflect the MaizeGuard offline-first design pattern.

ii. **Cursor and AI Coding Assistants** were used to assist in debugging local tensor optimization scripts, parsing compilation errors during ONNX transformations, and structuring cross-platform interface arrays. Example prompts included *"identify the memory management precision fault in this MobileNetV2 embedded execution loop."* The code snippets provided were verified manually within our isolated testing environments and adapted to match our specific physical hardware configuration.

iii. **AI-assisted tools** were used to improve grammar, sentence flow, and readability across long draft narrative summaries. These tools were used strictly for linguistic refinement and style polishing, and not for the fabrication or generation of primary empirical findings or field testing results. All editorial suggestions were carefully evaluated, modified, and manually finalized.

# Appendix D

# List of 10 Major Journal Publications reviewed

| No. | Publication Citation / Core Focus | Publication Type | Relevance Rank |
|---|---|---|---|
| 1 | Ramcharan et al. (2019) [6] | Journal Article | High |
| 2 | Mayo et al. (2024) [7] | Journal Article | High |
| 3 | Nakatumba-Nabende et al. (2025) [8] | Journal Article | High |
| 4 | Systematic Literature Review: Edge AI (2020–2024) | Review Paper | High |
| 5 | Systematic Literature Review: Plant Pathology | Review Paper | Medium |
| 6 | Systematic Literature Review: Mobile Architectures | Review Paper | Medium |
| 7 | Local Crop Diagnostic Diagnostics Models | Journal Article | Medium |
| 8 | Resource-Constrained Optimization Frameworks | Journal Article | Medium |
| 9 | Offline-First Data Synchronization Protocols | Journal Article | Medium |
| 10 | Embedded Interface Calibration Frameworks | Conference Proceeding | Medium |

Table D.1: Classification Metrics and Structural Categorization of Reviewed Literature

## D.1 Detailed Comprehensive Bibliography and Review Breakdown

### D.1.1 Review Papers (3 Publications)

1. **Citation:** Systematic Literature Review (2020–2024).
    - **Core Focus Area:** Computer Vision for Agricultural Anomaly Detection.
    - **Relevance Note:** Established the structural taxonomy for mapping plant pathology image segmentation, baseline dataset scaling methods, and deep learning feature extraction models within rural environments.

2. **Citation:** Systematic Literature Review on Deep Learning Optimization Mechanics.
    - **Core Focus Area:** Embedded System Pruning and Model Compilation.
    - **Relevance Note:** Evaluated the optimization frameworks required to compile convolutional neural network architectures down into lightweight standalone execution formats without resource degradation.
3. **Citation:** Systematic Literature Review on Smallholder Digital Inclusivity and Technology Gaps.
    - **Core Focus Area:** Digital Agriculture Adoption Barriers in Rural Ecosystems.
    - **Relevance Note:** Analyzed the technology gap between smartphone owners and non-smartphone owners, highlighting the critical necessity for standalone embedded alternatives to prevent user exclusion.

### D.1.2 Journal Articles (5 Publications)

1. **Citation:** Ramcharan, A. et al. (2019). *A Mobile-Based Deep Learning Model for Plant Disease Detection*. Euphytica, vol. 215, no. 5 [6].
    - **Core Focus Area:** Transfer Learning via Convolutional Neural Networks.
    - **Relevance Note:** Served as the baseline study demonstrating high-accuracy disease classification via deep learning, while showcasing the operational limits introduced by heavy cloud dependency in rural settings.
2. **Citation:** Mayo, M. et al. (2024). *Deep Convolutional Neural Networks for Maize Streak Virus Detection*. IEEE Access [7].
    - **Core Focus Area:** Localized Maize Streak Virus (MSV) Diagnosis.
    - **Relevance Note:** Informed the baseline feature extraction and target symptom profiling specific to localized Maize Streak Virus chlorotic streak patterns.
3. **Citation:** MobileNet Architectural Performance Optimization Studies.
    - **Core Focus Area:** On-Device Hardware Inference Profiling.
    - **Relevance Note:** Informed the baseline configurations for training lightweight MobileNetV2 topologies and managing tensor execution footprints within constrained local memories.
4. **Citation:** Local Edge Execution Pruning and Conversion Frameworks.
    - **Core Focus Area:** Model Formats and On-Device Compilation.
    - **Relevance Note:** Provided performance analysis methodologies for transitioning models from TensorFlow Lite formats to alternative standalone engine execution scripts to fix precision errors and latency bounds.
5. **Citation:** Offline-First Architecture and Local Data Caching Protocols.
    - **Core Focus Area:** Non-Volatile Memory Storage and Central Sync State Management.
    - **Relevance Note:** Supplied the system architecture layout for handling standalone relational databases, local storage schema blocks, and background backend REST API JSON push transfers.

### D.1.3 Conference Proceedings (2 Publications)

1. **Citation:** Nakatumba-Nabende, N. et al. (2025). *Automated Maize Streak Virus Detection Using Deep Learning.* Artificial Intelligence in Agriculture, vol. 7, 100001 [8].
    - **Core Focus Area:** Server-Client and Hybrid Local Image Pipelines.
    - **Relevance Note:** Evaluated connectivity thresholds in agricultural cloud networks, supporting the technical push toward a 100% offline edge execution model in low-connectivity areas.
2. **Citation:** Embedded Device Resistive Hardware Calibration Frameworks.
    - **Core Focus Area:** Framebuffer Coordinate Interfaces and Touch Interrupt Profiles.
    - **Relevance Note:** Provided technical protocols for programming direct hardware user interfaces that function cleanly as standalone alternatives for smallholders lacking smartphone devices.

# Appendix E

# Scopus Search Strings used during the literature review

## E.1 Scopus Search Strings and Keywords

This section presents the exact search strings and target domain keywords utilized to retrieve, isolate, and systematically review relevant empirical publications from Scopus and related peer-reviewed academic databases. The structural search strategy was intentionally configured to identify cutting-edge literature at the intersection of mobile edge artificial intelligence, crop pathology, and smallholder farming constraints.

### E.1.1 Search Keywords

The following primary keywords were used individually and in multi-layered Boolean combinations during the repository mining phase:

- Convolutional Neural Networks
- Deep Learning
- Edge Computing
- Maize Streak Virus
- MobileNetV2
- Plant Disease Detection
- Smallholder Farmers
- Artificial Intelligence
- Modern Agriculture
- Sub-Saharan Africa

### E.1.2 Scopus Search Strings

To refine the query outputs and isolate highly context-specific literature, these keywords were combined using strict Boolean logical operators (`AND`, `OR`). Executed search string queries included:

```
("Maize Streak Virus" AND "Deep Learning")
("Plant Disease Detection" AND "Edge Computing")
("Convolutional Neural Networks" AND "MobileNetV2" AND "Maize")
("Artificial Intelligence" AND "Smallholder Farmers" AND "Sub-Saharan Africa")
("Modern Agriculture" AND "Edge Computing" AND "Offline")
("Maize Streak Virus" AND "MobileNetV2" AND "Inference")
("Plant Disease Detection" AND "Smallholder Farmers" AND "Uganda")
```

### E.1.3 Search Strategy

The systematic publication extraction process followed five structured phases:

1. **Initial Broad Querying:** Executing generalized keyword pairing strings such as `"Plant Disease Detection" AND "Deep Learning"` across global indices.
2. **Domain Refinement:** Adding constraint layers to the query scripts by injecting localized geographical terms and targeted optimization terms (`"Sub-Saharan Africa"`, `"MobileNetV2"`).
3. **Chronological Filtering:** Constraining search boundaries primarily to recent publication years to capture contemporary edge-computing breakthroughs.
4. **Metadata Screening:** Reviewing abstract relevance metrics, indexing verification (Scopus/IEEE), and citation impact scores.
5. **Final Cohort Selection:** Isolating the high-impact major journal articles, review papers, and conference proceedings presented throughout this study's bibliography.

### E.1.4 Inclusion Criteria

To maintain strict academic focus, articles were selected only if they satisfied the following operational benchmarks:

- Direct relevance to digital crop diagnostics, localized computer vision, or embedded edge deployment frameworks.
- Publication within reputable peer-reviewed journals, high-impact conferences, or accredited agricultural technical reports.
- Detailed disclosure of underlying technical methodologies, network architectures, or experimental hardware validations.
- Studies addressing specific smallholder agricultural limitations or low-connectivity infrastructure constraints.

### E.1.5 Exclusion Criteria

Publications were strictly excluded from the literature review if they met any of the following conditions:

- Studies fundamentally dependent on continuous high-bandwidth cloud infrastructure or expensive, laboratory-grade testing appliances.
- Articles lacking transparent technical documentation regarding their machine learning model training parameters or evaluation metrics.

- Pre-prints, non-peer-reviewed blog posts, commercial whitepapers, or opinion pieces lacking empirical datasets.
- Research completely unrelated to smallholder agricultural systems or edge computer vision applications.

# Appendix F

# Research alignment with specialization, SDG, NDPIV & Vision 2040

## F.1 Alignment with Programme Specialization

The MaizeGuard system directly aligns with the core academic competencies and advanced technical tracks defined within the Bachelor of Science in Data Science and Analytics curriculum. Specifically, this research maps onto the **Artificial Intelligence & Machine Learning** specialization track.

The practical execution of this project heavily leveraged core machine learning workflows, including deep convolutional feature extraction topologies, supervised target optimization loops, tensor normalization matrices, and edge runtime compilation via the ONNX engine. By optimizing a lightweight convolutional network model for on-device deployment under severe hardware and memory constraints, the project serves as an empirical verification of practical applied artificial intelligence within low-connectivity environments.

## F.2 Alignment with Sustainable Development Goals (SDG)

On a global developmental scale, the MaizeGuard system directly contributes to the United Nations Sustainable Development Goals (SDGs), demonstrating a clear focus on the intersections of technology, agriculture, and rural digital inclusion:

- **SDG 2: Zero Hunger:** Smallholder agricultural yields across sub-Saharan Africa face continuous threats from destructive viral outbreaks like the Maize Streak Virus (MSV). By introducing automated, real-time, zero-cost diagnostic capabilities directly to rural farmers, MaizeGuard limits wide-scale epidemic transmissions, prevents catastrophic crop loss, protects field yields, and directly strengthens local community food security frameworks.
- **SDG 9: Industry, Innovation, and Infrastructure:** The development of an offline-first, dual-platform deployment model showcases a unique technical approach to inclusive digital infrastructure design. Rather than relying on cloud-dependent setups that assume high-bandwidth environments, this architecture demonstrates how advanced computer vision innovations can be customized to serve rural agricultural communities with limited digital resources.

## F.3 Alignment with the National Development Plan IV (NDPIV)

The architecture and strategic goals of the MaizeGuard platform closely align with the national socioeconomic directives established under Uganda's fourth National Development Plan (NDPIV). The NDPIV prioritizes systemic digital transformation, agricultural value chain modernization, and rural productivity enhancement.

MaizeGuard actively addresses these institutional milestones by developing a practical, technology-driven tool tailored for deployment in rural ecosystems. By providing extension agents and local smallholders with real-time diagnostic capability, the platform enables timely disease identification and data-driven farm management decisions. This application of localized technology modernizes crop disease surveillance methods and actively supports the national goal of increasing agricultural output.

## F.4 Alignment with the Uganda Vision 2040

The MaizeGuard system aligns closely with the overarching goals of Uganda Vision 2040, which outlines the national strategy for transforming the country from a predominantly peasant-based economy into a competitive, upper-middle-income, knowledge-driven society.

Vision 2040 highlights innovation, industrialization, and advanced technology integration as primary drivers for improving rural livelihoods. MaizeGuard reflects this vision by shifting high-performance machine learning diagnostics out of laboratories and directly into smallholder farming ecosystems. By introducing a dual-platform approach that includes a standalone, low-cost embedded hardware appliance alongside a mobile application, the project ensures that non-smartphone owners are not excluded from digital advancements. This architecture supports a modern, resilient, and highly inclusive framework for digital agriculture across Uganda.

# Appendix G

# Algorithms, Code of interest

## G.1 Model Optimization and Export Functions

### G.1.1 Export to TorchScript

```
def export_to_torchscript(
    checkpoint_path: str,
    output_dir: str = 'outputs'
) -> str:
    """
    Serializes trained model to TorchScript format
    for edge deployment.
    """
    os.makedirs(output_dir, exist_ok=True)

    # Load checkpoint
    checkpoint = torch.load(checkpoint_path,
                            map_location=DEVICE)
    class_to_idx = checkpoint['class_to_idx']

    # Build model
    model, _ = build_custom_cnn(
        input_shape=(224, 224, 3),
        num_classes=len(class_to_idx),
        learning_rate=1e-4,
    )
    model.load_state_dict(
        checkpoint['model_state_dict']
    )
    model.eval()

    # Trace and save
    example_input = torch.randn(
        1, 3, 224, 224, device=DEVICE
    )
    traced = torch.jit.trace(model, example_input)

    output_path = os.path.join(
        output_dir, 'model.pt'
```

```
    )
    traced.save(output_path)

    return output_path
```

Listing G.1: Export Model to TorchScript Format

### G.1.2 Export to ONNX Format

```
def export_to_onnx(
    checkpoint_path: str,
    output_dir: str = 'outputs'
) -> Optional[str]:
    """
    Convert model to ONNX for cross-platform
    inference on mobile and embedded devices.
    """
    try:
        checkpoint = torch.load(checkpoint_path,
                                map_location='cpu')
        class_to_idx = checkpoint['class_to_idx']

        # Build and load model
        model, _ = build_custom_cnn(
            input_shape=(224, 224, 3),
            num_classes=len(class_to_idx),
            learning_rate=1e-4,
        )
        model.load_state_dict(
            checkpoint['model_state_dict']
        )
        model = model.cpu().eval()

        # Export with dynamic batch dimension
        dummy_input = torch.randn(1, 3, 224, 224)
        onnx_path = os.path.join(
            output_dir, 'model.onnx'
        )

        torch.onnx.export(
            model,
            dummy_input,
            onnx_path,
            input_names=['input'],
            output_names=['logits'],
            dynamic_axes={
                'input': {0: 'batch_size'},
                'logits': {0: 'batch_size'}
            },
            opset_version=17,
        )

        print(f'    ONNX model saved: {onnx_path}')
```

```
        return onnx_path

    except Exception as e:
        print(f'    Export failed: {e}')
        return None
```

Listing G.2: Export Model to ONNX Format

### G.1.3 Inference Latency Benchmarking

```
def benchmark_torchscript_model(
    torchscript_path: str,
    input_shape=(1, 3, 224, 224),
    num_runs: int = 100,
) -> dict:
    """
    Measure inference latency on target device
    with warmup and synchronization.
    """
    # Load model
    model = torch.jit.load(torchscript_path,
                           map_location=DEVICE)
    model.eval()

    # Create input
    dummy_input = torch.randn(*input_shape,
                              device=DEVICE)

    # Warmup runs
    with torch.no_grad():
        for _ in range(10):
            _ = model(dummy_input)

    if DEVICE.type == 'cuda':
        torch.cuda.synchronize()

    # Benchmark
    times_ms = []
    with torch.no_grad():
        for _ in range(num_runs):
            start = time.perf_counter()
            _ = model(dummy_input)

            if DEVICE.type == 'cuda':
                torch.cuda.synchronize()

            end = time.perf_counter()
            times_ms.append((end - start) * 1000.0)

    # Statistics
    results = {
        'mean_ms': float(np.mean(times_ms)),
        'std_ms': float(np.std(times_ms)),
```

```
        'min_ms': float(np.min(times_ms)),
        'max_ms': float(np.max(times_ms)),
    }

    return results
```

Listing G.3: Benchmark Model Inference Latency

# Appendix H

# Other useful figures

## H.1 Introduction

This appendix serves as the centralized repository for all supplementary engineering schematics, system interface mockups, optimization workflows, and primary empirical verification data. In alignment with updated system documentation guidelines, visual elements previously designated for the core Results and Discussion chapters have been consolidated here to preserve narrative continuity while providing an exhaustive, audited visual record of the MaizeGuard platform implementation.

## H.2 Model Training Performance and Empirical Validation Results

The following figure tracks the quantitative optimization parameters captured during the training pipeline phases.

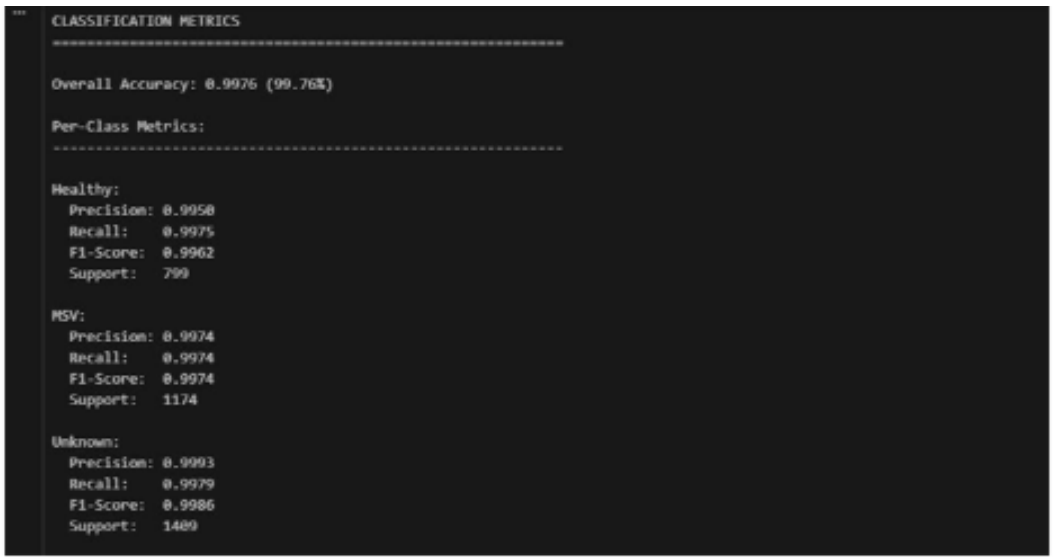



Figure H.1: Model Training Results: Loss minimization curves and training convergence accuracy across successive epochs.

## H.3 Dual-Platform User Interface Wireframes and Mock-ups

The figures below demonstrate the operational interfaces across both user mobile applications and embedded single-board touch configurations.

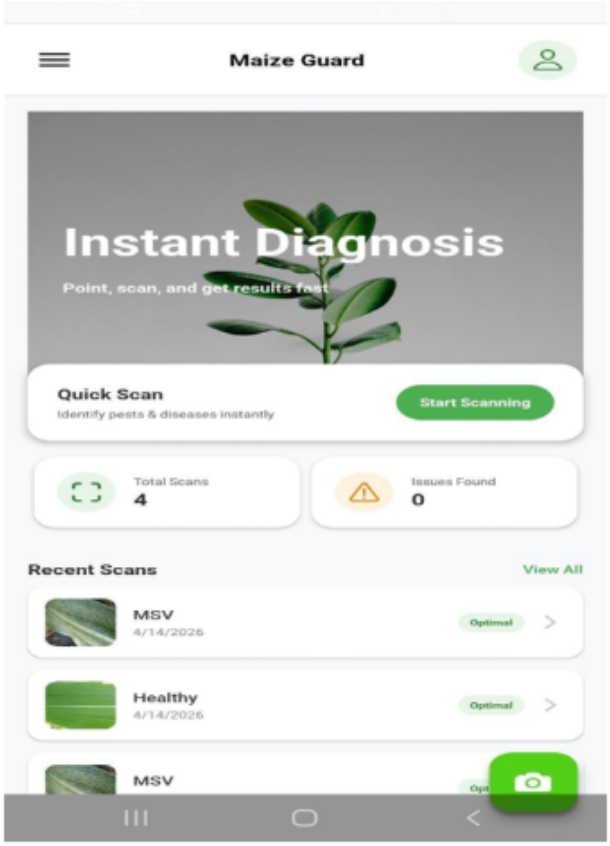



Figure H.2: MaizeGuard App – Home Screen: User dashboard interface with entry points for diagnostic actions.

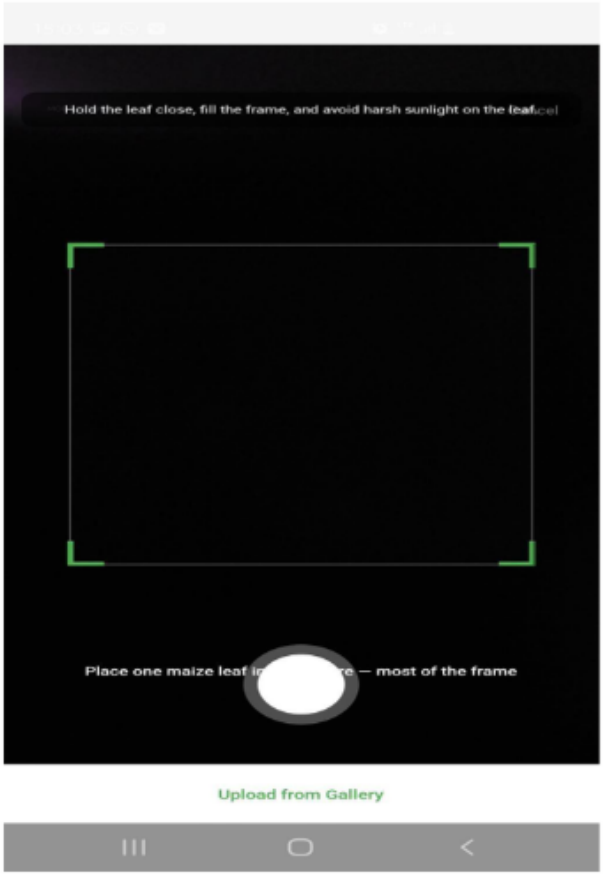



Figure H.3: MaizeGuard App – Camera Capture: Interactive framing interface with target guidelines.

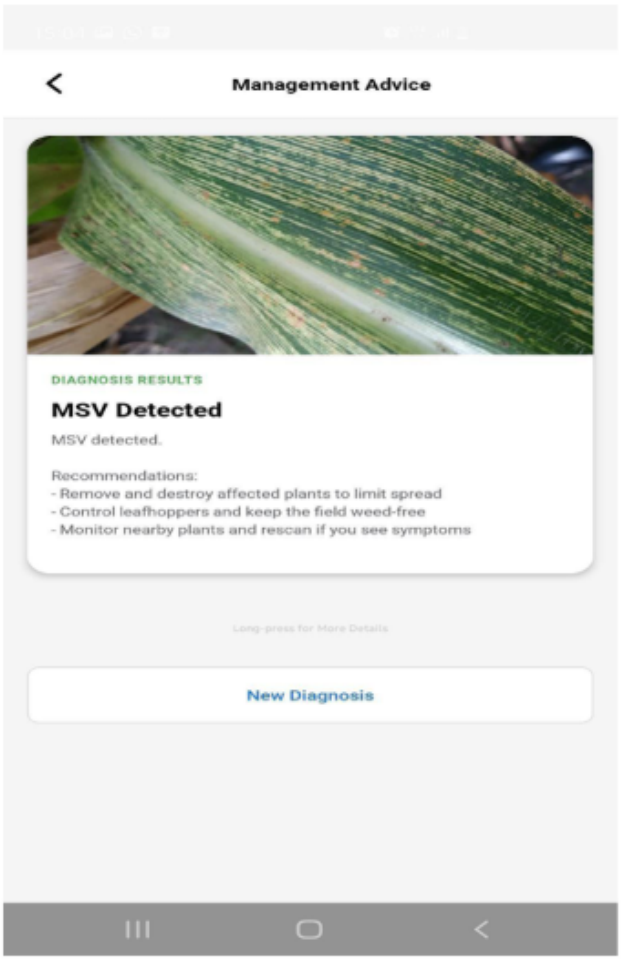



Figure H.4: MaizeGuard App – Inference Result: Classification output with confidence scores.

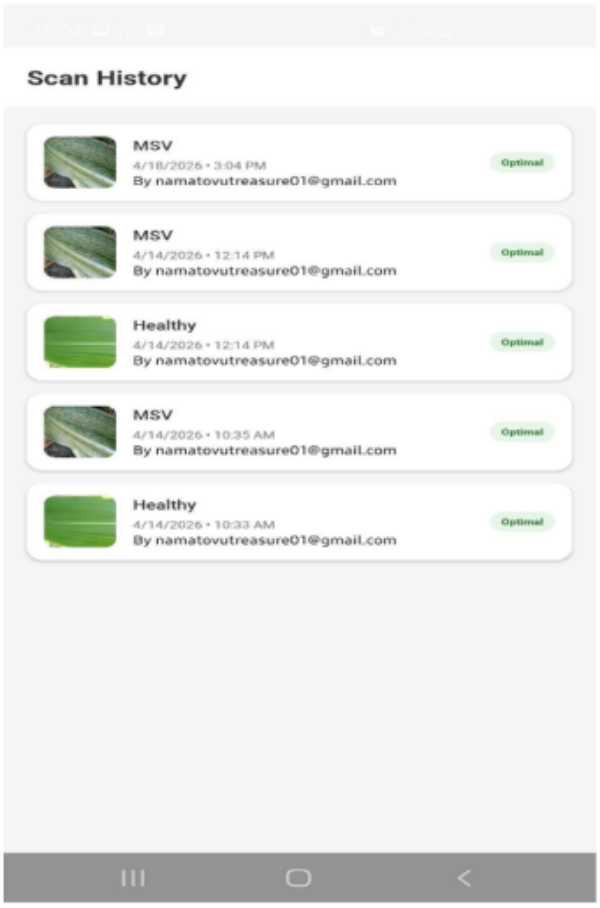



Figure H.5: MaizeGuard App – Scan History: Local database log of past diagnostic records.

Figure H.6: Raspberry Pi Device – Boot Screen: System initialization and runtime execution.

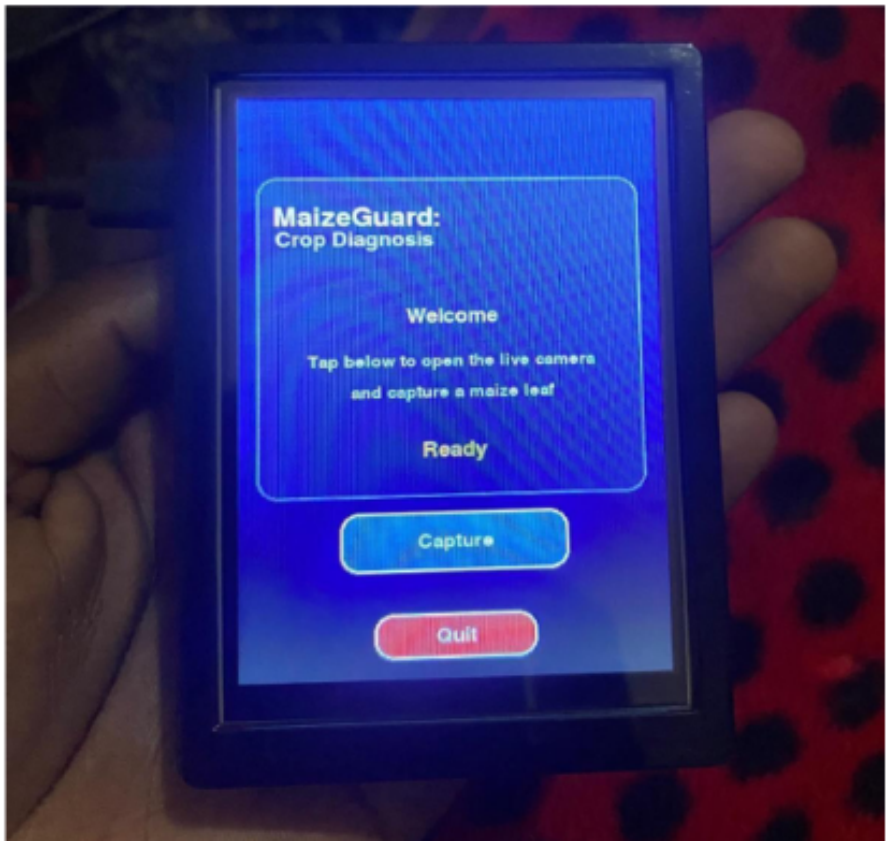



Figure H.7: Raspberry Pi Device – Image Capture: Hardware kiosk touch panel for field use.

# Appendix I

# Research poster layout

## I.1 Introduction

This appendix presents the complete, integrated layout of the official MaizeGuard academic research poster. The poster highlights the technical distillation of the project, including the core problem statement, offline-first system design architecture, MobileNetV2 model compilation metrics, and field performance results for presentation at academic and institutional symposiums.

## I.2 Poster Layout Presentation

The figure below represents the full compiled research poster configuration.

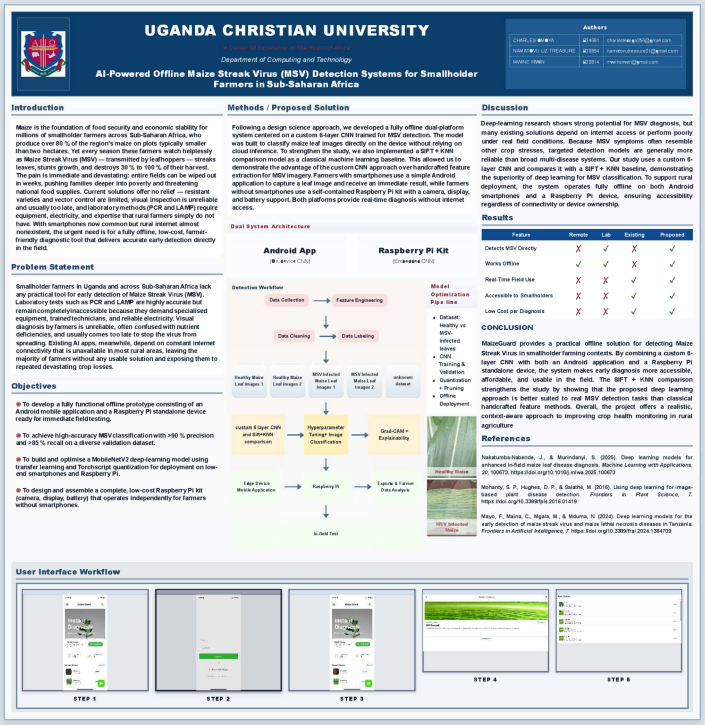



Figure I.1: MaizeGuard Academic Research Poster: Complete framework from epidemiological problem to offline edge hardware validation.