



Auckland University of Technology

School of Engineering, Computer and Mathematical Sciences

TinyML for Real-Time Crop Disease Detection in Precision Agriculture: A Proof-of-Concept Study on Maize Leaf Disease Detection

A thesis submitted in partial fulfilment of the requirements

for the degree of

Master of Engineering

Submitted by

Prama Nag

Student ID: 24268012

Supervised by

Professor Fakhrul Alam

10 July 2026

Acknowledgement

I want to sincerely thank everyone who has helped, inspired, and advised me in conducting this research.

First and foremost, I would like to sincerely thank my supervisor, Professor Fakhrul Alam, for his valuable advice, constructive feedback, enormous support, and constant encouragement during this research. His knowledge and guidance have been crucial to the accomplishment of my thesis.

I am particularly appreciative of the faculty and staff at Auckland University of Technology's (AUT) School of Engineering, Computer, and Mathematical Sciences for providing the resources required for this research along with an outstanding learning environment.

Lastly, I want to express my gratitude to my family and friends for their unwavering love, tolerance, support, and comprehension during my Master's journey. Throughout the difficulties and successes of my research, their unwavering support has been a source of courage and inspiration.

Overall, this research has been an incredibly fulfilling educational experience for me, and I genuinely thank everyone who helped make it happen.

Attestation of Authorship

I hereby declare that this submission is my own work and that, to the best of my knowledge and belief, it contains no material previously published or written by another person (except where explicitly defined in the acknowledgements), nor used artificial intelligence tools or generative artificial intelligence tools (unless it is clearly stated, and referenced, along with the purpose of use), nor material which to a substantial extent has been submitted for the award of any other degree or diploma of a university or other institution of higher learning.

Student Name: Prama Nag

Signature:

Date: 10 July 2026

ARTIFICIAL INTELLIGENCE DECLARATION TEMPLATE

Artificial Intelligence Guidelines:

Artificial Intelligence (AI) tools, including generative AI tools approved for use by AUT ([AI Tool Register](#)), can be used as learning tools when preparing to write your research proposal or research component. AI tools also have capability to assist with data modelling, data analysis and data visualisation. However, the research component you submit for examination must be substantively your own work. AI tools cannot solely be used to generate content when writing or creating an artwork/artefact, as this constitutes plagiarism. The formal guidelines for appropriate use of AI on the [AI Hub](#) must be followed. Also refer to the [Using Artificial Intelligence in your Research: Self-Assessments, Checklists, and Sample Text](#), for examples and sample text.

If you do plan to use AI tools, this declaration should be submitted with the initial research proposal, Confirmation of Candidature, and when you submit your research component for examination. You must complete the template below for each AI output used detailing the following:

- Chapters where AI was used in any way
- Purpose of AI Use
- AI Tool(s) Used
- Prompts Used
- Post-AI Processing Methods

Specify the chapter number(s) in your research proposal/research component where AI was used.	
Chapter number(s):	Chapters 1-5
Briefly explain why the AI tool was used (e.g., idea generation, summarization, data analysis, image creation, etc.).	
Purpose of AI Use:	Language editing, Grammar improvement, Improving academic writing and assistance with formatting latex document.
Clearly state the AI tools used.	
AI Tool(s) Used:	ChatGPT (OpenAI)
Prompts or queries entered into the AI tool (only the instructive portion of the prompt is required). Include all relevant variations.	
Prompts Used:	"Suggest improvement of this paragraph for academic writing."; "Correct grammar and improve clarity."; "Suggest wording for the research gap."; "Suggest improvement for the discussion section;"
A summary or description of the AI-generated output. (Optional: attach output in an appendix.)	
Output Received:	AI-generated suggestions for grammar, wording, sentence structure, explanations of concepts, and formatting advice.
Describe how the AI output was used, edited, or transformed.	
Post-AI Processing Methods:	All AI-generated text was carefully reviewed, verified, substantially edited, and integrated. Experimental design, implementation, data collection, analysis, model development, results, and conclusions were independently completed.

You may use this free text box to explain the intent for the use of the AI generated content. For example, rather than stating what prompt(s) were used, you may articulate the purpose for which the prompt was used (supported by examples) here:
AI tools were used primarily to improve the clarity, grammar, and academic style of the thesis, to obtain explanations of technical concepts, and to receive suggestions for structuring and refining sections of the document. All suggestions were critically evaluated and substantially revised before inclusion. The research design, implementation, data preprocessing, model development, experimentation, analysis, interpretation of results, and conclusions were independently conducted.

STUDENT DECLARATION		
By signing you are confirming that the AI use stated in the table(s) above are accurate and follow AUT's recommended guidelines detailed in the AI Hub and Postgraduate Handbook.		
Student Name:	Signature:	Date:
Prama Nag		13/07/26

Abstract

Traditional machine learning approaches for crop disease detection often require substantial computational resources, energy consumption, and network connectivity. Tiny Machine Learning (TinyML) addresses these limitations by enabling machine learning inference directly on resource-constrained edge devices, like microcontrollers. It is therefore well-suited for remote field applications with limited or nonexistent internet connectivity and cloud infrastructure. This study proposes a TinyML-based framework for real-time crop disease detection, demonstrated through a proof-of-concept study on binary maize leaf disease detection. Three lightweight deep learning models, namely a custom CNN, a custom MobileViT model, and a transfer learning-based MobileNetV1 model, were developed and evaluated on two publicly available maize leaf image datasets. To ensure a robust performance assessment, five-fold cross-validation was used, and model performance was evaluated using accuracy, precision, recall, F1-score, false positives, and false negatives. The trained models were subsequently optimised and converted to TensorFlow Lite format for deployment on Arduino Portenta H7. The comparative analysis showed that while MobileNetV1 demonstrated superior deployment efficiency with a latency of 71 ms and RAM of 125 KB, custom CNN and transformer-based MobileViT models achieved mean classification accuracy of 91.90% and 96.07%, comparable to, and in some cases exceeding, that of the transfer learning-based MobileNetV1 model. The results also demonstrate the trade-off between disease detection performance and deployment feasibility, highlighting the need to balance predictive accuracy with computational efficiency in TinyML applications. Using binary maize leaf disease detection as a proof-of-concept, this study demonstrates the feasibility of deploying lightweight deep learning models on resource-constrained edge devices for real-time crop disease detection in precision agriculture.

Contents

A	Introduction	7
A.1	Motivation of the Study	8
A.2	Aim of the study	9
A.3	Research Objectives	10
A.4	Research Questions	10
A.5	Scope of the Study	11
A.6	Contributions of the Study	12
A.7	Organisation of thesis	13
B	Literature Review	14
B.1	Precision Agriculture	15
B.2	Machine Learning in Agriculture	17
B.3	Machine Learning for Crop Disease Detection	18
B.4	Lightweight Deep Learning Architectures	21
B.5	Limitations of Traditional Machine Learning	25
B.6	TinyML for Edge Applications	26
B.7	Chapter Summary and Research Gap	31
C	Methodology	33
C.1	Datasets	33
C.2	Model Development Workflow	36
C.3	CNN Model	37
C.4	Transformer Based Architecture	40
C.5	Transfer Learning	41
C.6	Quantization	43
C.7	Deployment of TinyML Models	44
C.8	Development of the TinyML Model Using Edge Impulse	48
D	Results and Discussion	53
D.1	Cross-Validation Results	53
D.2	Deployment Results	61
D.3	Comparative Analysis of the Proposed Models	65
D.4	Validation with Edge Impulse	69

D.5 Overall Discussion and Research Implications	72
E Conclusion	81
F References	85

List of Figures

B.1	General Architecture of a Convolutional Neural Network (CNN)	21
C.1	Samples of Maize Leaf Images from Dataset1 (top row: healthy leaves; bottom row: diseased leaves)	34
C.2	Samples of Maize Leaf Images from Dataset2 (top row: healthy leaves; bottom row: diseased leaves)	35
C.3	Workflow of model development	36
C.4	Block Diagram of the CNN Binary Classification Model	38
C.5	Framework of the TinyML Deployment Process	45
C.6	Experimental deployment setup	47
C.7	Workflow of Edge Impulse Platform for Maize Leaf Disease Detection	49
C.8	CNN Model Architecture	50
D.1	Fold-wise Confusion Matrices of the CNN Model for Dataset1 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5	54
D.2	Fold-wise Confusion Matrices of the MobileViT Model for Dataset1: (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.	55
D.3	Fold-wise Confusion Matrices of the MobilenetV1 Model for Dataset1 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5	57
D.4	Fold-wise Confusion Matrices of the CNN Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5	58
D.5	Fold-wise Confusion Matrices of the MobileViT Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.	60
D.6	Fold-wise Confusion Matrices of the MobileNetV1 Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.	61

D.7 On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images using CNN model; top image: healthy maize leaf, bottom image: disease maize leaf	63
D.8 On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images using MobileNetV1 model; top image: healthy maize leaf, bottom image: disease maize leaf	64
D.9 Training vs Validation Accuracy and Loss	69
D.10 Confusion Matrix of the test dataset	70
D.11 On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images; top image: healthy maize leaf, bottom image: disease maize leaf	72
D.12 Examples of misclassification images; top row: healthy leaves misclassified as diseased, bottom row: diseased leaves misclassified as healthy	75

List of Tables

B.1	Technological Tools Enabling Machine Learning in Precision Agriculture	16
B.2	Machine learning-based related works on crop disease identification. Annotation: "-" indicates that the authors did not provide any information.	20
B.4	Applications of TinyML in smart farming. Annotation: "-" is used to signify that authors did not provide information. . .	28
C.1	Hyperparameter tuning range and selected optimal values . .	39
C.2	Hyperparameter tuning range and selected optimal values . .	41
C.3	Hyperparameter tuning range and selected optimal values . .	43
C.4	Candidate Microcontroller Platforms and Available Memory for TinyML Deployment	46
C.5	Hyperparameter tuning range	51
D.1	Fold-wise Performance Results	53
D.2	Fold-wise Performance Results	55
D.3	Fold-wise Performance Results	56
D.4	Fold-wise Performance Results	58
D.5	Fold-wise Performance Results	59
D.6	Fold-wise Performance Results	60
D.7	On-device Performance of Deployed Models on dataset1 . . .	62
D.8	On-device Performance of Deployed Models on dataset2 . . .	63
D.9	Performance Comparison of Three Architectures on Dataset1 .	66
D.10	Performance Comparison of Three Architectures on Dataset2 .	67
D.11	Performance metrics of the model evaluated in Edge Impulse	71
D.12	On-device Performance of Deployed Model	71
D.13	Comparison of Full-Sized (FP32) and INT8-quantised Model Accuracy and Statistical Significance for Dataset1	77
D.14	Comparison of Full-Sized (FP32) and INT8-quantised Model Accuracy and Statistical Significance for Dataset2	77

D.15 Comparison of Full-Sized and INT8-Quantised Model Sizes for Dataset1	78
D.16 Comparison of Full-Sized and INT8-Quantised Model Sizes for Dataset2	78

A. Introduction

Agriculture is one of the oldest and most important industries in the world. The rapid growth of the global population increases the demand for employment and food [1]. Climate change and the growing world population will put more pressure on agriculture by 2050. Pests, weeds, and diseases currently account for 42 percent of food crop losses, drastically lowering productivity and quality. Inappropriate technology often yields substandard results, even with significant expenditures on pest control [2]. On the other hand, Human activity and shifting lifestyles are causing farmlands to deteriorate, which could result in decreased food yields, starvation, and famine. Significant obstacles to agriculture and food security include dwindling soil fertility, decreased crop yields, ineffective fertilisers and pesticides, and scarce water supplies due to misuse and inadequate resource management [3].

An important strategy for promoting agricultural development is precision agriculture, which has recently drawn much attention. The International Society of Precision Agriculture (ISPA) defines precision agriculture as the use of spatial, temporal, and individual data to support informed decision-making that improves agricultural productivity, efficiency, quality, and sustainability [4]. This strategy enables farmers to utilise machinery, human, and natural resources more efficiently while minimising environmental impacts. Precision agriculture offers an effective approach to increasing food production, improving crop quality, and reducing environmental impacts as the global population grows and arable land declines due to urbanisation and climate change [5].

Plant diseases affect food safety and the quantity and quality of agricultural goods. These consequences lead to a decline in revenue in the production sectors, which is especially important for developing nations [6]. Since infections can cause significant yield losses in important crops such as rice, wheat, and maize, effective detection of plant diseases is crucial to maintaining food security. The inefficiency of traditional inspection methods for large-scale agriculture emphasises the importance of implementing preci-

sion agriculture techniques. To automate disease identification using field image analysis and enable precise and fast crop health monitoring, machine learning integration has grown in importance [7].

Machine learning's rapid advancement has completely transformed modern agriculture by promoting sophisticated data-driven methods for detecting crop health, increasing productivity, and promoting sustainable farming practices [8]. Machine learning has significantly increased the effectiveness and precision of decision-making procedures that previously relied on expert knowledge and manual observation by automatically identifying significant patterns from massive amounts of agricultural data. As a result, many agricultural applications have effectively used machine learning. Because early and precise disease identification enables timely intervention, reduces crop losses, and improves food security, image-based agricultural disease classification has attracted significant attention [9].

Conventional machine learning and deep learning approaches have several practical drawbacks in actual farming settings, despite their effectiveness in agricultural image analysis. Many cutting-edge models are not suitable for direct implementation in low-power embedded devices because they are computationally demanding and require significant processing power, memory, and storage resources [10]. As a result, these models frequently rely on cloud-based processing, which limits reliable operation in remote locations with limited or poor internet connectivity and adds communication latency and energy consumption. These drawbacks emphasise the need for fast and lightweight deep learning models that can perform real-time inference on edge devices with limited resources.

A.1 Motivation of the Study

Crop diseases have become a major challenge in modern agriculture, reducing crop yields, compromising crop quality, and threatening global food security. Conventional plant disease diagnosis techniques largely rely on manual inspection, which is often labor-intensive, time-consuming, unfeasible for large-scale agricultural sectors, and prone to human error and fa-

tigue [11]. Recent developments in computer vision and deep learning have shown great potential for automated plant disease identification using image-based classification methods. However, the application of deep learning models in real-time agricultural monitoring systems and resource-constrained environments is limited due to their high computational demands, significant memory usage, and cloud-based processing [12].

One promising approach for implementing lightweight deep learning models on low-power embedded systems is tiny machine learning, or TinyML. TinyML may reduce latency, optimise energy consumption, and eliminate the need for constant internet access by enabling real-time inference directly on edge devices. TinyML is ideal for precision agriculture applications because of these features, especially for real-time crop health monitoring in remote farming settings [13].

A.2 Aim of the study

The aim of this study is to develop and evaluate lightweight deep learning models for TinyML-based maize leaf disease detection and to investigate their suitability for deployment on resource-constrained edge devices. The detection of disease in maize leaf images is used as a proof-of-concept to demonstrate the feasibility of the proposed framework. Specifically, the study seeks to evaluate the classification performance of custom CNN, MobileViT, and MobileNetV1 models for binary maize leaf disease detection using conventional evaluation metrics, including accuracy, precision, recall, F1-score, false positive rate, and false negative rate. In addition, the study compares the deployment performance of these models on a resource-constrained edge device in terms of inference time, RAM usage, and flash memory consumption. Through this proof-of-concept case study, the research evaluates the predictive performance and deployment feasibility of lightweight deep learning models under the computational and memory constraints of edge devices, demonstrating their practical applicability for real-time crop disease detection in precision agriculture.

A.3 Research Objectives

To achieve the aim of this study, the following objectives are addressed:

1. Review existing machine learning, deep learning, and TinyML techniques for crop disease detection in precision agriculture.
2. Design and develop a lightweight CNN architecture and a transformer based MobileVit architecture specifically optimised for TinyML deployment on resource constrained embedded devices.
3. Develop a transfer learning model based on MobileNetV1 for binary maize leaf classification.
4. Compare custom CNN, MobileViT and MobileNetV1 models using standard classification metrics, including accuracy, precision, recall, F1-score, false positives, and false negatives .
5. Evaluate the deployment performance of the developed models on a TinyML-enabled edge device by analysing model size, inference time, and resource utilisation.
6. Verify the robustness and deployment viability of the proposed TinyML framework using the Edge Impulse development platform.
7. Explore the trade-offs between deployment viability and classification performance of lightweight deep learning models for the binary classification of maize leaf.
8. Identify the practical feasibility of lightweight deep learning models for the real-time diagnosis of maize leaf disease in actual agricultural settings.

A.4 Research Questions

This study seeks to answer the following research questions:

1. How effectively do lightweight deep learning models perform in binary maize leaf disease detection in terms of accuracy, precision, recall, F1-score, false-positive rate, and false-negative rate?

2. How do the proposed custom CNN and MobileViT models compare with the transfer learning-based MobileNetV1 model in terms of classification performance and deployment feasibility for maize leaf disease detection?
3. What trade-offs exist between classification performance and deployment constraints, including model size, inference time, and computational efficiency, when implementing these lightweight models on resource-constrained edge devices?

A.5 Scope of the Study

This study focuses on the development, evaluation, and deployment of lightweight deep learning models for TinyML-based crop disease detection in precision agriculture. Binary maize leaf disease detection, formulated as a healthy-versus-diseased image classification problem, is used as a proof-of-concept to evaluate the proposed framework. The scope of this study is limited to binary disease detection and does not consider multi-class disease classification, disease severity estimation, object detection, or field-based disease localisation. Furthermore, the study does not involve the collection of new image data or the integration of a camera with the edge device.

Three lightweight deep learning models are investigated: a custom CNN and a custom transformer-based MobileViT model, both designed for TinyML deployment and trained from scratch, and a transfer learning-based MobileNetV1 model. Two publicly available maize leaf image datasets are used in the study. Each dataset is partitioned into training, validation, and test sets for model development and evaluation. The training and validation sets are used for model training and hyperparameter tuning, while the unseen test set is used to evaluate the classification performance and generalization capability of the trained models.

The models are evaluated using standard performance metrics, including accuracy, precision, recall, F1-score, false positives, and false negatives. Their suitability for TinyML deployment is further assessed by comparing

model size, inference time, RAM usage, and flash memory consumption following deployment on an Arduino Portenta H7 edge device. Inference on the edge device is performed using the same unseen test images employed during model evaluation, allowing the deployed models' inference performance to be assessed under embedded execution conditions.

In addition to the Python-based implementation, this study investigates the Edge Impulse platform as an alternative TinyML development environment. Using Edge Impulse, a lightweight optimized CNN model is developed and deployed on the Arduino Portenta H7 to assess its performance. The Edge Impulse implementation is limited to Dataset1 and serves as an independent validation of the proposed TinyML framework using a different development and deployment workflow. All experiments are based exclusively on the two publicly available maize leaf image datasets, with the Python-based implementation evaluated on both datasets and the Edge Impulse implementation evaluated on Dataset1 only.

A.6 Contributions of the Study

The main contributions of this study are summarized as follows:

1. A TinyML framework for real-time crop disease detection is proposed and demonstrated using binary maize leaf disease detection as a proof-of-concept.
2. Two custom lightweight deep learning models, a CNN and a transformer-based MobileViT model, are developed specifically for TinyML deployment and compared with a transfer learning-based MobileNetV1 model.
3. The trade-offs between disease detection performance and deployment efficiency are systematically analysed using classification metrics together with deployment-related metrics, including inference time, memory usage, and model size.
4. The proposed models are successfully deployed and evaluated on an Arduino Portenta H7 edge device, demonstrating the practical feasi-

bility of real-time TinyML-based crop disease detection on resource-constrained hardware.

5. The proposed TinyML framework is independently validated using the Edge Impulse platform, demonstrating its applicability across different TinyML development and deployment workflows.

A.7 Organisation of thesis

Chapter 1 introduces the research background, motivation, research aim, objectives, research questions, scope, contributions, and overall thesis structure.

Chapter 2 reviews the literature on precision agriculture, machine learning, and deep learning for crop disease detection, TinyML, model optimization techniques for TinyML, and TinyML development platforms for edge deployment. The chapter critically examines current research challenges and best practices, identifies gaps in the existing literature, and establishes the motivation and the methodological foundation based on the literature for this study.

Chapter 3 presents the research methodology, including dataset preparation, the design of the custom CNN and transformer based MobileViT architecture, the implementation of the MobileNetV1 transfer learning model, model training procedures, evaluation metrics, TinyML deployment and validation process using Edge Impulse.

Chapter 4 presents the experimental results, compares the performance of the developed models, discusses their suitability for deployment on resource-constrained edge devices and practical implications of this study.

Chapter 5 concludes the thesis by summarising the main findings, outlining the contributions of the research, discussing its limitations, and providing recommendations for future work.

B. Literature Review

Precision agriculture has emerged as an important approach to improving agricultural productivity, sustainability, and resource efficiency through the use of advanced digital technologies. The integration of artificial intelligence (AI), the Internet of Things (IoT), machine learning (ML), and remote sensing has enabled data-driven decision-making for crop monitoring and management. More recently, TinyML has extended these capabilities to resource-constrained edge devices by enabling lightweight machine learning models to execute on low-power microcontrollers. This allows real-time crop health monitoring directly in the field, reducing reliance on cloud computing while improving response time and operational efficiency [14].

This chapter reviews the literature relevant to the development and deployment of TinyML-based crop disease detection systems for precision agriculture. It begins by introducing precision agriculture and the role of machine learning (ML) and deep learning (DL) in automated crop health monitoring, before examining the principles and applications of TinyML for resource-constrained edge devices. The chapter then reviews lightweight deep learning models, model optimisation techniques, and TinyML development platforms that enable the deployment of deep learning models on embedded systems.

In addition to summarising the current state of the art, the chapter critically analyses existing research on image-based crop disease detection, with particular emphasis on model accuracy, computational efficiency, inference time, and memory requirements, which are key considerations for edge deployment. It also identifies the limitations of current research, including the limited deployment and evaluation of TinyML models on real embedded hardware for agricultural applications.

The chapter concludes by synthesising the findings from the literature, identifying the research gap addressed by this thesis, and establishing the literature-informed methodological foundation for the development, opti-

misation, and deployment of lightweight deep learning models for real-time disease detection of maize leaf images on an edge device.

B.1 Precision Agriculture

Farmers in many countries continue to rely on traditional farming practices informed by generations of experience. While these practices have historically been effective, they are increasingly challenged by the growing variability and unpredictability of weather patterns. Spraying pesticides by hand frequently results in resource waste and environmental damage [15]. Precision agriculture is a data-driven strategy that monitors and controls field variability using tools such as sensors, drones, GPS, and machine learning. Developments in AI and machine learning have accelerated agricultural development, expanding its application beyond yield prediction and crop monitoring to a variety of agricultural uses [16]. Today, Machine learning is becoming more important in smart farming to improve agricultural choices. Regression methods predict the amount of water needed, while classification algorithms help evaluate crop suitability depending on the soil type. supervised and unsupervised learning techniques help to efficient crop management and selection [17].

To address the effects of climate change, such as changes in temperature and precipitation that impact crop yields, machine learning is becoming more important in agriculture. Farmers can gather sophisticated data stored on cloud platforms by combining IoT devices, such as drones, sensors, and cameras. Using IoT analytics and machine learning to analyse these data, farmers can make better and more educated decisions that improve food quality, yield, and crop failure rates [18].

B.1.1 Technologies Enabling Precision Agriculture

Table B.1 illustrates how cutting-edge technical instruments can improve machine learning-based agricultural development decision-making. Through the organisation and analysis of data from technologies like GPS, GIS, IoT

devices, sensors, smart phones, and drones, machine learning plays a critical role in smart farming.

Table B.1: Technological Tools Enabling Machine Learning in Precision Agriculture

Tools	Role in Agriculture	Relation to Machine Learning
GPS (Global Positioning System) [19]	Supplies precise location information for variable-rate applications, equipment tracking, and field mapping.	ML optimises planting and input distribution by using GPS data for geographic modeling.
GIS (Geographic Information System) [19]	Keeps and examines spatial data, such as yield maps, crop zones, and soil types.	ML makes use of GIS data to identify geographical trends and forecast events such as disease outbreaks or yield.
Machinery Tools (e.g., smart tractors, harvesters) [20]	Use embedded sensors to automate harvesting, spraying, and planting.	ML analyses machinery performance data to improve operational efficiency and reduce fuel/maintenance costs.
IoT Analytics (e.g., environmental and soil sensors) [21]	Gathers data on temperature, moisture content, pH, and other variables in real time.	ML models process this data to forecast irrigation needs, detect stress, and guide fertilization.
Artificial Intelligence (AI) [21]	Broad term encompassing ML, used for decision-making and intelligent automation.	ML is a branch of AI that uses agricultural data to create classification and prediction models
Aerial Vehicles (Drones/UAVs) [2]	Broad phrase that includes machine learning and is applied to intelligent automation and decision-making.	ML is a branch of AI that uses agricultural data to create classification and prediction models.
Sensors (soil, crop, climate, machinery) [22].	Give accurate, localised measurements of the plant and ambient conditions.	ML makes use of sensor data for resource optimisation, anomaly detection, and real-time monitoring.
Smart phones [23].	Devices that are portable for downloading data, accessing agricultural apps, and taking field images.	Smartphone machine learning apps provide real-time advising services, yield calculation, and on-the-spot disease diagnostics.

To facilitate well-informed decision-making, it transforms vast amounts of real-time data—such as weather patterns, crop health, and soil conditions—into meaningful patterns. By learning from this structured data, machine learning algorithms are able to identify diseases, forecast yields, and maximise resource utilisation. Consequently, farming becomes more productive, efficient, and ecologically friendly.

B.2 Machine Learning in Agriculture

An area of artificial intelligence (AI) that has shown potential for solving a number of agricultural problems is machine learning (ML). Machine learning provides effective and data-driven solutions for agricultural decision-making by enabling computer systems to recognise patterns in data and generate predictions without explicit programming [24]. The incorporation of machine learning into contemporary farming methods has been accelerated by the availability of agricultural datasets, developments in sensing technology, and improvements in processing power [25].

Many agricultural fields have seen the successful application of machine learning. Crop disease detection, pest detection, weed classification, crop yield prediction, soil quality evaluation, irrigation management, and environmental monitoring are some of these applications. Because it directly affects crop productivity and food security, crop disease detection has drawn much attention among these applications. Early and precise identification of plant diseases avoids the overuse of pesticides, allows prompt intervention, and reduces financial losses [26].

The capacity of machine learning to detect complex patterns and features that can be difficult to spot with conventional inspection techniques is a major factor in its efficacy in agricultural image analysis [27]. Machine learning models can automatically classify crops, detect diseases, and improve precision agriculture by analysing images captured by cameras, drones, smartphones, or agricultural sensors. This skill has aided in the creation of intelligent agricultural systems that reduce operating costs while increasing productivity [28].

Due to the effective processing of large amounts of image data by machine learning algorithms, crop health can be continuously monitored across vast agricultural areas. Machine learning-based solutions offer consistent, objective, and scalable analysis compared to traditional visual examination, which is often labor-intensive, subjective, and highly dependent on human expertise. These technologies enable farmers to take prompt action before major harvest damage by identifying minor visual signs of disease, nutrient shortages, and pest infestations. In addition to increasing crop quality and productivity, early diagnosis reduces excessive use of pesticides, saving production costs and eliminating adverse environmental effects [29].

Additionally, the adoption of smart agricultural practices has been accelerated by the integration of machine learning with cutting-edge sensor technology and Internet of Things (IoT) devices. Continuous field monitoring and well-informed decision-making in precision agriculture are made possible by real-time image capturing and automated analysis [30]. It is anticipated that machine learning will become crucial in creating intelligent agricultural systems that are more productive, sustainable, and can satisfy the expanding worldwide demand for food production as computational methods and image processing technologies improve [31].

B.3 Machine Learning for Crop Disease Detection

Machine learning is becoming a significant factor in the growth of agriculture because technology makes it possible to automate and optimise complex farming processes through data-driven insights. ML assists farmers in making well-informed decisions that improve production, efficiency, and sustainability by evaluating a wide range of agricultural data [32]. In the long run, it lowers expenses and resource waste by promoting early disease diagnosis, precise fertilisation and irrigation, and precise yield forecasts. With population expansion, resource restrictions, and climate change posing increasing difficulties to agriculture, machine learning (ML) is emerging as a potent tool to spur innovation and guarantee food security [33].

Intelligent decision-making is made possible by machine learning, which

enhances the capabilities of IoT devices. However, it is challenging to execute complex models, such as deep learning, on most IoT devices due to their limited memory, processing capacity, and energy constraints. Because of this, they frequently rely on cloud computing, which adds latency, raises communication costs, and poses privacy issues because of data transfer. These drawbacks emphasize the necessity of energy-efficient, lightweight machine-learning solutions that can operate directly on edge devices [34].

Many machine learning and deep learning models have been investigated in recent crop disease detection research; each has unique performance measures and constraints. A comparative overview of a few chosen machine learning methods is shown in Table B.2.

CNN models, such as the one presented by [35], achieve excellent accuracy. However, due to their relatively large memory footprint and processing demands, they are unsuitable for deployment on devices with limited resources. Similar difficulties include high memory consumption, lengthy inference times, and higher computational complexity for Random Forest [36] and SVM-based models [37]. Deep networks like ICS-ResNet [38] or MaizeNet [39] and hybrid models that combine SVM and DenseNet [40] make deployment in embedded environments even more challenging because of their high energy requirements and large model sizes (more than 7 to 25 million parameters). Despite being extremely accurate, architecture like Attention U-Net [41] requires substantial computing power. However, with fewer than a million parameters, [42] offers a lighter method that shows promise for on-device deployment. These drawbacks highlight the necessity of TinyML, which enables real-time, energy-efficient inference by compressing models and modifying them to function well on low-power microcontrollers.

Table B.2: Machine learning-based related works on crop disease identification. Annotation: "- -" indicates that the authors did not provide any information.

Method/Model Name	Crops	Datasets	Accuracy (%)	Inference Time	Parameters	Limitations
CNN [35]	Maize	Available on the Kaggle platform	94.97%	1.22 s	1.22 million	Unsuitable for resource-constrained devices
Random Forest (HOG, PHOG, GLCM) [36]	Tomato	Internet and Plant-Village color dataset	93.12%	Modest	- -	High memory footprint
SVM (GLCM) [37]	Tomato	Captured using Android phone camera	90%	- -	Not specified	Requires larger inference time, Memory Constraints
Hybrid Model (SVM & DenseNet201) [38]	Corn (Maize), Rice, and Wheat	Plant Village And Kaggle	87.23%	6.7 ms	20.2 million	Large Model Size (20.2 million parameters)
ICS-ResNet (based on ResNet50) [39]	Maize	Kaggle platform and Plant Village dataset	98.87%	- -	7.2 million	Still too large for microcontrollers
MaizeNet Model (based on Faster-RCNN and ResNet-50) [40]	Maize	Corn image dataset captured via mobile devices under diverse conditions	97.89%	- -	25.6 million	High Inference Latency, High Computational and Energy Demand
LFMNet [41]	Maize	Open-source crop disease datasets	94.12%	- -	0.88 million	Limited Fit for Microcontrollers
Attention U-Net model [42]	Maize	Images captured using a handheld set, the boom set, and the drone set	98.97%	- -		High Computational Complexity, Large Model Size

B.4 Lightweight Deep Learning Architectures

B.4.1 Convolutional Neural Networks

Convolutional Neural Networks (CNNs) are deep learning models designed to analyse grid-like data, such as images. They can learn useful patterns from images through a series of specialised layers, making them well-suited for tasks such as image sorting and visual recognition. The convolutional layer, which applies learnable filters to the input image to extract local characteristics such as edges, textures, and forms, is the core part of a CNN. These filters create feature maps that maintain spatial links within the data by sliding across the input [43].

CNNs regularly employ pooling layers, including max pooling, to increase resilience and decrease computing complexity. By summarising local areas, pooling layers down-sample the feature maps, reducing the spatial dimensions without sacrificing the most crucial information. This enhances translation invariance and improves the model's suitability for deployment on platforms with limited resources, like TinyML devices [44].

The obtained feature maps are flattened and sent to fully connected layers following several convolutional and pooling phases. By merging the learnt features to create final predictions, these layers carry out high-level reasoning. The fully connected layers map the retrieved features to output classes in classification tasks.

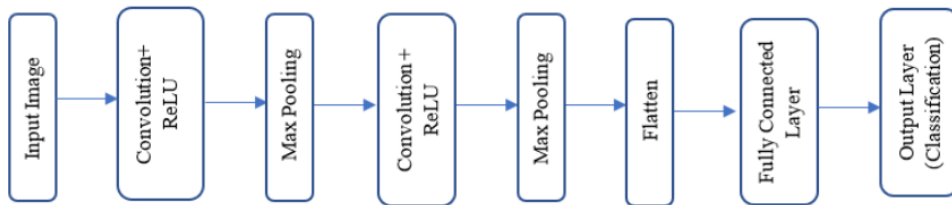


Figure B.1: General Architecture of a Convolutional Neural Network (CNN)

The basic process of a Convolutional Neural Network (CNN) for image classification is shown in Figure B.1. To find important spatial features, the input image is first passed through convolutional layers and activation

functions. Pooling layers then reduce the image size but keep key details. Finally, the features are flattened and sent through fully connected layers to get the classification result.

For crop disease detection, CNNs are trained on labeled images of healthy and diseased plant leaves to learn discriminative visual characteristics. CNNs automatically generate hierarchical feature representations without depending on manually created descriptors, moving from low-level features such as edges, textures, and color variations to higher-level semantic features like vein discoloration, lesion morphology, and disease-specific patterns [45]. When trained on a variety of datasets, CNNs' capacity to generalise intricate, non-linear correlations between image features and disease categories is another important benefit. Because CNN-based models offer improved accuracy and more resilience without requiring handcrafted feature extraction, they have consistently outperformed several classic machine learning methods in crop disease classification [46].

B.4.2 Transformer-Based Architectures

Transformer-based models have become a potent deep learning technique for computer vision challenges by successfully capturing long-range dependencies and global contextual information within images. These architectures use a self-attention mechanism that enables the model to learn relationships between image regions regardless of their spatial distance. Significant advancements in a number of image classification and identification tasks have resulted from this capacity [47].

Transformer-based models in image analysis commonly use feed-forward networks and multi-head self-attention to process feature maps after splitting them into patches or tokens. To maintain spatial information that might otherwise be lost during tokenization, positional embeddings are used. Transformer based designs offer improved discriminative power for challenging visual identification tasks by combining both local and global feature representations [48].

These models have recently received significant interest in agricultural

image analysis, especially crop disease identification, due to their robust feature representation. They enable the classification of healthy and diseased plant leaves by capturing small differences in color, texture, and structural patterns. As a result, these designs have emerged as a viable method, especially for applications involving intelligent precision agriculture [49].

B.4.3 MobileViT

MobileViT is a hybrid architecture that combines the global feature modeling capabilities of Vision Transformers (ViTs) with the local feature extraction capabilities of Convolutional Neural Networks (CNNs). MobileViT maintains computational efficiency while capturing both spatially local and long-range feature interdependence by including transformer blocks into a lightweight convolutional framework [50].

Standard convolutional layers form the MobileViT architecture, followed by MobileViT blocks that process features using transformers. While the transformer blocks enable the model to learn global contextual associations across many parts of the input image, the convolutional layers retrieve low-level image information. This combination maintains a small model size appropriate for deployment on edge devices with limited resources while enhancing feature representation and classification performance [51].

To classify crop diseases, the transformer blocks replicate global dependencies across different leaf regions, while the convolutional layers initially gather local visual data, such as lesion boundaries, texture variations, and color variations. MobileViT can increase classification robustness under different imaging settings and successfully identify diseases with similar visual symptoms by merging local and global feature representations. Additionally, its lightweight architecture enables effective deployment on embedded platforms with constrained memory, processing capacity, and energy consumption [52].

B.4.4 Transfer Learning

Transfer learning is a deep learning method that applies knowledge learned from a model trained on a significant source dataset to a relevant target task. Transfer learning reduces training time and the need for a lot of labeled data by using pre-trained feature representations learnt from large datasets like ImageNet, rather than developing a neural network from scratch [53]. Because of these benefits, transfer learning has gained acceptance in computer vision applications, especially in situations when task-specific datasets are limited.

Transfer learning has proven quite successful in identifying and classifying crop diseases in agricultural image analysis. To classify healthy and diseased plant leaves, pre-trained algorithms can extract distinct visual cues such as color, texture, and form patterns. Transfer learning is a realistic approach for agricultural applications with limited datasets, as prior research has shown that it enhances classification performance while reducing training complexity [54]. Transfer learning is considered a suitable approach for developing lightweight image classification models for precision agriculture and TinyML-based deployment because of its robust feature extraction capacity and computational efficiency.

B.4.5 MobileNetV1

MobileNetV1, a lightweight convolutional neural network architecture, is used to effectively classify images on mobile and embedded devices. Its architecture comprises two primary operations: pointwise convolution, which uses 1×1 convolutions to integrate the retrieved features, and depth-wise convolution, which filters each input channel independently. To maintain efficient classification performance while reducing computational cost and the number of trainable parameters, the model utilises depth-wise separable convolutions [55].

To detect crop diseases, the MobileNetV1 model performs a sequence of depthwise separable convolutional layers during training. On the other hand, learnable filters identify visual patterns at various levels of abstrac-

tion. While subsequent layers combine these data to identify more complicated disease-specific traits such as lesion patterns, chlorosis, and necrotic regions, the earliest layers identify low-level features such as edges, textures, and color changes [56]. To reduce classification error between the predicted and actual disease labels, the model iteratively changes its filter weights by gradient-based optimisation and backpropagation to learn hierarchical representations. After being trained, MobileNetV1 can reliably identify diseases across various imaging conditions by extracting these learnt feature representations from previously viewed leaf images without the need for manually developed features [57].

MobileNetV1 has been widely used for crop disease detection due to its lightweight architecture and processing performance. Previous research has shown that MobileNetV1 can maintain a reasonably modest model size and little computational cost while accurately classifying healthy and diseased plant leaves. As a result, MobileNetV1 has gained popularity as an architecture for developing intelligent crop disease detection systems that facilitate precision agriculture and prompt disease identification.

B.5 Limitations of Traditional Machine Learning

The constrained processing resources present several problems in deploying edge devices in agricultural areas. Large datasets can be handled by cloud computing, although latency problems are introduced. Even though edge AI tools lessen this load, there are still issues, such as massive data volume, a lack of real-time processing, and storage and communication limitations [19]. Another drawback of conventional machine learning algorithms is the limited capacity to recognise intricate visual patterns present in images of plant diseases. Due to environmental factors, lighting, background clutter, and leaf orientation, crop diseases frequently show modest variations in lesion appearance, color distribution, texture, and severity. These intricate and highly variable qualities may not be effectively captured by hand-crafted features, which could result in worse classification performance and poor generalisation to new data [58].

Even though both conventional and innovative models have demonstrated excellent accuracy in disease detection tasks, present resource limitations limit their applicability for end-user deployment. Implementing these models on devices with limited resources, such as those seen in actual agricultural settings, can be difficult because they usually demand a large amount of processing power [59].

B.6 TinyML for Edge Applications

The combination of edge computing and the Internet of Things (IoT) offers a fresh chance to apply machine learning methods to embedded devices with limited resources at the network's edge. Traditional machine learning needs a tremendous amount of processing power. The goal of embedded machine learning, or Tiny Machine Learning (TinyML), paradigm, is to bring machine learning to low-power, resource-constrained devices like microcontrollers. These devices typically operate with just 1 mW of power, with limited memory and small connectivity. One of the goals of the TinyML paradigm is to go from conventional high-end systems to low-end customers [60].

By processing on the edge, TinyML reduces the need for frequent communication, saving energy, extending battery life, and enabling the use of low-bandwidth networks (e.g., LPWAN). On-device inference also enhances data privacy and security by eliminating the need to transfer sensitive information over the internet [61]. The importance of edge computing in contemporary technological applications is becoming more widely recognised, especially in real-time data processing. For time-sensitive agricultural operations like sorting, classification, and disease detection, edge computing significantly lowers latency and improves responsiveness by enabling computation closer to the data source. TinyML has made it possible for lightweight machine learning models to operate on low-resource edge devices, allowing small and medium-sized farmers to take advantage of sophisticated AI capabilities without the need for expensive infrastructure [62].

Environmental stressors, which can be classified as either biotic or abi-

otic, cause crops to lose output. Biotic stress is caused by pests and pathogens like fungi, bacteria, and nematodes. Abiotic stress, on the other hand, is caused by events like drought, floods, and extremely high or low temperatures [63]. According to [38], diseases cause over 40% of the world's crops to be lost each year. Thus, early and precise detection is crucial for efficient management and increased output. Although leaves are essential for diagnosis because they exhibit the earliest symptoms of illness, physical inspection is time-consuming and prone to mistakes. More accurate and effective methods for detecting and managing plant diseases are provided by emerging technologies such as deep learning, machine learning, and the Internet of Things [38].

TinyML can execute sophisticated machine learning algorithms and enable devices to generate accurate results for specific scenarios. Implementing lightweight deep learning models on resource-constrained devices like smart cameras makes it possible to identify crop diseases efficiently [64]. On the other hand, traditional deep learning and machine learning methods for fruit quality classification rely heavily on cloud computing, which is not ideal for edge deployment due to high power and connectivity requirements. In contrast, edge AI processes data locally, saving bandwidth and energy. Therefore, more researchers are promoting TinyML, enabling efficient, low-power fruit quality analysis independent of cloud infrastructure [22].

Without relying on online infrastructure, TinyML enables sensor devices to perform machine learning tasks locally, including crop health monitoring, disease detection, and crop prediction. It offers several key comparative advantages, including enhanced security, reduced latency, improved privacy, and lower energy consumption. TinyML exhibits great promise in smart farming, especially in areas where embedded AI is underutilised. The use cases, datasets, algorithms, frameworks, and performance metrics—such as power, memory, latency, and accuracy—of some key TinyML applications in agriculture are listed in Table B.4.

Table B.4: Applications of TinyML in smart farming. Annotation: "- -" is used to signify that authors did not provide information.

Application	Dataset	ML Algorithm	Target device	Accuracy (%)	Latency	Framework	Memory Consumption
Plant health monitoring [65]	Collecting Data Using Camera	CNN	Sony Spresense	96.2%	1544 ms	TensorFlow	362.5 KB
Grape leaf Esca disease identification [66]	ESCA dataset	CNN with DP decomposition	OpenMV Cam STM32H7	98%	100 ms	TensorFlow	- -
Identification of coffee leaf diseases [67]	Coffee dataset	CNN quantized with Q-format	STM32 F746G DISCOVERY kit	95%	299.60 ms	TensorFlow and X-cube- AI	32.35 KB
Recognition of fruits and vegetables [68]	Fruit-360 dataset	Mobile NetV1	ESP 32 Cam	17 classes: 77 classes: 98	51 ms	TensorFlow	66.1 KB
Detection of crop diseases [69]	Tomato and potato dataset	CNN	Sony CXD5602	99%	140 ms	- -	370 KB
Recommending optimal crops to cultivate [70]	Crop-recommender dataset	Neural Network	STM32-F401CC	92.2%	1 ms	Edge impulse	1.7 KB
Crop disease detection [23]	Harvard Diverse and Mendeley Data	CNN	Arduino Ble 33 Sense	94.60%	7.60 ms	TensorFlow and Edge impulse	344.7 KB
Identification and quantification of strawberry yield [71]	Collecting data by sensors	FOMO	Arduino Portenta H7	90%	148 ms	Edge impulse	2,439 KB

The table illustrates how TinyML provides effective, real-time intelligence directly on edge devices, thereby bringing about a revolutionary change in smart farming. Time-sensitive agricultural operations require quick solutions, and TinyML models provide latency as low as 1 ms. Additionally, they have a tiny memory footprint (some as small as 1.7 KB), which makes them ideal for low-resource microcontroller deployment. These models maintain high accuracy despite their small size; in tasks such as agricultural disease detection, fruit recognition, and crop recommendation, several of these achieve above 95% accuracy. The flexibility of TinyML in balancing model complexity with the resource limits of various edge devices is further demonstrated by its implementation on a wide range of microcontrollers, with inference times ranging from as low as 1 ms on the STM32-F401CC to 1544 ms on the Sony Spresense. TinyML is a key facilitator of sustainable and scalable smart agriculture because of its combination of speed, efficiency, and dependability, which reduces the need for cloud processing, decreases costs, and permits localised, real-time decision-making.

B.6.1 Model Optimisation for TinyML

Quantization is the process of transforming model parameters and activations from 32-bit floating-point (Float32) representations to 8-bit integer (INT8) representations to reduce memory needs and computational complexity.

Deep learning models are commonly trained using 32-bit floating-point representations, which offer high numerical precision throughout the learning process. However, these models often require a lot of memory and processing power, which makes them unsuitable for use on edge devices with limited resources, such as microcontrollers. Model optimisation strategies that reduce computational complexity while preserving satisfactory predictive performance are required since tinyML applications frequently run on devices with constrained flash memory, RAM, and computing power. One of the most popular model optimisation strategies for TinyML deployment is quantization.

Quantization has been effectively used in several studies to enable the deployment of deep learning models on edge devices. For instance, [72] converted 32-bit floating-point models into lower-precision INT8 representations to study the implementation of deep neural networks on microcontrollers. Their findings showed that memory footprint and computational complexity could be significantly reduced without sacrificing predictive performance. In a similar vein, [73] recognised quantization as one of the primary optimisation strategies in TinyML, emphasising how it reduces model size, lowers energy consumption, and permits real-time inference on embedded hardware.

By transforming trained TensorFlow models into TensorFlow Lite (TFLite) INT8 formats prior to microcontroller deployment, other researchers have utilised post-training quantization for convolutional neural network (CNN) models. Quantization is an efficient method for implementing deep learning models on devices with limited resources, as these experiments showed significant reductions in RAM and flash memory requirements with just slight drops in classification accuracy.

Based on observations from previous studies, quantization was used in this study to enhance the lightweight models that were developed for use with the edge device. To reduce memory and computational requirements while maintaining acceptable classification performance, the trained models were converted into TensorFlow Lite format and quantized to INT8 precision. This supported the overall goal of creating a workable TinyML-based maize leaf disease identification system and allowed for effective real-time inference.

B.6.2 TinyML Development Platform

The effective implementation of deep learning models on embedded devices needs specialised software frameworks and optimisation strategies that reduce computational complexity while preserving fair predictive performance. Three essential components are usually used to deploy models in TinyML applications: TensorFlow Lite (TFLite), INT8 quantisation, and

a TinyML development platform such as Edge Impulse. While INT8 quantisation reduces model size and computational requirements by converting floating-point parameters to low-precision integer representations, TensorFlow Lite provides a lightweight runtime environment that enables trained deep learning models to run efficiently on embedded hardware. To optimize deep learning models for embedded devices, prior TinyML studies frequently use the TensorFlow Lite (TFLite) format with INT8 quantization. In keeping with this strategy, this study also utilizes TFLite and INT8 quantization to ensure effective deployment within the Arduino Portenta H7's memory and computational limits.

On the other hand, Edge Impulse is a complete TinyML development platform that enables the effective deployment of machine learning solutions on resource constrained edge devices by streamlining data preprocessing, model training, data validation, model tuning, and deployment. For agricultural applications where devices need to be both low-power and efficient, Edge Impulse is a good fit because of its integrated optimization capabilities, support for TensorFlow Lite deployment, and capacity to produce lightweight yet accurate models [74]. Moreover, the platform offers a comprehensive workflow for dataset preparation, preprocessing, neural network training, and microcontroller deployment [75]. This is critical for this study, which aims to provide real-time crop health monitoring in the field, where reliable internet connectivity may not always be available.

These technologies work together to enable the realistic deployment of intelligent models on low-power microcontrollers and form the basis of contemporary TinyML systems.

B.7 Chapter Summary and Research Gap

This study examined TinyML's expanding use in precision agriculture, particularly for real-time crop health monitoring and image-based crop disease identification. Previous studies show that while conventional machine learning and deep learning models can achieve excellent classification accuracy, their high memory, computational, and energy requirements restrict

deployment on edge devices. By enabling machine learning inference directly on low-power embedded systems, TinyML overcomes these constraints by lowering latency, communication overhead, and reliance on cloud infrastructure. The growing use of lightweight deep learning architectures for agricultural applications was also mentioned in the review. Developing models that meet the memory of edge devices while also offering excellent classification performance still remains a challenge. Additionally, almost all current research focuses on classification accuracy, paying little attention to deployment viability and practical application on microcontroller-based devices.

This study intends to evaluate several lightweight TinyML models for binary classification of maize leaf images based on these observations and to ensure the models' edge deployment. This study aims to demonstrate how TinyML can overcome the drawbacks of conventional machine learning techniques and offer an approachable, real-time crop disease diagnosis solution for precision agriculture by assessing model performance along with deployment-related factors, such as model size, inference efficiency, and hardware compatibility.

The limited investigation of the trade-off between classification performance and deployment feasibility in TinyML-based agricultural applications is another significant research gap found in the previous studies. Few studies have thoroughly examined how deployment constraints, such as inference latency, flash memory consumption, and RAM utilisation, affect predictive performance when models are implemented on resource-constrained edge devices, even though many prior studies have mainly focused on increasing classification accuracy. As a result, little is known about how various lightweight model architectures balance hardware resource requirements and classification efficiency. To close this gap, the current study compares custom CNN, MobileViT, and MobileNetV1 architectures to determine the trade-offs between deployment feasibility and classification performance. This provides useful information about whether lightweight deep learning models are suitable for real-time TinyML deployment in precision agriculture.

C. Methodology

The main objective of this study is to investigate the use of TinyML in precision agriculture by developing, optimising, and implementing lightweight deep learning models for disease detection in maize leaves. In addition, the research intends to provide real-time crop health monitoring and timely decision-making for sustainable agriculture practices by combining edge computing with image-based analysis. Lightweight deep learning architectures, such as CNN, MobileViT, and MobileNetV1, were chosen for analysis because the study focuses on the deployment in resource-constrained edge devices. Data preprocessing, model development, training, testing, optimization, quantization, and deployment comprised the overall procedure of the study. To assess the effectiveness and capacity for generalisation of the suggested models in various dataset settings, these models were applied separately to two distinct datasets (Dataset1 and Dataset2).

C.1 Datasets

This study uses a dataset of maize leaf images from Harvard Dataverse [76] that is openly accessible. These images, taken in direct sunlight to replicate field conditions, make up the dataset. To ensure consistency for model training, preprocessing was applied to address resolution discrepancies. The researchers of this study [23] also used this dataset to classify maize leaf diseases with a TinyML-based method. The current study focused on binary classification of healthy and diseased maize leaves, while the referenced study used the same dataset for a five-class classification task. Furthermore, this study examined three lightweight deep learning models—CNN, a transformer-based MobileViT, and MobileNetV1—to determine their applicability for TinyML-based maize leaf disease classification, whereas the reference study developed a CNN model. The reference research only served as a methodological baseline due to the disparities in classification objectives. Additionally, this study evaluated the proposed

models on a different resource-constrained edge computing platform using the Arduino Portenta H7, whereas the reference study implemented its model on the Arduino Nano 33 BLE Sense.

The balanced dataset of 3000 images of maize leaves—1500 of which were healthy and 1500 of which were diseased—was split into training and testing subsets for developing the model and assessing its performance using an 80:20 ratio. All images were consistently downsized to 48x48 pixels in RGB format to ensure consistency in input dimensions while preserving important color-based features. A normalisation step that brought the pixel values into a defined range using $\text{rescale} = 1./255$ improved the stability and convergence of the training. Representative samples of images of healthy and diseased maize leaves from Dataset1 are shown in Figure C.1. As the dataset includes substantial variability in leaf appearance, disease severity, lighting conditions, and background features, it is well-suited for assessing the robustness and generalisation capabilities of the proposed TinyML models.



Figure C.1: Samples of Maize Leaf Images from Dataset1 (top row: healthy leaves; bottom row: diseased leaves)

Another dataset of maize leaves was also used to assess the study's robustness and generalisability. This dataset was created using images from

the PlantVillage and PlantDoc datasets and was sourced from a publicly accessible Kaggle repository. The original dataset developers excluded several images during dataset preparation to enhance the quality and significance of the data. The authors of these studies [77] and [78] provided the initial datasets. To ensure consistency, a balanced dataset of 2320 images of maize leaves—1160 healthy and 1160 diseased—was preprocessed in the same way as the prior dataset. Figure C.2 demonstrates the representative images from dataset2.



Figure C.2: Samples of Maize Leaf Images from Dataset2 (top row: healthy leaves; bottom row: diseased leaves)

Since maize (*Zea mays* L.) is one of the most extensively grown cereal crops in the world and is used as a staple meal, animal feed, and industrial raw material in many nations, the maize leaf dataset was utilised for this study. Maize is a crop of great agronomic and commercial significance that greatly enhances rural livelihoods and global food security. However, a number of foliar diseases can significantly reduce crop yield, grain quality, and overall productivity if these are not identified and treated in a timely manner. To promote precision agriculture and sustainable crop management, the development of dependable and automated maize leaf disease detection systems has become crucial.

C.2 Model Development Workflow

As shown in Figure C.3, the workflow for classifying maize leaves into binary categories comprises multiple steps. Data acquisition is the first step of the process, followed by preprocessing. The dataset is first divided into training and testing subsets, and then data normalisation is performed to ensure a uniform range of pixel values. The best-performing model is then obtained via hyperparameter tuning during model training using normalised data.

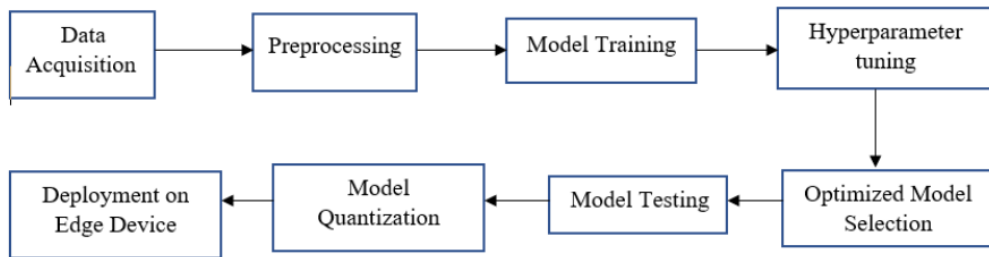


Figure C.3: Workflow of model development

To enable deployment on a resource-constrained edge device, the optimal model is chosen based on the obtained classification accuracy. Then, the model testing phase evaluates the performance of the trained and optimised model using unseen test data to determine its classification accuracy, generalisation capability, and overall reliability. After the conversion of the optimised model to TensorFlow Lite format, 8-bit quantization was used to minimise accuracy loss while reducing the model size. To assess the trained model's real-time inference capabilities, it was finally implemented on a microcontroller.

For all three models, the same development process was used to ensure uniformity and maintain an even comparison. The same data preprocessing, dataset partitioning, training, hyperparameter tuning, validation, testing, quantization, and deployment processes were used for every model. Additionally, the assessment criteria, experimental conditions, and input image resolution were all kept constant during the development process. A solid foundation for evaluating the viability of the three lightweight mod-

els for both precise disease identification and effective edge deployment is provided by the consistent development and evaluation procedure.

C.3 CNN Model

The TensorFlow and Keras frameworks were used to build a convolutional neural network (CNN) because they handle image-based classification tasks well. Data preprocessing, model development, hyperparameter tuning, cross-validation, and performance evaluation are all part of the whole procedure. For implementation in an embedded device, the proposed method aims to maintain computational efficiency while achieving high classification accuracy.

Convolutional Neural Networks (CNNs) were selected because they excel at identifying disease patterns in crop leaves, as they can automatically learn hierarchical spatial information from images. Recent research has shown that CNN-based models have greatly increased the accuracy of plant disease identification [79]. They are also well suited for real-time TinyML-based crop health monitoring due to their processing efficiency and ability to capture local visual features [80].

C.3.1 Optimised CNN model

As shown in Figure C.4, the optimised convolutional neural network (CNN) model consists of four 2D convolutional layers, a fully connected dense layer, and an output layer. A Rectified Linear Unit (ReLU) activation function and a max-pooling operation with a pooling size of 2×2 were applied with 64 filters and a kernel size of 3×3 in the first convolutional layer. 16 filters with a 3×3 kernel, ReLU activation, and 2×2 max-pooling were used in the second convolutional layer.

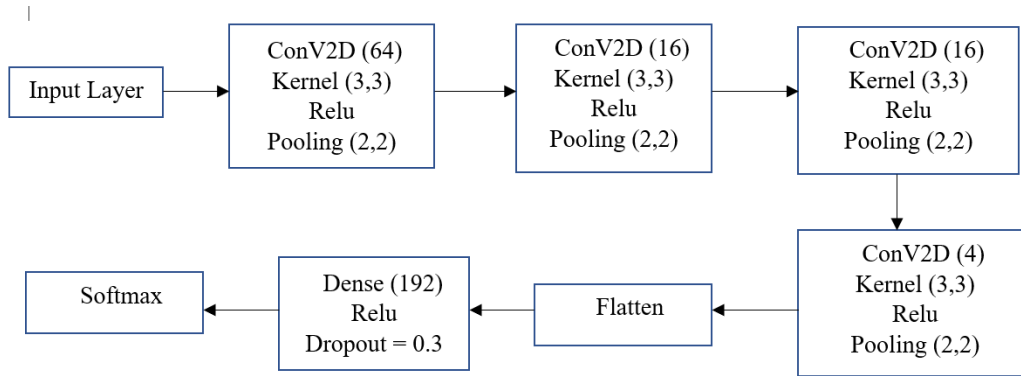


Figure C.4: Block Diagram of the CNN Binary Classification Model

Similarly, the third and fourth convolutional layers—each with a 3×3 kernel, ReLU activation, and 2×2 max-pooling—used 16 and 4 filters, respectively. After being flattened, the feature maps produced by the last convolutional layer were sent to a fully connected dense layer for classification. To facilitate fast gradient-based optimisation and successful learning for the binary classification problem, the network was trained using the Adam optimiser with categorical cross-entropy as the loss function.

To effectively extract local spatial characteristics while reducing the number of parameters and computational complexity, the convolutional layers were built using 3×3 kernels. To gradually reduce spatial dimensions and improve feature resilience, a 2×2 max pooling method was used, which reduced memory and processing requirements. The same architectural arrangements were also used in the [79] study, demonstrating their efficacy and suitability for applications for the classification of plant diseases. Due to its computational effectiveness and ability to address vanishing gradient problems, the ReLU activation function was selected for use in a TinyML framework for microcontrollers with limited resources.

The hyperparameter configurations explored and the corresponding ideal values chosen for the proposed convolutional neural network model are presented in Table C.1. Based on the suggestions in the systematic review by [81], which identified the number of convolutional layers, convolutional filters, and dense units as crucial architectural hyperparameters that affect CNN performance, the hyperparameter search space was established for the

CNN model. Several filter ranges were examined across the convolutional layers of the proposed CNN model to optimise hyperparameters. While significantly smaller filter ranges were considered in the deeper convolutional layers to reduce computational complexity and improve model efficiency, larger filter ranges were investigated in the first convolutional layer to enable broad feature extraction from maize leaf images. As a result, for the first convolutional layer, filter values between 8 and 128 were evaluated, whereas lower ranges were looked into for the subsequent convolutional layers. The number of convolutional blocks ranged from 1 to 5 to balance feature extraction and model complexity. Dropout values between 0.2 and 0.5 were assessed to efficiently manage overfitting, while learning rates between 0.01 and 0.0001 were examined to guarantee steady convergence during training.

Table C.1: Hyperparameter tuning range and selected optimal values

Layer/Parameter	Tuning Range	Selected Value
Conv Block	1-5	4
Conv Layer 1- Filters	8,16,32,64,128	64
Conv Layer 2- Filters	4,8,16,32,64	16
Conv Layer 3- Filters	4,8,16,32,64	16
Conv Layer 4- Filters	2,4,8,16,32,64	4
Learning Rate	0.01, 0.001, 0.0001	0.001
Dropout	0.2-0.5, step=0.1	0.3
Dense Units	32-256, step=32	192

Dense layer units from 32 to 256 were considered to offer sufficient representation of strength without sacrificing computational performance. Four convolutional blocks with filter sizes of 64, 16, 16, and 4, a learning rate of 0.001, a dropout rate of 0.3, and 192 dense units made up the ideal configuration, according to performance evaluation.

A batch size of 32 was chosen to strike a balance between computational efficiency and stable gradient updates during model training. For hyperparameter tuning, the number of epochs was set to 10 to reduce computational cost and accelerate the search process. As multiple model configurations are evaluated during tuning, a moderate epoch setting allows sufficient learning for comparison without excessive training time.

C.4 Transformer Based Architecture

C.4.1 MobileViT

This study investigated a custom transformer-based MobileViT model for classifying maize leaf images. This model was chosen because it allows efficient feature representation by combining the global contextual learning capabilities of transformer blocks with the local feature extraction capability of convolutional layers. In addition, its lightweight hybrid architecture makes it a good fit for TinyML applications, where computational efficiency and classification performance are crucial factors.

C.4.2 Optimised MobileViT Model

This model was developed using the workflow presented in Figure C.3. The range of pixel values of 0 to 1 was used to normalise the input images after resizing to 48×48 pixels. The categorical cross-entropy loss function and the Adam optimiser were used to train the MobileViT model. By returning the top-performing model weights, an early stopping technique was used during training to maintain validation accuracy and avoid overfitting. After training, the best model configuration for classifying maize leaves was found by optimising hyperparameters.

The Random Search technique, developed by Keras Tuner, was used to optimise hyperparameters to find the best model configuration. Several architectural and training parameters were investigated during the optimisation process. To find the configuration that offered the best generalisation performance, validation accuracy was chosen as optimisation goal. The final MobileViT model was chosen for further evaluation and deployment based on the hyperparameter combination that produced the best validation accuracy.

The examined hyperparameters and the best values found for the MobileViT model using Random Search are shown in Table C.2. To find a configuration that improved validation accuracy while preserving compu-

tational efficiency, the tuning procedure investigated various projection dimensions, numbers of transformer blocks, dense layer sizes, learning rates, and dropout rates. The chosen model included two transformer blocks, 64 dense units, a projection size of 64, a learning rate of 0.0005, and a dropout rate of 0.5. In addition, a batch size of 32 was chosen as it offers a good balance between training stability, memory usage, and computational performance.

Table C.2: Hyperparameter tuning range and selected optimal values

Hyperparameter	Tuning Range	Selected Value
Projection Dimension	32,64,96,128	64
Number of Transformer Blocks	2,3,4	2
Dense Units	32,64,128,192,256	64
Learning Rate	0.0001, 0.0005, 0.001	0.0005
Dropout	0.3-0.5, step=0.1	0.5

Based on suggestions from earlier research on lightweight vision transformer designs and hyperparameter optimisation, the hyperparameter search ranges were chosen. In accordance with the MobileViT design guidelines recommended by Mehta and Rastegari [50], the projection size and transformer block structure were selected to balance between feature representation capabilities and computational efficiency. The number of transformer blocks governs the model’s capacity to capture global feature associations, while the projection dimension controls the size of the feature embeddings processed by the transformer. According to widely used optimisation and regularization techniques documented in hyperparameter tuning literature [81], the learning rate and dropout ranges were chosen. To ensure compatibility for TinyML deployment, the search space was limited to minimal setups.

C.5 Transfer Learning

C.5.1 MobileNetV1

MobileNetV1 was selected for this study as the lightweight architecture after the development of the custom CNN and transformer-based MobileViT

models, as it is one of the most effective convolutional neural network architectures for resource-constrained applications. To retain competitive classification performance while drastically reducing computational complexity, model size, and memory constraints, MobileNetV1 uses depthwise separable convolutions. As MobileNetV1 reduces the number of trainable parameters, it is the ideal choice for deployment on an edge device. Additionally, it is a suitable transfer learning baseline for evaluating the performance of the proposed custom lightweight models in terms of both classification accuracy and deployment feasibility because of its widespread use in TinyML research and agricultural image classification.

C.5.2 Optimized MobileNetV1 model

The process shown in Figure C.3 was used to build the model. The same normalisation and scaling techniques were used to preprocess the dataset images prior to training. The core feature extractor of the model was the pretrained MobileNetV1 model with ImageNet weights. The input images were original 48×48 -pixel images of maize leaves. Before the images were fed into the MobileNetV1 architecture, zero-padding was applied to enhance feature representation while preserving spatial information, resulting in an effective input size of 64×64 pixels. A custom classification comprising a Global Average Pooling layer, a fully connected dense layer, batch normalisation, dropout regularisation, and a final softmax output layer for binary classification was used in place of the top classification layers of the pre-trained model.

To optimise the MobileNetV1 model, a variety of layers were unfrozen during transfer learning. While unfreezing more than 30 layers increased memory and processing needs, configurations with 10 or 20 unfrozen layers produced lower accuracy. For TinyML deployment, 30 unfrozen layers were chosen, providing a balance between resource efficiency and classification performance.

Table C.3 shows the hyperparameters that were examined during the MobileNetV1 model optimisation procedure. To enhance classification per-

formance, the tuning process concentrated on determining the ideal values for the alpha parameter, dense layer units, dropout rate, and learning rate. The optimal configuration for the suggested model was found to have an alpha value of 0.5, 64 dense units, a dropout rate of 0.3, and a learning rate of 0.001 based on the tuning results.

Table C.3: Hyperparameter tuning range and selected optimal values

HyperParameter	Tuning Range	Optimized Value
Alpha	0.25, 0.5, 0.75	0.5
Dense Units	32-256, step=32	64
Dropout Rate	0.2, 0.3, 0.4	0.3
Learning Rate	0.0001, 0.0005, 0.001	0.001

Standard MobileNet settings, prior deep learning research, and TinyML deployment constraints were taken into account while choosing the hyperparameter tuning ranges. To assess the trade-off between computational efficiency and classification performance, this work employed alpha values (0.25–0.75) from the original MobileNetV1 design [53]. While the learning rate range was determined based on commonly used fine-tuning parameters for transfer learning-based image classification models, the dense unit and dropout ranges were chosen to balance model complexity, generalisation capability, and memory efficiency.

For effective learning and memory management, a batch size of 32 was employed throughout the training procedure. Due to its adaptive learning capabilities and efficacy in deep learning applications, the Adam optimiser was utilised. Early stopping was used to minimise overfitting and restore the top-performing weights based on validation accuracy during the model’s 25 epochs of training.

C.6 Quantization

The CNN, MobileViT, and MobileNetV1 models were post-trained and quantized to enable deployment on resource-constrained edge devices. Each model was trained, then converted to the TensorFlow Lite (TFLite) format

and quantized to INT8. To accurately convert floating-point values to integers, the dynamic range of activations was calibrated using a sample dataset. After that, the quantized models were assessed to determine the effects of quantization on resource usage and classification performance.

To facilitate a fair comparison of their deployment characteristics, the quantization process was applied uniformly to the CNN, MobileViT, and MobileNetV1 models. Metrics such as accuracy, latency, flash memory utilization, and RAM consumption were examined after the final INT8 models were deployed and benchmarked on the edge device.

The feasibility of lightweight deep learning models for edge devices with limited resources can be determined by the chosen deployment metrics. The responsiveness of a real-time disease-detecting process is directly influenced by latency, which is the duration of time required for a model to analyse an input and produce a prediction. A crucial factor for microcontrollers with limited non-volatile memory is flash memory consumption, which shows the amount of storage space needed to support the deployed model and related program code. In a similar vein, RAM consumption indicates how much working memory is required for model inference and establishes whether a model can run within the target device's hardware limitations. In general, these metrics provide a useful indication of whether a model can be effectively implemented for real-time crop disease monitoring in precision agricultural settings with limited computational resources. So, reducing these metrics demands makes it more likely that the quantised model will allow real-time inference and be performed effectively on embedded microcontrollers.

C.7 Deployment of TinyML Models

Machine learning models can be run on resource-constrained embedded devices because of TinyML [82]. The deployment method used in this study to bring the trained deep learning models on a microcontroller platform with limited resources is shown in Figure C.5. To ensure a fair comparison of the CNN, MobileViT, and MobileNetV1 models' deployment feasibility and

inference performance, the identical deployment process was used for each model.

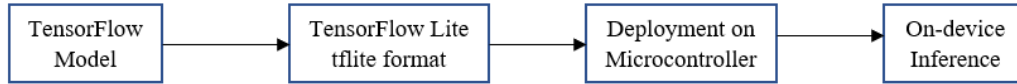


Figure C.5: Framework of the TinyML Deployment Process

After the training and evaluation stages, the trained TensorFlow model was obtained. For additional improvement and deployment, the model architecture and learnt parameters were recorded in TensorFlow format. The trained model was converted into TensorFlow Lite (TFLite) format in the second step. The lightweight model representation designed especially for edge and embedded devices is offered by TensorFlow Lite. This modification reduces computational overhead and gets the model ready to run in settings with limited memory.

The TensorFlow Lite (TFLite) model was uploaded to the Edge Impulse platform for deployment after conversion. The improved model and associated libraries were included in the deployment package that Edge Impulse created and exported as a ZIP file. After that, the deployment package was integrated into the firmware for on-device execution and imported into the Arduino IDE. To provide effective model inference on the intended embedded platform, sufficient memory allocation and library configurations were set.

Then, the deployed model was fed raw image feature values via the Arduino IDE to perform on-device inference. The embedded system successfully executed the model on the target edge platform by processing the input data locally and generating classification predictions.

C.7.1 Selection of an Edge Device

Several microcontroller platforms were considered as possible edge devices for implementing the developed TinyML models. The selection was largely based on CPU capability, available Flash memory, and RAM, as these re-

sources determine the ability of a microcontroller to store the model and conduct the accompanying inference processes. To evaluate hardware demands, Table C.4 compares the Arduino Portenta H7 and several additional microcontroller platforms.

Table C.4: Candidate Microcontroller Platforms and Available Memory for TinyML Deployment

Microcontroller Platform	Processor	Flash Memory	RAM Memory
Arduino Portenta H7	Arm Cortex M7-M4 Dual Core 480 MHZ	2 MB	1 MB
Arduino Nano 33 BLE Sense	nRF52840 64MHZ	1 MB	256 KB
STM32F411RE	Arm Cortex-M4, up to 100 MHz	512 KB	128 KB
ESP32-based MCU	RISC-V single-core, up to 160 MHz	4 MB	400 KB

As shown in Table C.4, the Arduino Portenta H7 offers greater processing capability and RAM capacity than the other platforms considered, although the ESP32-C3 configuration has greater Flash capacity. These resources are important for TinyML deployment, as memory is required not only to store the quantised model but also to accommodate the firmware, inference runtime, input data, and intermediate tensors. The Portenta H7 was therefore selected as the experimental platform because its dual-core processor and comparatively large memory capacity provide sufficient resources to deploy and evaluate the developed models without imposing overly restrictive hardware constraints.

Previous studies, such as [74], which examined TinyML-based fault detection using the Edge Impulse platform and microcontroller deployment, also support using the Arduino Portenta H7 microcontroller as a TinyML deployment platform. Additionally, the future objective of this study is to expand binary classification to multiclass maize disease identification, which will likely demand higher computational power and model capacity. As a result, the Portenta H7 was selected to offer adequate hardware capability for both the present binary classification task and the anticipated

future development of a multiclass TinyML system.

C.7.2 Experimental Deployment Setup

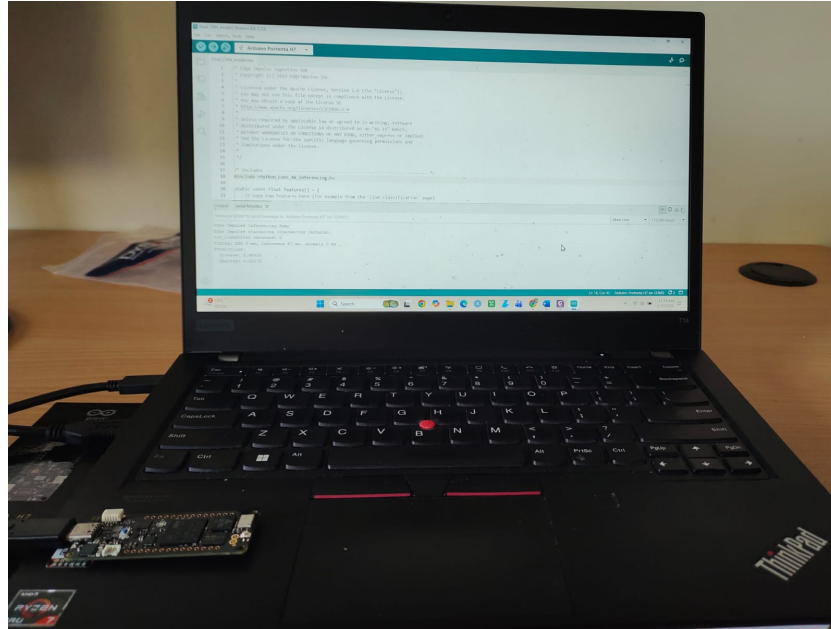


Figure C.6: Experimental deployment setup

The experimental deployment configuration used to assess the suggested TinyML models is shown in Figure C.6. The Arduino Portenta H7 microcontroller was used to implement the optimised and quantized TensorFlow Lite models. To upload models, communicate serially, and track inference performance, the device was connected to a host computer via USB.

The deployment architecture was developed to simulate a practical edge-computing environment for maize leaf disease detection. Performance measures, including RAM usage, ROM, inference time, and classification accuracy, were monitored while the deployed model processed the test images. This setup enabled it to assess the models in real-world deployment settings and demonstrated how well-suited they are for resource-constrained agricultural applications.

C.8 Development of the TinyML Model Using Edge Impulse

To confirm model performance and dataset consistency on a specialised TinyML platform, the same dataset (dataset1) were optimised in Edge Impulse after the models were developed in Python. Since the main goal was to test the TinyML deployment method rather than conduct a thorough comparative evaluation across many datasets, this model was developed using only Dataset1. This stage enabled the assessment of deployment-related elements such as hardware compatibility, memory requirements, and model efficiency. Additionally, lightweight designs appropriate for edge deployment were found using the EON Tuner. Confidence in the robustness, repeatability, and usefulness of the proposed crop disease detection framework was strengthened by cross-platform validation in both Python and Edge Impulse.

C.8.1 Edge Impulse-Based Model Development Workflow

The workflow of the Edge Impulse platform's binary classification method for maize leaves is shown in Figure C.7. To build a balanced dataset, the method starts with collecting and labeling images of both healthy and diseased maize leaves. Following preprocessing steps like resizing and normalisation, the collected images are split into training and validation sets. After data preparation and feature extraction, model training is performed. The best model configuration that can reliably classify maize leaves while preserving computational performance is then determined by hyperparameter tuning. The classification performance and reliability of the trained model are then evaluated by model testing. To be deployed on edge devices with limited resources, the optimised model is finally quantized and converted into TensorFlow Lite format.

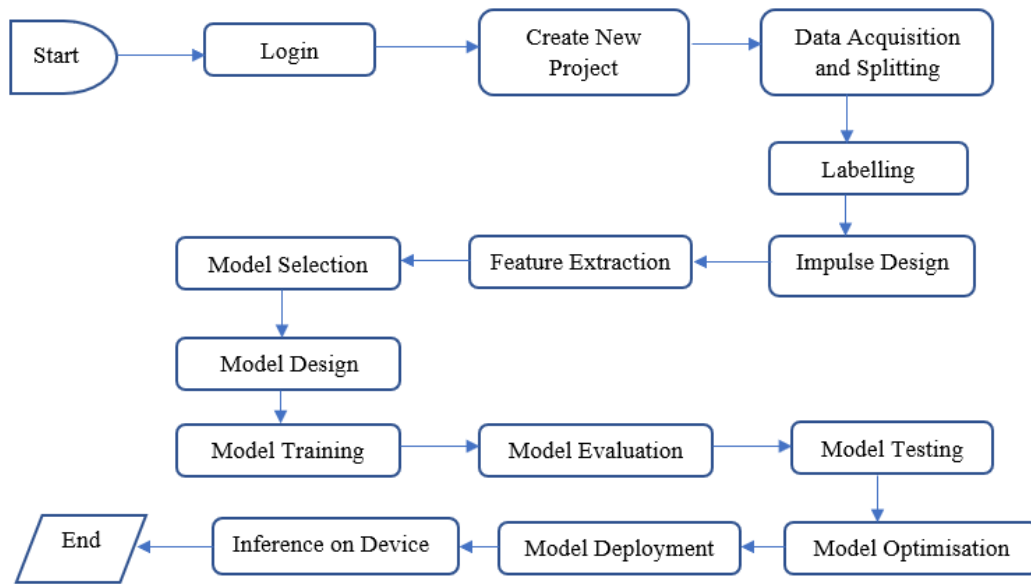


Figure C.7: Workflow of Edge Impulse Platform for Maize Leaf Disease Detection

C.8.2 Model Selection and Optimisation Using EON Tuner

Edge Impulse EON Tuner was used to determine the best model for TinyML-based maize leaf classification. The EON Tuner effectively finds the optimal trade-off between accuracy, latency, and model size—a crucial factor for deployment on edge devices with limited resources—in contrast to manual tuning. This tool was used to test different models on the dataset and compare their performance systematically.

The Python-based TinyML models were developed and evaluated using the same Datasets, which were also used in the Edge Impulse environment. The same data partitioning approach for the training-validation-test and the 48×48 pixel image resolution were maintained to ensure a fair comparison and preserve experimental consistency. The fit-short resize mode was used to standardise the input block layout, whereas RGB and grayscale image processing were the two Digital Signal Processing (DSP) blocks that were evaluated.

The Convolutional Neural Network (CNN) was chosen as the optimal model because it achieved the highest classification accuracy among the neural network architectures studied. Three convolutional layers, each fol-

lowed by a max-pooling operation, a flatten layer, a fully connected dense layer, and a dropout layer with a rate of 0.25 to prevent overfitting, make up the model.

The optimised CNN model design is shown in Figure C.8. The network uses convolutional layers with filter sizes of 64, 32, and 8, respectively, to process $48 \times 48 \times 3$ input images. Throughout the network, ReLU activation functions are used to improve feature learning and add non-linearity. Before the final classification step, the flattened feature maps are passed through a dense layer with 96 neurons.

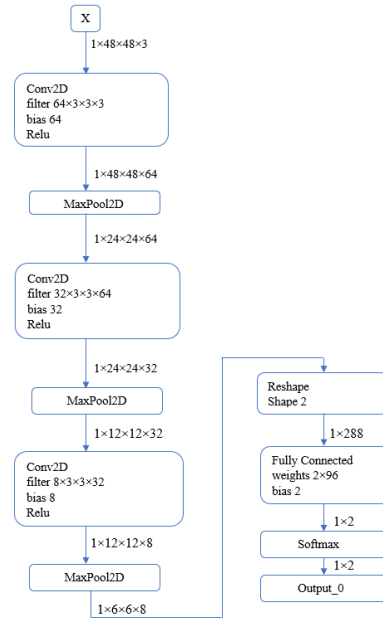


Figure C.8: CNN Model Architecture

The range of hyperparameters investigated during the model building and optimisation process is shown in Table C.5. The effects of various pre-processing methods (grayscale and RGB channels with fit-short scaling) on performance were assessed. To balance model complexity and overfitting, the number of convolutional layers, number of filters, dropout rate, learning rate and dense unit were tested for the unique CNN architecture.

To maintain stable learning, the training parameters- epoch and batch size were kept fixed. Different dense layer sizes, dropout rates, and training cycles were also investigated for transfer learning models, such as Mo-

Table C.5: Hyperparameter tuning range

Component	Parameter	Values Explored
Input Block	Resize Method	Fit-short
Pre-processing	Image Channels	Grayscale, RGB
CNN Architecture	Number of Convolutional Layers	2-5
	Number of Filters	4, 8, 16, 32, 64, 128
	Dropout Rate	0.25, 0.5
	Learning Rate	0.001, 0.0005, 0.0001
	Dense Units	16, 64, 96, 128
Training Parameters	Training Cycles (Epochs)	20
	Batch Size	32
Transfer Learning Models	Model Type	MobileNetV1, MobileNetV2
	Dense Units	16, 64, 96, 128
	Dropout Rate	0.25, 0.5
	Training Cycles	0.001, 0.0005, 0.0001

MobileNetV1 and MobileNetV2. To enhance generalisation, data augmentation techniques—were also taken into consideration. The model was able to investigate various architectural configurations and find the best solution while lowering the risk of overfitting and enhancing the robustness of the suggested framework due to the predefined hyperparameter search ranges, which offered a methodical and optimisation framework. All things considered, this methodical investigation helped determine the task’s most effective and precise model arrangement.

C.8.3 Model Deployment

To further validate the proposed TinyML framework, this edge-Impulse-based TinyML was also deployed on the Arduino Portenta H7, as the Python-developed models. To reduce computational complexity and memory requirements for optimal execution on the Arduino Portenta H7, the Edge Impulse model was optimised and converted to TensorFlow Lite (TFLite) INT8 format using post-training quantization. The classification performance, inference time, RAM utilization, and flash memory consumption of the deployed model were then assessed. The robustness and usefulness of

the established maize leaf disease detection process were strengthened by this deployment, which enabled an independent validation of the proposed method by a separate TinyML development environment.

D. Results and Discussion

This chapter presents and discusses the experimental results obtained from the evaluation of the proposed lightweight deep learning models for TinyML-based maize leaf disease detection. The results are organised according to the two maize leaf image datasets and include both classification performance and deployment performance on the Arduino Portenta H7 edge device. Comparative analyses of the proposed custom CNN and MobileViT models against the transfer learning-based MobileNetV1 model are also presented.

D.1 Cross-Validation Results

This section presents the classification performance of the three lightweight deep learning models using five-fold cross-validation on the two publicly available maize leaf image datasets.

D.1.1 Results Using Dataset1

This subsection presents the classification results obtained using Dataset1.

D.1.1.1 CNN Model

The classification performance of the proposed custom CNN model on Dataset1 is summarised in Table D.1. The results are presented for each fold together with the mean and standard deviation to assess the model's performance and consistency across the five cross-validation folds.

Table D.1: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	91.67	91.85	91.67	91.66
Fold 2	89.67	90.46	89.67	89.62
Fold 3	91.00	91.47	91.00	90.97
Fold 4	94.33	94.49	94.33	94.33
Fold 5	92.83	92.89	92.83	92.83
Mean $\pm$ SD	91.90 $\pm$ 1.59	92.23 $\pm$ 1.37	91.90 $\pm$ 1.59	91.88 $\pm$ 1.61

The 5-fold cross-validation results for the proposed CNN model are summarised in Table D.1. The model achieved classification accuracies ranging from 89.67% to 94.33%, with a mean accuracy of 91.90% and a standard deviation of $\pm 1.59\%$, indicating consistent performance across the five validation folds. The mean precision, recall, and F1-score were 92.23%, 91.90%, and 91.88%, respectively, demonstrating balanced classification performance for both healthy and diseased maize leaf images. The relatively low standard deviations across all performance metrics suggest that the proposed CNN generalizes well to different data partitions and learns robust feature representations.

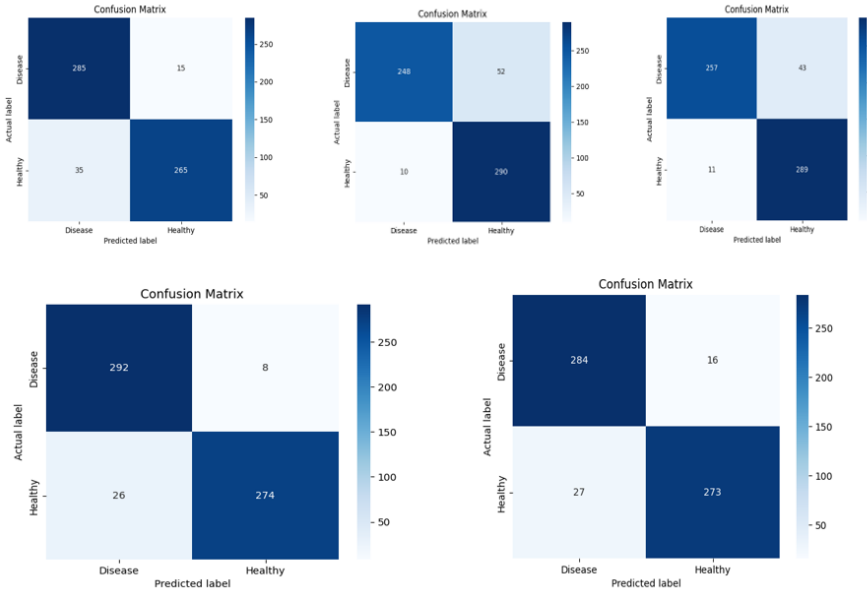


Figure D.1: Fold-wise Confusion Matrices of the CNN Model for Dataset1 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5

The confusion matrices obtained from the 5-fold cross-validation of the proposed CNN model are shown in Figure D.1. Across all validation folds, the model correctly classified the majority of healthy and diseased maize leaf images, with relatively few misclassifications. The consistent dominance of the diagonal elements across the five confusion matrices indicates stable classification performance and good generalisation to different data partitions. These results further demonstrate the CNN model's ability to learn discriminative features for binary maize leaf disease detection.

D.1.1.2 MobileViT Model

The 5-fold cross-validation results for the proposed MobileViT model on Dataset1 are summarised in Table D.2. The model achieved a mean classification accuracy of 96.07% with a standard deviation of $\pm 1.07\%$, outperforming the custom CNN while maintaining consistent performance across all validation folds. The mean precision, recall, and F1-score all exceeded 96%, demonstrating balanced and reliable classification of both healthy and diseased maize leaf images. The improved classification performance suggests that the transformer-based MobileViT architecture may capture both local and global image features more effectively, resulting in more discriminative feature representations.

Table D.2: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	96.83	96.84	96.83	96.83
Fold 2	95.33	95.41	95.33	95.33
Fold 3	95.83	95.95	95.83	95.83
Fold 4	97.67	97.74	97.67	97.67
Fold 5	94.67	94.68	94.67	94.67
Mean $\pm$ SD	96.07 $\pm$ 1.07	96.12 $\pm$ 1.07	96.07 $\pm$ 1.07	96.07 $\pm$ 1.07

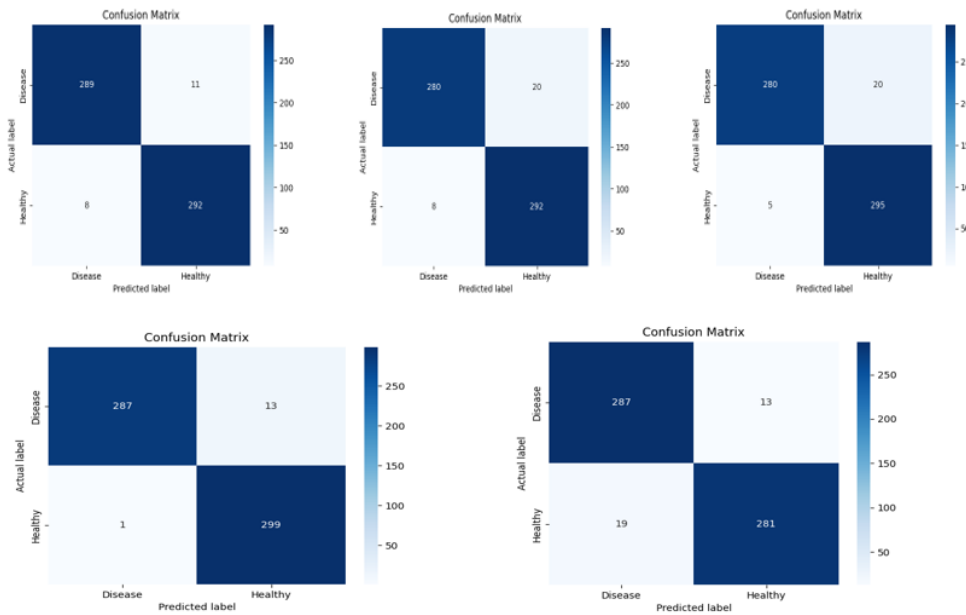


Figure D.2: Fold-wise Confusion Matrices of the MobileViT Model for Dataset1: (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.

The confusion matrices in Figure D.2 further confirm the strong classification performance of the MobileViT model, with very few misclassifications observed across all five validation folds. Compared with the custom CNN, the MobileViT model exhibited fewer off-diagonal elements, indicating improved discrimination between healthy and diseased maize leaf images. These findings suggest that the transformer-based feature extraction enhances disease detection performance. However, the increased architectural complexity is expected to incur higher computational and memory requirements, and this trade-off is examined in the deployment analysis presented later in this chapter.

D.1.1.3 MobileNetV1 Model

The 5-fold cross-validation results for the transfer learning-based MobileNetV1 model on Dataset1 are summarised in Table D.3. The model achieved a mean classification accuracy of 90.37% with a standard deviation of $\pm 1.03\%$, demonstrating stable performance across the five validation folds. The mean precision, recall, and F1-score of 90.83%, 90.37%, and 90.33%, respectively, indicate balanced classification performance for both healthy and diseased maize leaf images. Although the classification performance was slightly lower than that of the proposed custom CNN and MobileViT models, the results demonstrate that the transfer learning-based MobileNetV1 model provides a reliable baseline for maize leaf disease detection.

Table D.3: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	91.17	91.22	91.17	91.16
Fold 2	88.33	90.04	88.33	88.21
Fold 3	91.00	91.18	91.00	90.99
Fold 4	90.67	90.93	90.67	90.65
Fold 5	90.67	90.76	90.67	90.66
Mean $\pm$ SD	90.37 $\pm$ 1.03	90.83 $\pm$ 0.43	90.37 $\pm$ 1.03	90.33 $\pm$ 1.08

The confusion matrices in Figure D.3 show that the MobileNetV1 model correctly classified the majority of healthy and diseased maize leaf images across all validation folds. However, compared with the proposed custom

CNN and MobileViT models, the MobileNetV1 model exhibited a slightly higher number of misclassifications, particularly for diseased maize leaves. This may indicate that the ImageNet pre-trained features used by MobileNetV1 are not fully optimised for the subtle visual characteristics of maize leaf disease. Nevertheless, as demonstrated in the deployment analysis later in this chapter, MobileNetV1 offers favorable deployment characteristics, providing an effective balance between classification performance and computational efficiency for resource-constrained TinyML applications.

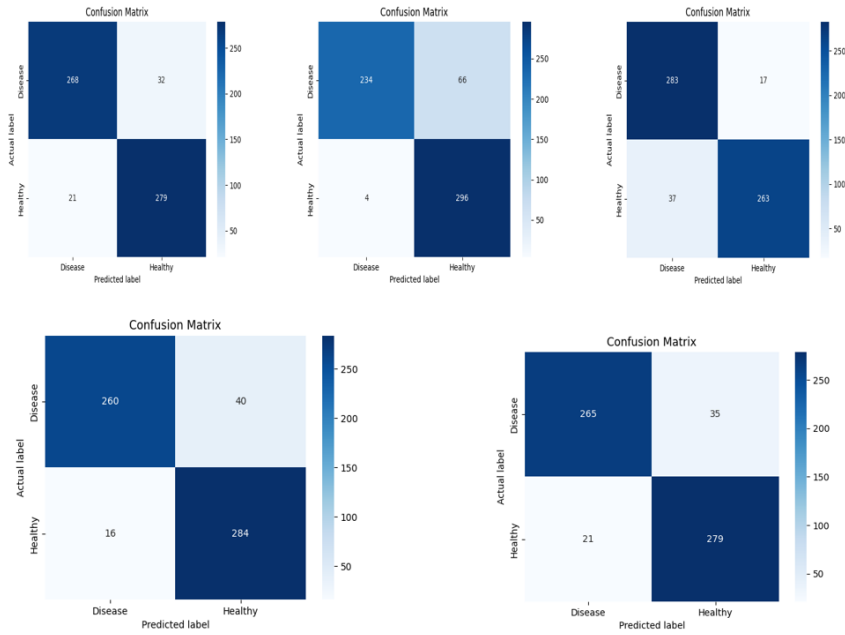


Figure D.3: Fold-wise Confusion Matrices of the MobilenetV1 Model for Dataset1 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5

D.1.2 Results Using Dataset2

This subsection presents the cross-validation results of the three lightweight deep learning models using Dataset2. The objective is to evaluate whether the models maintain their classification performance and generalisation capability on a second publicly available maize leaf image dataset with different image characteristics. The identical architectural configurations and optimised hyperparameter values as Dataset1 were used for Dataset2 to assess the resilience and generalisability of the proposed models.

D.1.2.1 CNN Model

The 5-fold cross-validation results for the proposed CNN model on Dataset2 are summarised in Table D.4. The model achieved a mean classification accuracy of 98.58% with a standard deviation of $\pm 0.52\%$, indicating highly consistent performance across the five validation folds. The mean precision, recall, and F1-score were 98.62%, 98.58%, and 98.58%, respectively, demonstrating balanced classification performance for both healthy and diseased maize leaf images.

Table D.4: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	98.71	98.72	98.71	98.71
Fold 2	98.49	98.54	98.49	98.49
Fold 3	98.92	98.95	98.92	98.92
Fold 4	99.14	99.14	99.14	99.14
Fold 5	97.63	97.74	97.63	97.63
Mean $\pm$ SD	98.58 $\pm$ 0.52	98.62 $\pm$ 0.48	98.58 $\pm$ 0.52	98.58 $\pm$ 0.52

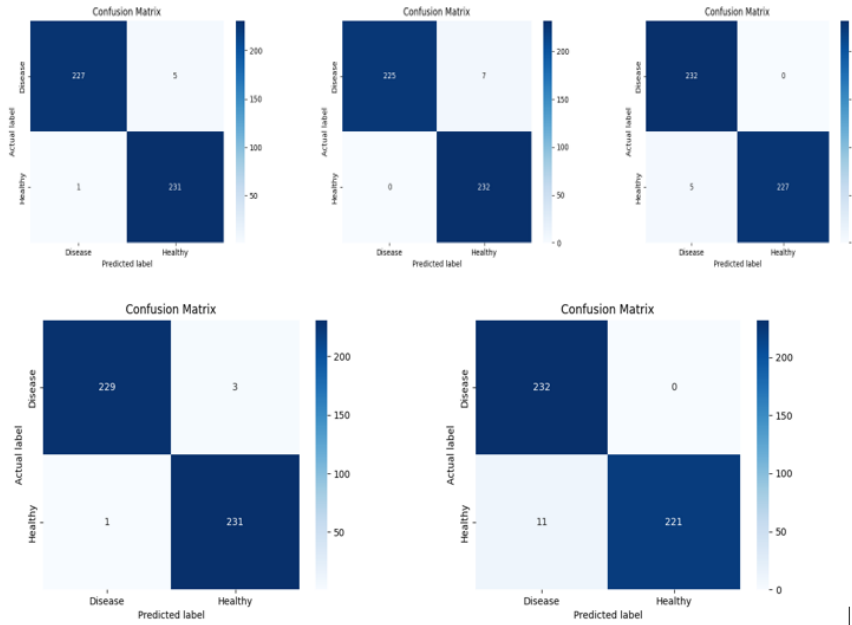


Figure D.4: Fold-wise Confusion Matrices of the CNN Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5

The confusion matrices shown in Figure D.4 further confirm the strong classification performance of the proposed CNN model. Across all five val-

idation folds, the majority of healthy and diseased maize leaf images were correctly classified, with only a small number of misclassifications. Compared with the results obtained on Dataset1, the higher classification accuracy suggests that Dataset2 provides more discriminative image features for disease detection while further demonstrating the robustness of the proposed CNN model.

D.1.2.2 MobileViT Model

The 5-fold cross-validation results for the proposed MobileViT model on Dataset2 are summarised in Table D.5. The model achieved a mean classification accuracy of 98.28% with a standard deviation of $\pm 0.81\%$, indicating consistent performance across the five validation folds. The mean precision, recall, and F1-score were 98.33%, 98.28%, and 98.28%, respectively, demonstrating balanced and reliable classification of both healthy and diseased maize leaf images.

Table D.5: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	99.14	99.14	99.14	99.14
Fold 2	96.77	96.91	96.77	96.76
Fold 3	98.28	98.33	98.28	98.28
Fold 4	98.71	98.71	98.71	98.71
Fold 5	98.49	98.54	98.49	98.49
Mean $\pm$ SD	98.28 $\pm$ 0.81	98.33 $\pm$ 0.75	98.28 $\pm$ 0.81	98.28 $\pm$ 0.81

The confusion matrices in Figure D.5 further confirm the strong classification performance of the MobileViT model. Across all five validation folds, the majority of healthy and diseased maize leaf images were correctly classified, with only a small number of misclassifications. The consistently strong performance across Dataset2 further confirms the strong generalisation capability of the MobileViT architecture.

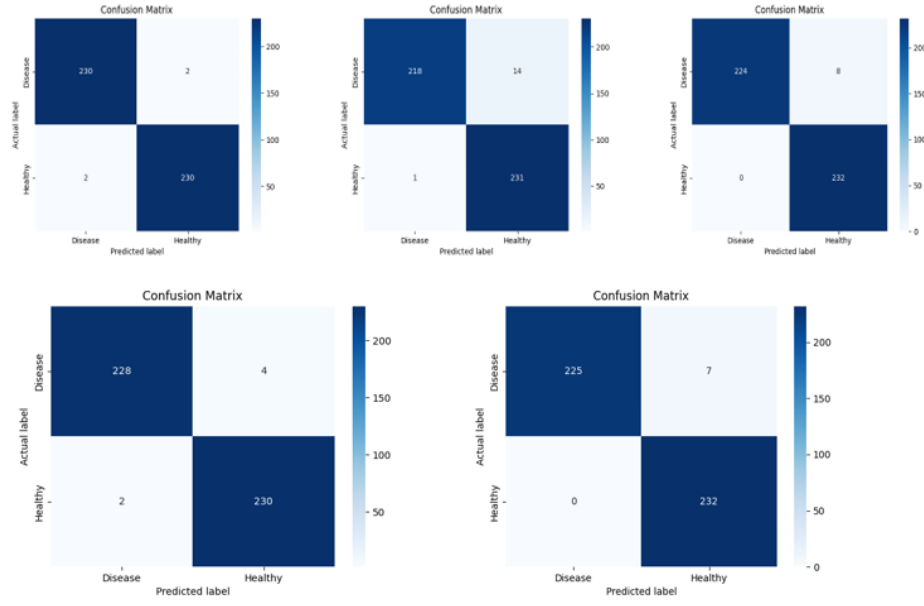


Figure D.5: Fold-wise Confusion Matrices of the MobileViT Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.

D.1.2.3 MobileNetV1 Model

The 5-fold cross-validation results for the MobileNetV1 model on Dataset2 are summarised in Table D.6. The model achieved a mean classification accuracy of 98.97% with a standard deviation of $\pm 0.25\%$, indicating highly consistent performance across the five validation folds. The mean precision, recall, and F1-score were all 98.97%, demonstrating balanced and reliable classification of both healthy and diseased maize leaf images.

Table D.6: Fold-wise Performance Results

Fold	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)
Fold 1	98.71	98.72	98.71	98.71
Fold 2	99.14	99.14	99.14	99.14
Fold 3	98.92	98.92	98.92	98.92
Fold 4	98.71	98.72	98.71	98.71
Fold 5	99.35	99.36	99.35	99.35
Mean $\pm$ SD	98.97 $\pm$ 0.25	98.97 $\pm$ 0.25	98.97 $\pm$ 0.25	98.97 $\pm$ 0.25

The confusion matrices in Figure D.6 further confirm the strong classification performance of the MobileNetV1 model. Across all five validation folds, the majority of healthy and diseased maize leaf images were correctly

classified, with only a small number of misclassifications. Compared with the results obtained on Dataset1, the improved classification performance suggests that the transfer learning-based MobileNetV1 model generalises well when evaluated on Dataset2.

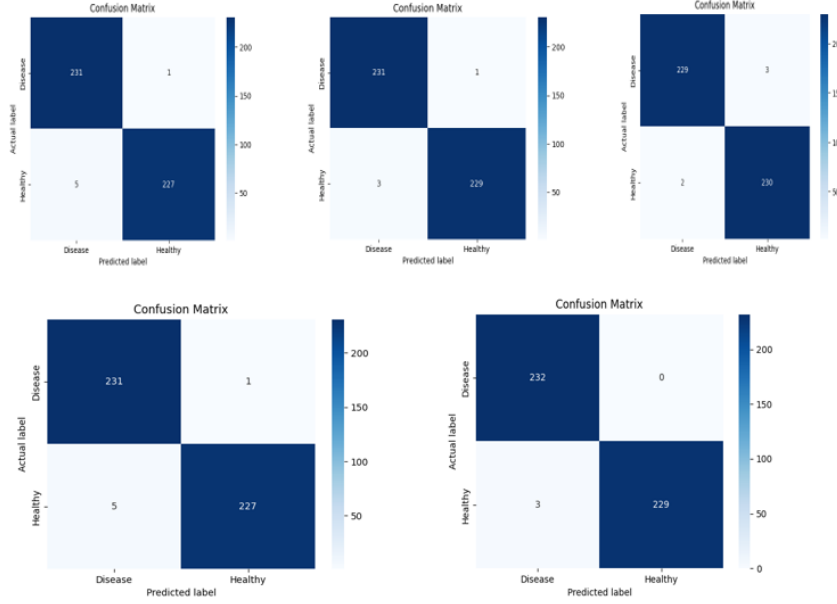


Figure D.6: Fold-wise Confusion Matrices of the MobileNetV1 Model for Dataset2 (a) Fold 1, (b) Fold 2, (c) Fold 3, (d) Fold 4, and (e) Fold 5.

D.2 Deployment Results

D.2.1 Deployment Performance on Dataset1

This subsection evaluates the deployment performance of the three lightweight models on the Arduino Portenta H7 using Dataset1. To evaluate the models' efficacy in a real-world TinyML environment, the evaluation considered both deployment-related features and classification performance. Table D.7 summarises the deployment performance of the three models in terms of inference latency, RAM usage, and Flash memory usage. These metrics are important for TinyML deployment. Inference latency reflects the time required to process a single image, RAM usage indicates the runtime memory required during inference, and Flash memory usage provides a practical indication of the storage required to deploy each model on the target mi-

crocontroller. Deploying TinyML models on microcontrollers requires minimising latency, RAM usage and flash storage where processing power and memory resources are constrained.

Table D.7: On-device Performance of Deployed Models on dataset1

Model Name	Latency (ms)	RAM (KB)	Flash (KB)
CNN	67	222.8	72.4
MobileViT	121	300.3	542
MobileNetV1	71	125	316.1

The deployment results show that the three models exhibit different trade-offs between inference latency, RAM usage, and Flash memory usage, highlighting the varying computational and memory requirements of each architecture. The custom CNN model demonstrated its applicability for edge devices with limited resources by achieving the lowest inference time (67 ms) and requiring the smallest flash memory footprint (72.4 KB). On the other hand, with an inference time of 121 ms, RAM utilization of 300.3 KB, and flash memory consumption of 542 KB, MobileViT required significantly more compute and memory resources. Additionally, MobileNetV1 demonstrated the effectiveness of its lightweight transfer learning architecture with lower inference time (71 ms) and lower RAM (125 KB).

These findings are consistent with the additional computational complexity associated with the transformer-based MobileViT architecture, which required higher RAM and Flash memory than the other models. Overall, the deployment results highlight the trade-off between classification performance and computational efficiency when selecting lightweight deep learning models for TinyML applications.

Figure D.7 illustrates representative on-device inference results of the deployed CNN model on the Arduino Portenta H7. The successful prediction of both healthy and diseased maize leaf images, together with the reported confidence scores and execution time, demonstrates that the TensorFlow Lite model retained its classification capability after deployment. These results confirm the feasibility of performing real-time maize leaf disease detection directly on the target microcontroller without relying on cloud-based

processing.

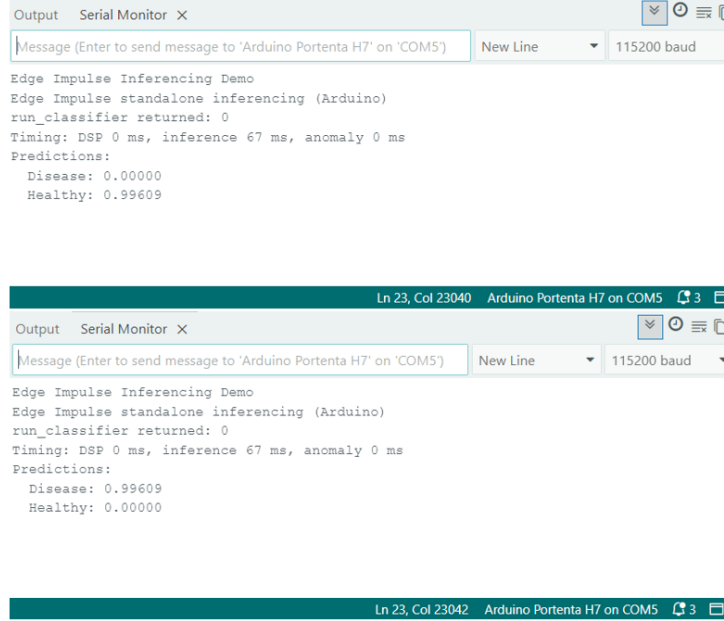


Figure D.7: On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images using CNN model; top image: healthy maize leaf, bottom image: disease maize leaf

D.2.2 Deployment Performance on Dataset2

This subsection evaluates the deployment performance of the three lightweight models on the Arduino Portenta H7 using Dataset2 and thereby providing an additional evaluation of the model. The on-device inference results of the TinyML models are shown in Table D.8.

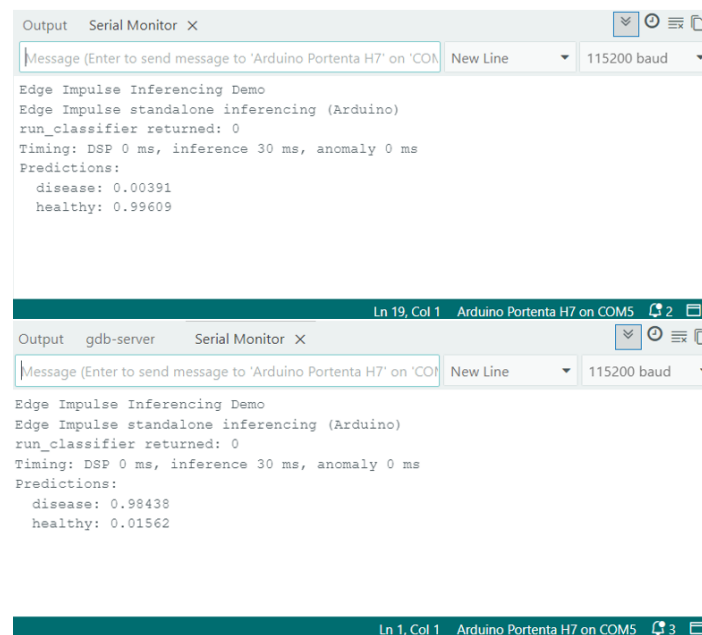
Table D.8: On-device Performance of Deployed Models on dataset2

Model Name	Latency (ms)	RAM (KB)	Flash (KB)
CNN	143	115.3	106.4
MobileViT	105	215.7	471
MobileNetV1	30	75.9	368.6

The deployment performance of the CNN, MobileViT, and MobileNetV1 models on Dataset 2 using the Arduino Portenta H7 is summarised in Table D.8. The findings show that the three architectures differ noticeably in terms of inference performance and memory usage. MobileNetV1 demonstrated its computational efficiency during deployment by achieving the

lowest inference time (30 ms) and requiring the least amount of RAM (75.9 KB). CNN, on the other hand, required a moderate amount of memory, but showed a longer inference time (143 ms). Due to its more complex design, MobileViT had the highest RAM usage (215.7 KB) despite having intermediate inference performance (105 ms).

In terms of flash memory usage, MobileNetV1 and MobileViT required much higher flash memory allocations of 368.6 KB and 471 KB, respectively, whereas CNN needs the smallest ROM footprint (106.4 KB). These results emphasise the trade-offs between memory requirements and inference performance in TinyML implementation. More complex feature extraction techniques may require more hardware resources, even though lightweight designs can reduce inference latency and memory usage.



```

Output  Serial Monitor X
Message (Enter to send message to 'Arduino Portenta H7' on 'COM5') New Line 115200 baud

Edge Impulse Inferencing Demo
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 0 ms, inference 30 ms, anomaly 0 ms
Predictions:
  disease: 0.00391
  healthy: 0.99609

Ln 19, Col 1  Arduino Portenta H7 on COM5

Output  gdb-server  Serial Monitor X
Message (Enter to send message to 'Arduino Portenta H7' on 'COM5') New Line 115200 baud

Edge Impulse Inferencing Demo
Edge Impulse standalone inferencing (Arduino)
run_classifier returned: 0
Timing: DSP 0 ms, inference 30 ms, anomaly 0 ms
Predictions:
  disease: 0.98438
  healthy: 0.01562

Ln 1, Col 1  Arduino Portenta H7 on COM5

```

Figure D.8: On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images using MobileNetV1 model; top image: healthy maize leaf, bottom image: disease maize leaf

Figure D.8 presents representative on-device inference results of the deployed MobileNetV1 model on the Arduino Portenta H7 using Dataset2. The successful classification of both healthy and diseased maize leaf images demonstrates that the deployed model maintained its predictive performance under real-time embedded execution. These results further con-

firm the practical feasibility of deploying lightweight deep learning models for on-device crop disease detection in resource-constrained precision agriculture applications.

Overall, the deployment results demonstrate that all three models can be successfully implemented on the Arduino Portenta H7 for real-time inference. Although the MobileViT model achieved the highest classification accuracy on Dataset1, this improvement was accompanied by increased inference latency and memory requirements. In contrast, MobileNetV1 provided the best deployment efficiency while maintaining competitive classification performance. These findings highlight the trade-off between predictive performance and computational efficiency when selecting lightweight deep learning models for TinyML-based crop disease detection.

D.3 Comparative Analysis of the Proposed Models

The performance characteristics of the proposed CNN, MobileViT, and MobileNetV1 models for maize leaf disease detection were assessed by a comparative analysis. The comparison considered classification metrics such as accuracy, precision, recall, F1-score, false positives, and false negatives, as well as deployment-related factors, including inference time and memory requirements. The objective of this analysis is to evaluate the trade-offs between classification performance and deployment efficiency, thereby identifying the most appropriate model for different TinyML deployment scenarios rather than selecting a single best-performing architecture.

D.3.1 Analysis Using Dataset1

The comparative classification performance is shown in Table D.9. The findings show that all three models successfully identified healthy and diseased leaf samples, achieving good classification performance for the detection of maize leaf disease. Each architecture is able to provide reliable predictions across the assessed datasets. While recall shows how well the models iden-

tify diseased maize leaves, precision shows how well they can reduce false disease detections. The high F1-scores reflect reliable classification performance in a variety of image conditions, and a balanced performance between precision and recall also maintains this.

Table D.9: Performance Comparison of Three Architectures on Dataset1

Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	False Positive (%)	False Negative (%)
CNN	91.90±1.59	92.23±1.37	91.90±1.59	91.88±1.61	6.0	8.33
MobileViT	96.07±1.07	96.12±1.07	96.07±1.07	96.07±1.07	2.0	3.00
MobileNetV1	90.37±1.03	90.83±0.43	90.37±1.03	90.33±1.08	6.67	9.33

The efficiency of the models can be verified again by analysing false positives and false negatives. False negatives are diseased leaves that are mistakenly classified as healthy, while false positives are healthy leaves that are incorrectly labeled as diseased. False negatives are particularly critical in crop disease detection because failing to identify diseased plants may delay treatment, increase disease spread, and ultimately reduce crop yield. The relatively low false negative rates observed for the MobileViT model compared to CNN and MobileNetV1 demonstrate the model's ability to reliably identify diseased maize leaves, making it suitable for early disease detection in precision agriculture.

Despite all three lightweight models demonstrating good classification performance, MobileNetV1 showed a modest reduction in accuracy, precision, recall, F1-score, false positives and false negatives compared to the custom CNN and MobileViT models. This suggests that, compared with the transfer learning-based MobileNetV1 model, the custom architectures can develop more discriminative task-specific features for the binary maize leaf disease classification. MobileNetV1 showed favorable deployment characteristics regardless of a small reduction in classification performance. In Table D.7, it has been demonstrated that the inference time of the MobileNetV1 model was marginally longer than the custom CNN but faster than the

MobileViT model. More significantly, due to its lightweight design, MobileNetV1 required the lowest RAM among the three architectures. These results imply that MobileNetV1 provides a useful trade-off between computational efficiency and classification performance, making it a viable option for memory-constrained TinyML applications where hardware resources are more limited than maximum accuracy demands.

D.3.2 Analysis Using Dataset2

The comparative classification performance of the CNN, MobileViT, and MobileNetV1 models on Dataset2 is shown in Table D.10. The results show that all three models achieved satisfactory accuracy, precision, recall, and F1-score, indicating their ability to classify images of healthy and diseased maize leaves. The proposed models appear to have good generalisation capabilities and can consistently extract discriminative features from images collected across various sources, as evidenced by their consistently high performance across all evaluation measures.

Table D.10: Performance Comparison of Three Architectures on Dataset2

Architecture	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	False Positive (%)	False Negative (%)
CNN	98.58±0.52	98.62±0.48	98.58±0.52	98.58±0.52	0.86	1.1
MobileViT	98.28±0.81	98.33±0.75	98.28±0.81	98.28±0.81	0.65	1.1
MobileNetV1	98.97±0.25	98.97±0.25	98.97±0.25	98.97±0.25	0.65	0.86

Since a few percent of healthy leaves were mistakenly classified as diseased and a small number of diseased leaves were mistakenly classified as healthy, the false positive and false negative values further demonstrate the efficacy of the proposed models. The low false negative rates observed on Dataset2 further reinforce the suitability of the proposed models for early crop disease detection, consistent with the findings for Dataset1.

Another observation is that the improvements in accuracy, precision, re-

call, and F1-score on Dataset2 demonstrate that all three lightweight models performed better in classification than on Dataset1. This suggests that the models were able to classify healthy and diseased maize leaves more accurately by learning more discriminative feature representations from Dataset2. Differences in image attributes, such as more consistent disease symptoms, better image quality, less background variability, or a sharper visual contrast between the two classes, might be the cause of the higher performance. Additionally, false-positive and false-negative rates in dataset2 were significantly lower, suggesting fewer classification errors.

The findings from Dataset2 show that the three lightweight models performed very similarly in terms of classification performance. This implies that despite variability in the dataset, all models can learn strong discriminative features. However, as shown in Table D.8, a more noticeable difference was found in their deployment characteristics. MobileNetV1's computational efficiency for edge devices with limited resources was demonstrated by its faster inference latency and much lower RAM requirements compared to the custom CNN and MobileViT models. The custom models, on the other hand, needed more processing power during deployment. These results demonstrate that deployment practicality becomes a crucial factor when classification performance is similar, with MobileNetV1 presenting a notable advantage in terms of computational efficiency and memory usage for TinyML applications.

Overall, the comparative analysis demonstrates that no single model is optimal for all TinyML deployment scenarios. The custom CNN and MobileViT models generally provide superior classification performance, whereas the transfer learning-based MobileNetV1 model offers the best deployment efficiency in terms of inference latency and memory usage. Consequently, the choice of model depends on the relative importance of predictive performance and computational resource constraints for the intended precision agriculture application.

D.4 Validation with Edge Impulse

To independently validate the proposed TinyML framework, a lightweight optimised CNN model was developed and evaluated using the Edge Impulse platform. Unlike the previous Python-based TensorFlow Lite workflow, Edge Impulse provides an alternative end-to-end TinyML development and deployment environment. Although the implementation is limited to Dataset1, it provides an independent validation of the proposed framework by demonstrating that strong classification performance and successful deployment can be achieved using a different TinyML development workflow.

D.4.1 Classification Performance of the Model

As shown in Figure D.9, the model achieves high accuracy, with training and validation accuracies of 97.34% and 95%, respectively. The resilience of the model is demonstrated by the consistency between the training data and the unknown data. At Epoch 19, the training and validation losses reached their lowest values of 0.0961 and 0.1532, respectively, implying the best model performance and convergence.

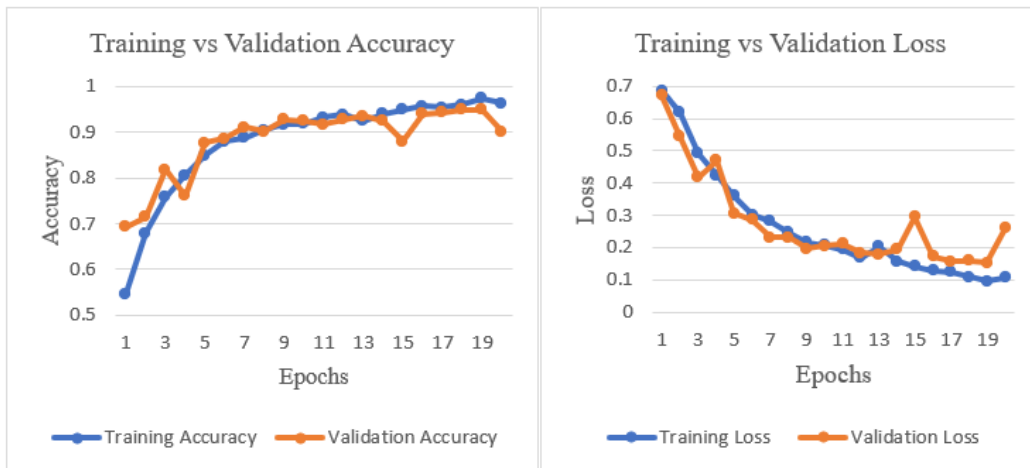


Figure D.9: Training vs Validation Accuracy and Loss

At Epoch 20, there was a small increase in both losses, indicating that further training only marginally improved the model and might have some-

what diminished its generalization capacity. However, overall performance remained the same, indicating that the 20-epoch training environment was suitable.

The testing procedure was made easier by the Edge Impulse platform, which reported the model's performance metrics: 96.50% test accuracy and a test loss of 0.14. This stage assessed the predictive power of the trained machine learning model using new data. While accuracy reflects the percentage of correct predictions, test loss indicates how far the model deviates from the predicted outcomes. These measures demonstrate an appropriate degree of accuracy in the Edge Impulse test and serve as a standard for evaluating the model's efficacy and generalization to unseen data.

The trained model was evaluated for both disease and healthy image classes using a confusion matrix and a few performance metrics, including precision, recall, and F1 score. The confusion matrix (Figure D.10) indicates that while the model correctly identified 295 diseased images and 284 healthy images, it misidentified 5 diseased images as healthy and 16 healthy images as diseased. The low number of false negatives further supports the suitability of the proposed framework for early crop disease detection.

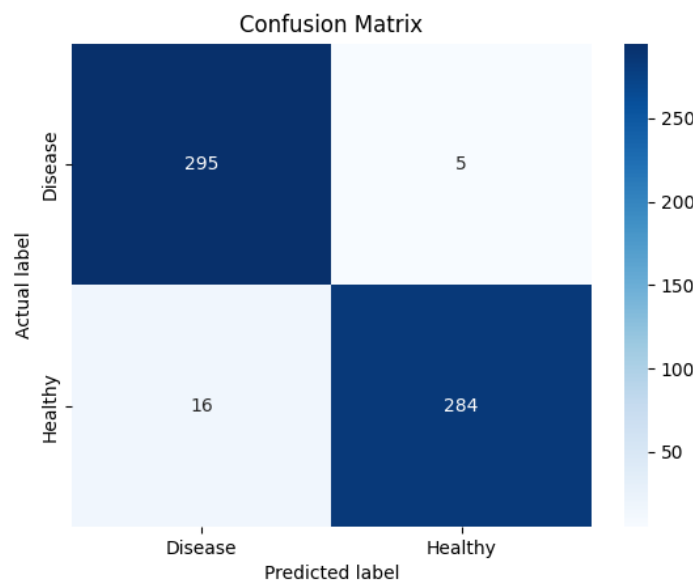


Figure D.10: Confusion Matrix of the test dataset

Table D.11: Performance metrics of the model evaluated in Edge Impulse

Metric	Diseased Images (%)	Healthy Images (%)
Precision	97.00	99.00
Recall	98.00	95.00
F1 Score	98.00	97.00

The efficiency of the model in classifying maize leaves is further demonstrated by the class-wise assessment of the metrics shown in Table D.11. The model can correctly identify diseased leaves while reducing false detections, as seen by the diseased class's precision of 97%, recall of 98 %, and F1- score of 98%. Similarly, the model's ability to identify healthy leaves was confirmed by the healthy class's precision of 99%, recall of 95%, and F1-score of 97%. The robustness and dependability of the model for maize leaves disease detection are demonstrated by the consistently high precision, recall, and F1-score values across both classes.

D.4.2 Deployment Feasibility of the Model

The deployment features of the model developed by Edge Impulse on the Arduino Portenta H7 are shown in Table D.12. The model can be implemented within the hardware limitations of the target edge device, as seen by its 156 ms inference time and 186.6 KB RAM and 80.0 KB flash memory requirements. The model's memory requirements are comparatively low, despite inference latency being higher than that of some lightweight architectures. These findings show that using the proposed TinyML model for real-time maize leaf disease classification on embedded hardware is practically feasible.

Table D.12: On-device Performance of Deployed Model

Model Name	Latency (ms)	RAM (KB)	Flash (KB)
CNN	156	186.6	80.0

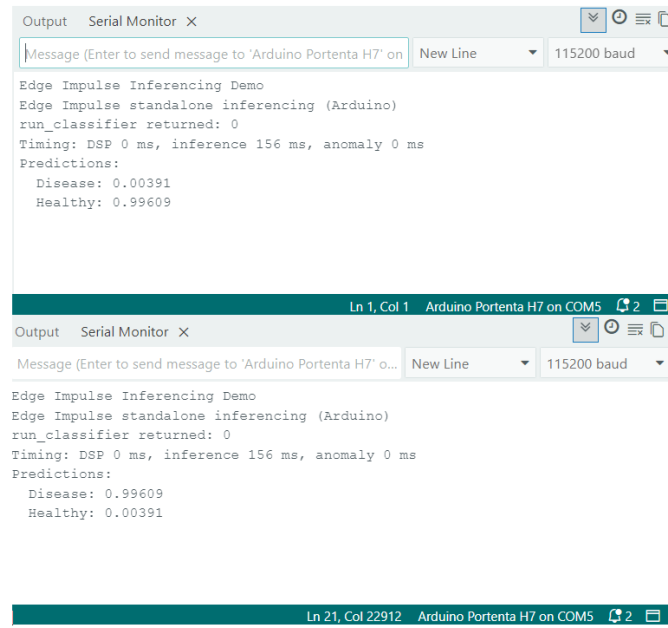


Figure D.11: On-device inference results on Arduino Portenta H7 for binary classification of maize leaf images; top image: healthy maize leaf, bottom image: disease maize leaf

The on-device inference results of the Edge Impulse-developed model on the Arduino Portenta H7 for both healthy and diseased maize leaf images are shown in Figure D.11. The picture shows that both image categories were successfully categorized by the deployed model, which also provided the associated confidence scores in real time. These outcomes further show that the suggested TinyML model for real-time maize leaf disease classification is practically feasible. Thus, the study's robustness is strengthened by the evaluation conducted using the Edge Impulse platform, which offers independent validation of the proposed methodology.

Although the Edge Impulse implementation was evaluated only on Dataset1, its successful deployment and comparable classification performance provide independent validation of the proposed TinyML framework using a different development and deployment workflow.

D.5 Overall Discussion and Research Implications

The successful implementation of the proposed TinyML models on the Arduino Portenta H7 demonstrated both the technical viability and the useful

potential of edge-based maize leaf disease identification. The proposed approach enables real-time disease identification without cloud computing or high-performance computing infrastructure by combining high classification performance with efficient resource use. This skill is especially useful for precision agriculture, where prompt disease identification can enhance crop management and promote well-informed decision-making. According to the study's findings, lightweight deep learning models deployed on edge devices may offer a practical and affordable way to diagnose maize diseases early, particularly in resource-constrained settings or with unreliable internet connectivity.

D.5.1 Key Observations and Research Implications

1. Effect of Low Image Resolution (48×48 pixels)

The classification performance and deployment viability of TinyML models are significantly influenced by image resolution. This study has shown that the mean accuracy of the MobilnetV1 model for Dataset1 is $90.37 \pm 1.03\%$. However, classification performance improved as image resolution increased, with mean accuracies of $93.33 \pm 0.88\%$ and $95.73 \pm 1.15\%$ at 64×64 and 96×96 pixels, respectively, using the same architecture. The retention of finer visual cues, such as lesion boundaries, texture patterns, and minor color variations, which provide more discriminative information to classify maize leaves, accounts for this improvement.

Higher image resolutions boosted classification performance, but also increased memory and computational complexity, making deployment on edge devices more difficult. On the other hand, the 48×48 pixel resolution reduced computational overhead, leading to lower memory usage and faster inference—both of which are crucial for deployment on the Arduino Portenta H7. These results demonstrate the trade-off between deployment efficiency and classification performance, suggesting that a small decrease in accuracy is a reasonable trade-off to achieve effective real-time TinyML deployment. This trade-off is a fundamental characteristic of TinyML system design, where modest reductions in classification performance are often

accepted to achieve substantial improvements in computational efficiency, memory usage, and energy consumption on resource-constrained edge devices.

The computational constraints of the TinyML deployment have led to the choice of a lower image resolution, but this does not limit the practical application of the proposed system in real-world scenarios. In a real-world application, users would use a smartphone or an embedded camera to take pictures of maize leaves at their native resolution. Before classification, the application software would automatically preprocess the taken image by scaling it to the input dimensions required by the trained TinyML model (48×48 pixels) and normalising the pixel values. To ensure that each input image is compatible with the deployed model, this preprocessing phase can be incorporated into the program and performed automatically. Because the complete preparation pipeline would run automatically before inference, users would not need to resize images manually.

2. Misclassification assessment

Although all three models performed well in classification, there were a few false positive and false negative predictions. The observed misclassifications are more likely attributable to challenging image characteristics than to limitations of the proposed model architectures. Misclassification may have been caused by a number of factors, including early-stage disease symptoms that closely resemble healthy leaves, comparable color and texture patterns between healthy and diseased tissues, fluctuations in illumination, shadows, camera views, and complex background conditions.

Representative images of incorrectly classified maize leaves from the proposed classification models are shown in Figure D.12. The images in the top row depict healthy leaves that were mistakenly identified as diseased (false positives), while the images in the bottom row depict diseased leaves that were mistakenly identified as healthy (false negatives). Shadows, uneven lighting, slight colour changes, and naturally occurring leaf textures that mimic diseased symptoms are the main causes of the false-positive instances.



Figure D.12: Examples of misclassification images; top row: healthy leaves misclassified as diseased, bottom row: diseased leaves misclassified as healthy

On the other hand, shadows and leaves with modest or early-stage diseased symptoms—lesions that are tiny, sparsely scattered, or have little visual contrast with the surrounding healthy tissue—are the main causes of false-negative instances. Additionally, minor lesion details were definitely lost when the input images were resized to 48×48 pixels, making it harder to identify delicate disease characteristics. By taking pictures in suitable lighting, avoiding shadows, and ensuring visibility, these misclassifications can be minimised in real-world agricultural applications. The quality of the images and the accuracy of the real-time disease classification would both be improved by such picture-capturing techniques.

3. Findings from two distinct datasets

Two publicly accessible datasets were used to evaluate the suggested models across various image acquisition scenarios. A thorough evaluation of the models' capacity to generalise across various imaging settings is made feasible by the two datasets, which comprise images with noticeably diverse background features. The CNN, MobileViT, and MobileNetV1 models demonstrated that they learned resilient and discriminative features rather than becoming specialised for a single dataset, as evidenced by their

consistent good classification performance. These results support the practical usefulness of the proposed models in precision agriculture by indicating that they can retain dependable performance when given images of maize leaves captured under various real-world settings.

4. Trade-off Between Classification Performance and Deployment Feasibility

Despite the fact that all three models perform well in classification, their deployment characteristics varied in terms of ROM footprint, RAM utilization, and inference time. While MobileNetV1 showed effective memory usage and quick inference, the CNN model required a comparatively modest flash memory footprint. However, MobileViT required more memory resources due to its more complex architecture. Additionally, the choice of 48×48 input image resolution represented a significant design trade-off, whereas significant gains in memory utilisation, computational complexity, and real-time inference capability were achieved at the expense of a slight decrease in classification performance. These findings suggest that choosing a suitable TinyML model should take into account deployment constraints, such as memory needs, inference speed, and hardware limits, in addition to prediction accuracy. The relative importance of these factors depends on the target application. For example, applications requiring rapid real-time inference or deployment on highly resource-constrained devices may prioritize computational efficiency and memory usage, whereas applications where early disease detection is critical may favor higher classification performance despite the additional computational cost. To ensure dependable and useful implementation in actual precision agriculture applications, an efficient TinyML solution must strike a balance between classification performance and deployment viability.

5. Effect of Quantisation on Classification Performance

To assess the effect of INT8 quantization on classification performance, the full-precision (FP32) and quantised (INT8) versions of the three models were evaluated using the same test images. McNemar's test was applied to their paired predictions to determine whether quantisation resulted in sta-

tistically significant changes in classification outcomes, using a significance level of $\alpha = 0.05$. Tables D.13 and D.14 summarise the results for Dataset1 and Dataset2, respectively.

Table D.13: Comparison of Full-Sized (FP32) and INT8-quantised Model Accuracy and Statistical Significance for Dataset1

Model	Full-sized Model Accuracy (%)	TinyML Accuracy (%)	Accuracy Reduction (pp)	McNemar's p-value	Significant Difference ($\alpha = 0.05$)
CNN	91.9	91.9	0	1.00	No
MobileViT	96.07	94.82	1.25	0.08561	No
MobileNetV1	90.37	89.20	1.17	0.09145	No

Table D.14: Comparison of Full-Sized (FP32) and INT8-quantised Model Accuracy and Statistical Significance for Dataset2

Model	Full-sized Model Accuracy (%)	TinyML Accuracy (%)	Accuracy Reduction (pp)	McNemar's p-value	Significant Difference ($\alpha = 0.05$)
CNN	98.58	98.58	0	1.00	No
MobileViT	98.28	97.17	1.11	0.093234	No
MobileNetV1	98.97	97.53	1.44	0.072912	No

Across both datasets, INT8 quantisation resulted in relatively small changes in classification accuracy. The CNN retained its original accuracy on both datasets, while MobileViT and MobileNetV1 experienced reductions ranging from 1.11 to 1.44 percentage points. McNemar's test found no statistically significant differences between the full-precision and quantised predictions for any model on either dataset ($p > 0.05$). These findings suggest that INT8 quantisation preserved much of the predictive performance of the original models, although the absence of statistical significance does not establish that the two versions are equivalent.

6. Trade-offs Between Model Size and Classification Performance

The preceding analysis showed that INT8 quantisation produced relatively small reductions in classification accuracy, with no statistically significant differences detected between the full-precision and quantised mod-

els. However, classification performance alone does not determine suitability for TinyML deployment. Model storage requirements are particularly important for resource-constrained microcontrollers, where available Flash memory may limit the complexity of deployable models. This section therefore examines the reduction in model size achieved through INT8 quantisation and its implications for embedded deployment.

Table D.15: Comparison of Full-Sized and INT8-Quantised Model Sizes for Dataset1

Model	Full-sized Model Size (KB)	TinyML Model Size (KB)	Model Size Reduction (%)
CNN	90	22	75.56%
MobileViT	666	132	80.18%
MobileNetV1	1242	209	83.17%

Table D.16: Comparison of Full-Sized and INT8-Quantised Model Sizes for Dataset2

Model	Full-sized Model Size (KB)	TinyML Model Size (KB)	Model Size Reduction (%)
CNN	221	56	74.66%
MobileViT	702	135	80.77%
MobileNetV1	1232	309	74.92%

As shown in Tables D.15 and D.16, INT8 quantisation reduced model sizes by 75.56%–83.17% for Dataset1 and 74.66%–80.77% for Dataset2. These substantial reductions demonstrate the effectiveness of quantisation in reducing the storage requirements of all three architectures. Notably, MobileViT achieved the greatest relative reduction on Dataset2, while MobileNetV1 achieved the greatest reduction on Dataset1. Although the absolute model sizes differed across architectures and datasets, all three benefited substantially from quantisation.

When considered alongside the classification results, these findings highlight the effectiveness of INT8 quantization for TinyML deployment. Sub-

stantial reductions in model size were achieved with relatively small losses in predictive accuracy. The custom CNN was particularly resilient to quantisation, retaining its classification accuracy on both datasets while achieving considerable storage savings. Although MobileViT and MobileNetV1 experienced modest accuracy reductions, their substantially smaller quantised models offer important advantages for deployment on memory-constrained devices. These results demonstrate that the benefits of quantisation depend on the model architecture and should be evaluated in terms of both predictive performance and deployment requirements.

7. Practical Implications for Precision Agriculture

The successful implementation of the proposed TinyML models on the Arduino Portenta H7 demonstrates the feasibility of real-time crop disease detection on resource-constrained edge devices. Binary maize leaf disease detection serves as a proof-of-concept for the proposed TinyML framework. The proposed approach decreases communication latency, lessens reliance on cloud computing, and facilitates operation in settings with unreliable internet connectivity by enabling on-device inference. These capabilities demonstrate the potential of lightweight deep learning models to support practical, low-cost crop disease detection in precision agriculture, particularly in regions with limited computational resources or unreliable internet connectivity.

Although this study focuses on maize leaf disease detection, the proposed TinyML framework can be readily adapted to other image-based crop disease detection applications through retraining with crop-specific datasets, providing a foundation for broader precision agriculture solutions. Overall, the study's results show that lightweight deep learning models can reliably classify maize leaves while still being suitable for use on edge devices. The deployment results demonstrated the trade-off between classification performance and computational efficiency, which is essential to TinyML applications, while the evaluation using two publicly available datasets confirmed the proposed models' capacity to generalise across different image sources and background conditions. Additionally, validation of the pro-

posed framework under a different development and deployment context by the Edge Impulse platform boosted confidence in its resilience and usefulness.

E. Conclusion

This study developed and evaluated lightweight deep learning models for TinyML-based maize leaf disease detection on resource-constrained edge devices. Binary maize leaf disease detection served as a proof-of-concept for the proposed TinyML framework. Two publicly available maize leaf image datasets were used to develop and assess three lightweight architectures: a custom Convolutional Neural Network (CNN), a custom transformer-based MobileViT model, and a transfer learning based MobileNetV1 model. After training and optimisation in TensorFlow, the models were evaluated using 5-fold cross-validation and standard classification metrics, including accuracy, precision, recall, F1-score, false positives, and false negatives. The trained models were implemented on the Arduino Portenta H7, and deployment performance was evaluated in terms of inference time, RAM utilisation, and flash memory consumption to determine their practical suitability for TinyML applications.

The experimental results showed that all three lightweight models achieved high classification performance while exhibiting different trade-offs between predictive accuracy, inference latency, and memory requirements. The custom CNN and MobileViT models generally achieved superior classification performance, whereas the transfer learning-based MobileNetV1 model offered the best deployment efficiency. Independent validation using the Edge Impulse platform further demonstrated the robustness of the proposed TinyML framework under an alternative development workflow.

The three research questions posed in this study were successfully addressed. The first research question established that all three lightweight models achieved strong classification performance for binary maize leaf disease detection. The second demonstrated that the custom CNN and MobileViT models generally outperformed the transfer learning-based MobileNetV1 model in classification performance, while MobileNetV1 offered superior deployment efficiency. The third showed that model selection for TinyML applications requires balancing predictive performance with com-

putational and memory constraints.

The principal contributions of this study are threefold. First, it provides a comprehensive comparative evaluation of three lightweight deep learning architectures for binary maize leaf disease detection. Second, the study confirms that accurate disease detection can be achieved on a resource-constrained edge device with reasonable computing and memory requirements by demonstrating the viability of implementing these models on the Arduino Portenta H7. By testing the deployment process using a separate TinyML development platform, the addition of another Edge Impulse-based model improves the robustness of the proposed approach.

In practice, the proposed TinyML framework could enable real-time crop disease diagnostics directly on embedded devices without relying on cloud-based processing. In agricultural settings with unreliable or inconsistent internet connectivity, this facilitates quick decision-making, reduces communication latency, and allows reliable operation. Additionally, the results provide useful details on the trade-off between deployment efficiency and classification performance, which can help professionals and researchers select suitable lightweight models for future edge-based precision agriculture applications.

Beyond the specific application of maize leaf disease detection, this research demonstrates how lightweight model architectures and quantisation can extend the range of deep learning approaches that are computationally feasible on resource-constrained microcontrollers. The successful deployment of CNN, transformer-based MobileViT, and MobileNetV1 architectures on the Arduino Portenta H7 shows that embedded inference need not be restricted to conventional CNN-based approaches. By achieving substantial reductions in model storage requirements while largely preserving classification accuracy, INT8 quantisation further demonstrates how computational constraints can be addressed through model optimisation. Collectively, these findings establish a practical foundation for extending TinyML applications to more complex agricultural classification tasks and for investigating deployment on devices with tighter computational and memory

constraints.

Despite the proposed TinyML framework showing promising classification and deployment performance, this study has some limitations. First, the models do not identify disease types because they were designed for binary classification of maize leaf images, just differentiating between healthy and diseased leaves. Second, to meet the computational limitations of TinyML deployment, all models were trained at a comparatively low image resolution (48×48 pixels). Although this made it possible for the target edge device to execute efficiently, some fine visual features may have been lost during image resizing, potentially affecting the classification of visually challenging samples. Additionally, two publicly accessible datasets were used for the experimental evaluation, which may not accurately reflect the variety of actual agricultural situations. Lastly, only the Arduino Portenta H7 microcontroller was used to evaluate deployment performance; the results may vary if other edge computing platforms with differing hardware capabilities are used. Despite these drawbacks, the study offers a thorough assessment of lightweight deep learning models for TinyML-based maize leaf disease identification and lays an effective foundation for future edge-based precision agriculture research.

Although this study demonstrated that using lightweight deep learning models for TinyML-based maize leaf disease classification is feasible, there are still several possibilities for further investigation. First, rather than just classifying healthy and diseased leaves, future research may expand the proposed framework from binary classification to multi-class disease classification, enabling the detection of specific maize leaf diseases. The viability of multi-class classification has been demonstrated in prior research using the same publicly available dataset, suggesting that the proposed TinyML framework could be further modified to enable more comprehensive disease diagnosis while remaining deployable on resource-constrained edge devices.

Another major area is the exploration of higher image resolutions and more advanced lightweight architectures. Future research may examine

adaptive image resolutions or effective preprocessing techniques that preserve more visual detail without significantly increasing computational requirements, even though the chosen image resolution of 48×48 pixels offered an effective balance between classification performance and deployment efficiency. In addition, recently proposed lightweight transformer-based and hybrid neural network models may increase classification accuracy while maintaining compatibility with TinyML platforms.

Future research should further evaluate the proposed methodology using larger and more diversified datasets collected under different environmental circumstances, lighting fluctuations, camera perspectives, and locations. Incorporating images of diverse crop species and new plant diseases would provide a broader examination of the robustness of the model and the generalisability of the model. Furthermore, deployment on multiple embedded platforms, such as alternative Arduino boards, STM32 microcontrollers, or ESP32-based systems, would enable a more comprehensive evaluation of hardware compatibility and deployment efficiency across different resource-constrained devices.

Lastly, a significant direction for future research is practical field validation of the proposed TinyML framework. This would involve developing an end-to-end embedded system that integrates camera acquisition, automated image preprocessing, and on-device inference for fully autonomous crop disease detection. Integrating the proposed framework with the Internet of Things (IoT)-based smart farming applications or portable agricultural monitoring systems would enable real-time disease diagnosis in practical field conditions. Such developments could support timely decision-making, facilitate early disease management, and promote the wider adoption of low-cost intelligent precision agriculture solutions.

In conclusion, this study shows that lightweight deep learning models can feasibly perform TinyML-based crop disease detection on an edge device. Using binary maize leaf disease detection as a proof of concept, the proposed TinyML framework shows that effective edge AI solutions require balancing predictive performance with deployment constraints.

F. References

- [1] A. H. Mohd Javaid, I. H. Khan, and R. Suman, "Understanding the potential applications of artificial intelligence in agriculture sector," *Advanced Agrochem*, vol. 2, no. 1, pp. 15–30, 2023.
- [2] A. Ehrampoosh et al., "Intelligent weed management using aerial image processing and precision herbicide spraying: An overview," *Crop Protection*, p. 107 206, 2025, ISSN: 0261-2194.
- [3] M. Zain et al., "Nanotechnology in precision agriculture: Advancing towards sustainable crop production," *Plant Physiology and Biochemistry*, vol. 206, p. 108 244, 2024, ISSN: 0981-9428.
- [4] T. Jin and X. Han, "Robotic arms in precision agriculture: A comprehensive review of the technologies, applications, challenges, and future prospects," *Computers and Electronics in Agriculture*, vol. 221, p. 108 938, 2024, ISSN: 0168-1699.
- [5] B. Petrović, R. Bumbálek, T. Zoubek, R. Kuneš, L. Smutný, and P. Bartoš, "Application of precision agriculture technologies in central europe-review," *Journal of Agriculture and Food Research*, vol. 15, p. 101 048, 2024, ISSN: 2666-1543.
- [6] R. Chin, C. Catal, and A. Kassahun, "Plant disease detection using drones in precision agriculture," *Precision Agriculture*, vol. 24, no. 5, pp. 1663–1682, 2023, ISSN: 1385-2256.
- [7] S. Perez, N. Dilshad, N. S. Alghamdi, T. M. Alanazi, and J. W. Lee, "Visual intelligence in precision agriculture: Exploring plant disease detection via efficient vision transformers," *Sensors*, vol. 23, no. 15, p. 6949, 2023, ISSN: 1424-8220.
- [8] V. Meshram, K. Patil, V. Meshram, D. Hanchate, and S. Ramkteke, "Machine learning in agriculture domain: A state-of-art survey," *Artificial Intelligence in the Life Sciences*, vol. 1, p. 100 010, 2021.
- [9] L. Benos, A. C. Tagarakis, G. Dolias, R. Berruto, D. Kateris, and D. Bochtis, "Machine learning in agriculture: A comprehensive updated review," *Sensors*, vol. 21, no. 11, p. 3758, 2021.

-
- [10] S. F. Ahmed et al., "Deep learning modelling techniques: Current progress, applications, advantages, and challenges," *Artificial Intelligence Review*, vol. 56, no. 11, pp. 13 521–13 617, 2023.
 - [11] R. Rashid, W. Aslam, R. Aziz, and G. Aldehim, "An early and smart detection of corn plant leaf diseases using iot and deep learning multi-models," *IEEE Access*, vol. 12, pp. 23 149–23 162, 2024, ISSN: 2169-3536.
 - [12] K. Sunkara, B. Sarada, G. Sahitya, C. Rajasekhar, and C. Gowthami, "Lapgraph-cutfs: A deep learning-based framework for multi-class classification of maize leaf pests, diseases, and healthy states," *IEEE Access*, 2026, ISSN: 2169-3536.
 - [13] N. N. Alajlan and D. M. Ibrahim, "Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications," *Micromachines*, vol. 13, no. 6, p. 851, 2022, ISSN: 2072-666X.
 - [14] S. Bhattacharya and M. Pandey, "Deploying an energy efficient, secure high-speed sidechain-based tinyml model for soil quality monitoring and management in agriculture," *Expert Systems with Applications*, vol. 242, p. 122 735, 2024, ISSN: 0957-4174.
 - [15] A. Sharma, A. Jain, P. Gupta, and V. Chowdary, "Machine learning applications for precision agriculture: A comprehensive review," *IEEE Access*, vol. 9, pp. 4843–4873, 2020, ISSN: 2169-3536.
 - [16] M. Padhiary, D. Saha, R. Kumar, L. N. Sethi, and A. Kumar, "Enhancing precision agriculture: A comprehensive review of machine learning and ai vision applications in all-terrain vehicle for farm automation," *Smart Agricultural Technology*, p. 100 483, 2024, ISSN: 2772-3755.
 - [17] Y. Akkem, S. K. Biswas, and A. Varanasi, "Smart farming using artificial intelligence: A review," *Engineering Applications of Artificial Intelligence*, vol. 120, p. 105 899, 2023, ISSN: 0952-1976.
 - [18] R. Akhter and S. A. Sofi, "Precision agriculture using iot data analytics and machine learning," *Journal of King Saud University-Computer*

- and Information Sciences*, vol. 34, no. 8, pp. 5602–5618, 2022, ISSN: 1319-1578.
- [19] Z. Lv et al., “Efficient deployment of peanut leaf disease detection models on edge ai devices,” *Agriculture*, vol. 15, no. 3, p. 332, 2025, ISSN: 2077-0472.
 - [20] M.-Y. Kim and K. H. Lee, “Electrochemical sensors for sustainable precision agriculture—a review,” *Frontiers in chemistry*, vol. 10, p. 848 320, 2022, ISSN: 2296-2646.
 - [21] K. Sharma and S. K. Shivandu, “Integrating artificial intelligence and internet of things (iot) for enhanced crop monitoring and management in precision agriculture,” *Sensors International*, p. 100 292, 2024, ISSN: 2666-3511.
 - [22] R. Srinivasagan, M. Mohammed, and A. Alzahrani, “Tinyml-sensor for shelf life estimation of fresh date fruits,” *Sensors*, vol. 23, no. 16, p. 7081, 2023, ISSN: 1424-8220.
 - [23] D. A. N. Gookyi et al., “Tinyml for smart agriculture: Comparative analysis of tinyml platforms and practical deployment for maize leaf disease identification,” *Smart Agricultural Technology*, vol. 8, p. 100 490, 2024, ISSN: 2772-3755.
 - [24] S. Maniam, Y.-K. Tee, E. Memar, H. Wong, and M. Zaman, “Smart irrigation management: Iot-based rnn-lstm model for soil moisture prediction in precision agriculture,” *Smart Agricultural Technology*, p. 101 866, 2026.
 - [25] A. S. Rathor, S. Choudhury, A. Sharma, R. K. Pachauri, G. Shah, N. Fatema, et al., “Deep convolutional network for plant monitoring in precision agriculture,” *Sustainable Computing: Informatics and Systems*, p. 101 302, 2026.
 - [26] S. Y. Hammad, G. P. Kovács, and G. Milics, “Ambrosia artemisiifolia in hungary: A review of challenges, impacts, and precision agriculture approaches for sustainable site-specific weed management using uav technologies,” *AgriEngineering*, vol. 8, no. 1, p. 30, 2026.

- [27] A. A. Khoddamzadeh, S. Singh, and M. S. Balaji Bhaskar, *Optimizing fertilizer and irrigation for specialty crops using precision agriculture technologies*, 2026.
- [28] A. Rana and P. Vaidya, "Yolo-based deep learning framework for real-time multi-class plant health monitoring in precision agriculture," *Scientific Reports*, vol. 16, no. 1, p. 197, 2026.
- [29] S. Sudha and J. Loret, "A review on machine learning-based precision agriculture techniques for crop farming monitoring with iot," *Discover Environment*, vol. 4, no. 1, p. 10, 2026.
- [30] P. Sharma, P. Dadheech, N. Aneja, and S. Aneja, "Predicting agriculture yields based on machine learning using regression and deep learning," *IEEe Access*, vol. 11, pp. 111 255–111 264, 2023.
- [31] J. Botero-Valencia et al., "Machine learning in sustainable agriculture: Systematic review and research perspectives," *Agriculture*, vol. 15, no. 4, p. 377, 2025.
- [32] S. O. Araújo, R. S. Peres, J. C. Ramalho, F. Lidon, and J. Barata, "Machine learning applications in agriculture: Current trends, challenges, and future perspectives," *Agronomy*, vol. 13, no. 12, p. 2976, 2023, ISSN: 2073-4395.
- [33] S. Ghazal, A. Munir, and W. S. Qureshi, "Computer vision in smart agriculture and precision farming: Techniques and applications," *Artificial Intelligence in Agriculture*, 2024, ISSN: 2589-7217.
- [34] N. N. Alajlan and D. M. Ibrahim, "Tinyml: Enabling of inference deep learning models on ultra-low-power iot edge devices for ai applications," *Micromachines*, vol. 13, no. 6, p. 851, 2022, ISSN: 2072-666X.
- [35] F. Alpsalaz, Y. Özüpak, E. Aslan, and H. Uzel, "Classification of maize leaf diseases with deep learning: Performance evaluation of the proposed model and use of explicable artificial intelligence," *Chemometrics and Intelligent Laboratory Systems*, p. 105 412, 2025, ISSN: 0169-7439.
- [36] S. Khan and M. Narvekar, "Novel fusion of color balancing and superpixel based approach for detection of tomato plant diseases

- in natural complex environment," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 6, pp. 3506–3516, 2022, ISSN: 1319-1578.
- [37] S. U. Rahman, F. Alam, N. Ahmad, and S. Arshad, "Image processing based system for the detection, identification and treatment of tomato leaf diseases," *Multimedia Tools and Applications*, vol. 82, no. 6, pp. 9431–9445, 2023, ISSN: 1380-7501.
- [38] A. Bhola and P. Kumar, "Deep feature-support vector machine based hybrid model for multi-crop leaf disease identification in corn, rice, and wheat," *Multimedia Tools and Applications*, vol. 84, no. 8, pp. 4751–4771, 2025, ISSN: 1573-7721.
- [39] Z. Ji, S. Bao, M. Chen, and L. Wei, "Ics-resnet: A lightweight network for maize leaf disease classification," *Agronomy*, vol. 14, no. 7, p. 1587, 2024, ISSN: 2073-4395.
- [40] M. Masood et al., "Maizenet: A deep learning approach for effective recognition of maize plant leaf diseases," *IEEE Access*, vol. 11, pp. 52 862–52 876, 2023, ISSN: 2169-3536.
- [41] J. Hu, X. Jiang, J. Gao, and X. Yu, "Lfmnet: A lightweight model for identifying leaf diseases of maize with high similarity," *Frontiers in Plant Science*, vol. 15, p. 1 368 697, 2024, ISSN: 1664-462X.
- [42] C. K. Rai and R. Pahuja, "Northern maize leaf blight disease detection and segmentation using deep convolution neural networks," *Multimedia Tools and Applications*, vol. 83, no. 7, pp. 19 415–19 432, 2024, ISSN: 1573-7721.
- [43] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical systems and signal processing*, vol. 151, p. 107 398, 2021, ISSN: 0888-3270.
- [44] A. M. Hayajneh, M. Hafeez, S. A. R. Zaidi, and D. McLernon, "Tinyml empowered transfer learning on the edge," *IEEE Open Journal of the Communications Society*, vol. 5, pp. 1656–1672, 2024, ISSN: 2644-125X.

- [45] H. I. Peyal et al., "Plant disease classifier: Detection of dual-crop diseases using lightweight 2d cnn architecture," *Ieee Access*, vol. 11, pp. 110 627–110 643, 2023.
- [46] S. Vijayan and C. L. Chowdhary, "Hybrid feature optimized cnn for rice crop disease prediction," *Scientific Reports*, vol. 15, no. 1, p. 7904, 2025.
- [47] Y. Borhani, J. Khoramdel, and E. Najafi, "A deep learning based approach for automated plant disease classification using vision transformer," *Scientific Reports*, vol. 12, no. 1, p. 11 554, 2022.
- [48] U. Barman et al., "Vit-smartagri: Vision transformer and smartphone-based plant disease detection for smart agriculture," *Agronomy*, vol. 14, no. 2, p. 327, 2024.
- [49] S. Perez, N. Dilshad, N. S. Alghamdi, T. M. Alanazi, and J. W. Lee, "Visual intelligence in precision agriculture: Exploring plant disease detection via efficient vision transformers," *Sensors*, vol. 23, no. 15, p. 6949, 2023.
- [50] S. Mehta and M. Rastegari, "Mobilevit: Light-weight, general-purpose, and mobile-friendly vision transformer," *arXiv preprint arXiv:2110.02178*, 2021.
- [51] F. He, R. Wu, X. Zeng, H. Song, G. Li, and Z. Wei, "Skin lesion classification network based on improved mobilevit," *Engineering Applications of Artificial Intelligence*, vol. 159, p. 111 726, 2025.
- [52] M. Zhang et al., "Dual-attention-enhanced mobilevit network: A lightweight model for rice disease identification in field-captured images," *Agriculture*, vol. 15, no. 6, p. 571, 2025.
- [53] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.
- [54] M. I. Hossen et al., "Transfer learning in agriculture: A review," *Artificial Intelligence Review*, vol. 58, p. 97, 2025.
- [55] M. M. Mijwil, R. Doshi, K. K. Hiran, O. J. Unogwu, and I. Bala, "Mobilenetv1-based deep learning model for accurate brain tumor

- classification," *Mesopotamian Journal of Computer Science*, vol. 2023, pp. 29–38, 2023.
- [56] Z. Wang et al., "Research on crop diseases classification model based on mobilenet," *International Journal of Wireless and Mobile Computing*, vol. 24, no. 2, pp. 169–180, 2023.
- [57] K. Kadam, S. Ahirrao, K. Kotecha, and S. Sahu, "Detection and localization of multiple image splicing using mobilenet v1," *IEEE Access*, vol. 9, pp. 162 499–162 519, 2021.
- [58] R. J. Desai, S. V. Wang, M. Vaduganathan, T. Evers, and S. Schneeweiss, "Comparison of machine learning methods with traditional models for use of administrative claims with electronic medical records to predict heart failure outcomes," *JAMA network open*, vol. 3, no. 1, e1918962, 2020.
- [59] A. T. Khan, S. M. Jensen, A. R. Khan, and S. Li, "Plant disease detection model for edge computing devices," *Frontiers in Plant Science*, vol. 14, p. 1 308 528, 2023, ISSN: 1664-462X.
- [60] P. P. Ray, "A review on tinymml: State-of-the-art and prospects," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 4, pp. 1595–1623, 2022, ISSN: 1319-1578.
- [61] G. Delnevo, S. Mirri, C. Prandi, and P. Manzoni, "An evaluation methodology to determine the actual limitations of a tinymml-based solution," *Internet of Things*, vol. 22, p. 100 729, 2023, ISSN: 2542-6605.
- [62] A. M. Hayajneh, S. A. Aldalhmeh, F. Alasali, H. Al-Obiedollah, S. A. Zaidi, and D. McLernon, "Tiny machine learning on the edge: A framework for transfer learning empowered unmanned aerial vehicle assisted smart farming," *IET Smart Cities*, vol. 6, no. 1, pp. 10–26, 2024, ISSN: 2631-7680.
- [63] T. B. Shahi, C.-Y. Xu, A. Neupane, and W. Guo, "Recent advances in crop disease detection using uav and deep learning techniques," *Remote Sensing*, vol. 15, no. 9, p. 2450, 2023, ISSN: 2072-4292.
- [64] S. Adebayo, H. O. Aworinde, O. O. Olufemi, C. O. Osueke, A. E. Adeniyi, and O. J. Aroba, "Understanding mushroom farm envi-

- ronment using tinyml-based monitoring devices," *Environmental Research Communications*, vol. 7, no. 4, p. 045 014, 2025, ISSN: 2515-7620.
- [65] D. Sheth, B. Sudharsan, J. G. Breslin, and M. I. Ali, "Embedded ml pipeline for precision agriculture," in *2022 21st ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, IEEE, pp. 527–528, ISBN: 166549624X.
- [66] L. Falaschetti et al., "A low-cost, low-power and real-time image detector for grape leaf esca disease based on a compressed cnn," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*, vol. 11, no. 3, pp. 468–481, 2021, ISSN: 2156-3357.
- [67] F. De Vita, G. Nocera, D. Bruneo, V. Tomaselli, D. Giacalone, and S. K. Das, "Quantitative analysis of deep leaf: A plant disease detector on the smart edge," in *2020 IEEE International Conference on Smart Computing (SMARTCOMP)*, IEEE, pp. 49–56, ISBN: 1728169976.
- [68] V. Gutti and R. Karthi, "Real time classification of fruits and vegetables deployed on low power embedded devices using tiny ml," in *International Conference on Image Processing and Capsule Networks*, Springer, pp. 347–359.
- [69] P. Du, T. Polonelli, M. Magno, and Z. Cheng, "Towards lightweight deep neural network for smart agriculture on embedded systems," in *2022 IEEE Sensors Applications Symposium (SAS)*, IEEE, pp. 1–6, ISBN: 1665409819.
- [70] R. Nalwanga, J. Nsenga, G. Rushingabigwi, and I. Gatere, "Design of an embedded machine learning based system for an environmental-friendly crop prediction using a sustainable soil fertility management," in *2021 IEEE 19th Student Conference on Research and Development (SCOReD)*, IEEE, pp. 251–256, ISBN: 1665401931.
- [71] C. Nicolas, B. Naila, and R.-C. Amar, "Tinyml smart sensor for energy saving in internet of things precision agriculture platform," in *2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN)*, IEEE, pp. 256–259, ISBN: 1665485507.

- [72] P.-E. Novac, G. Boukli Hacene, A. Pegatoquet, B. Miramond, and V. Gripon, "Quantization and deployment of deep neural networks on microcontrollers," *Sensors*, vol. 21, no. 9, p. 2984, 2021.
- [73] R. Immonen and T. Hämäläinen, "Tiny machine learning for resource-constrained microcontrollers," *Journal of Sensors*, vol. 2022, no. 1, p. 7437023, 2022.
- [74] A. Mellit, N. Blasuttigh, S. Pastore, M. Zennaro, and A. M. Pavan, "Tinyml for fault diagnosis of photovoltaic modules using edge impulse platform and ir thermography images," *IEEE Transactions on Industry Applications*, 2025.
- [75] M. El Barkani, N. Benamar, H. Talei, and M. Bagaa, "Gas leakage detection using tiny machine learning," *Electronics*, vol. 13, no. 23, p. 4768, 2024, ISSN: 2079-9292.
- [76] N. Mduma et al., *The nelson mandela african institution of science and technology maize dataset*, version 6, 2022.
- [77] G. Geetharamani and A. Pandian, "Identification of plant leaf diseases using a nine-layer deep convolutional neural network," *Computers & Electrical Engineering*, vol. 76, pp. 323–338, 2019.
- [78] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, and N. Batra, "Plantdoc: A dataset for visual plant disease detection," in *Proceedings of the 7th ACM IKDD CoDS and 25th COMAD*, 2020, pp. 249–253.
- [79] B. R. Reddy, G. Kalnoor, M. Devashish, and P. S. K. Reddy, "Deep learning based mobile application for automated plant disease detection," *IEEE Access*, 2025, ISSN: 2169-3536.
- [80] A. M. Hayajneh, S. Batayneh, E. Alzoubi, and M. Alwedyan, "Tinyml olive fruit variety classification by means of convolutional neural networks on iot edge devices," *AgriEngineering*, vol. 5, no. 4, pp. 2266–2283, 2023, ISSN: 2624-7402.
- [81] M. Raiaan et al., *A systematic review of hyperparameter optimization techniques in convolutional neural networks. decis. anal. j.* 11, 100470 (2024), 2024.
- [82] O. B. Eldine, M. Selim, N. El-Makky, and N. M. Ghanem, "Tinyml prototype of infant cry classification system," *IEEE Access*, 2025.