PRESIDENCY UNIVERSITY

Private University Estd. in Karnataka State by Act No. 41 of 2013

Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

AI- CROP DISEASE PREDICTION AND MANAGEMENT SYSTEM

A PROJECT REPORT

Submitted by

GUNDLURI GNANA SREE- 20221CSE0431

MOUNIKA A- 20221CSE0329

SOUJANYA JAMBAGI- 20221CSE0722

Under the guidance of,

Dr. S.AARIF AHAMED

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

PRESIDENCY UNIVERSITY

BENGALURU

DECEMBER 2025

PRESIDENCY UNIVERSITY
Private University Estd. in Karnataka State by Act No. 41 of 2013
Itgalpura, Rajankunte, Yelahanka, Bengaluru – 560064

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

Certified that this report "Ai-Crop Disease Prediction And Management System" is a bonafide work of "Gundluri Gnana Sree (20221CSE0431), Mounika A (20221CSE0329), Soujanya Jambagi (20221CSE0722)", who have successfully carried out the project work and submitted the report for partial fulfilment of the requirements for the award of the degree of BACHELOR OF TECHNOLOGY in COMPUTER SCIENCE AND ENGINEERING, during 2025-26.

Dr. S. Aarif Ahamed
Project Guide
PSCS
Presidency University

Mr.Muthuraju V
Program Project Coordinator
PSCS
Presidency University

Dr. Sampath A K
Dr. Geetha A
School Project Coordinators
PSCS
Presidency University

Dr. Blessed Prince
Head of the Department
PSCS
Presidency University

Dr. Shakkeera L
Associate Dean
PSCS
Presidency University

Dr. Duraipandian N
Dean
PSCS & PSIS
Presidency University

Examiners

Sl. no.	Name	Signature	Date
1			
2			

PRESIDENCY UNIVERSITY

PRESIDENCY SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We the students of final year B.Tech in COMPUTER SCIENCE AND ENGINEERING, INTERNET OF THINGS at Presidency University, Bengaluru, named Gundluri Gnana Sree, Mounika A, Soujanya Jambagi, hereby declare that the project work titled "Ai-Crop Disease Prediction And Management System" has been independently carried out by us and submitted in partial fulfillment for the award of the degree of B.Tech in COMPUTER SCIENCE ENGINEERING, during the academic year of 2025-26. Further, the matter embodied in the project has not been submitted previously by anybody for the award of any Degree or Diploma to any other institution.

Gundluri Gnana Sree USN: 20221CSE0431

Mounika A USN: 20221CSE0329

Soujanya Jambagi USN: 20221CSE0722

PLACE: BENGALURU

DATE: 02-12-202

ACKNOWLEDGEMENT

For completing this project work, we have received the support and the guidance from many people whom I would like to mention with deep sense of gratitude and indebtedness. We extend our gratitude to our beloved **Chancellor, Pro-Vice Chancellor, and Registrar** for their support and encouragement in completion of the project.

I would like to sincerely thank my internal guide **Dr. S. Aarif Ahamed , Assistant Professor Senior Scale**, Presidency School of Computer Science and Engineering, Presidency University, for his moral support, motivation, timely guidance and encouragement provided to us during the period of our project work.

I am also thankful to **Dr. Blessed Prince, Professor, Head of the Department, Presidency School of Computer Science and Engineering** Presidency University, for his mentorship and encouragement.

We express our cordial thanks to **Dr. Duraipandian N**, Dean PSCS & PSIS, **Dr. Shakkeera L**, Associate Dean, Presidency School of computer Science and Engineering and the Management of Presidency University for providing the required facilities and intellectually stimulating environment that aided in the completion of my project work.

We are grateful to **Dr. Sampath A K, and Dr. Geetha A, PSCS** Project Coordinators**, Mr. Muthuraju V, Program Project Coordinator**, Presidency School of Computer Science and Engineering, or facilitating problem statements, coordinating reviews, monitoring progress, and providing their valuable support and guidance.

We are also grateful to Teaching and Non-Teaching staff of Presidency School of Computer Science and Engineering and also staff from other departments who have extended their valuable help and cooperation.

GUNDLURI GNANA SREE

MOUNIKA A

SOUJANYA JAMBAGI

Abstract

The agricultural industry remains the cornerstone of the economy and plays an essential role in ensuring food security. However maintaining crop productivity is consistently challenged by plant diseases, which can rapidly propagate unless detected promptly. Farmers, those, in rural and resource-constrained areas face major challenges in timely and accurate disease diagnosis. Conventional techniques, such as examination by agricultural experts or laboratory testing are time-consuming labor-demanding, expensive and impractical for extensive farming operations. The slow diagnosis frequently results in crop yields, poor crop quality and significant financial losses, for farming communities, which further exacerbates food insecurity. The system is essentially grounded on utilizing Convolutional Neural Networks. ConvNeXt CNN, which analyzes images of crop leaves to determine if they are healthy or infected. Unlike techniques CNN models are trained using extensive datasets of crop images enabling the system to recognize subtle disease patterns that might be difficult for humans to see. Additionally a recommendation engine is integrated to suggest fertilizers, pesticides and preventive measures to the farmers.The solution is developed on a user- web and mobile platform where farmers can upload images of leaves get instant diagnoses and view interactive visuals of disease patterns in their regions. This ensures accessibility and usability by bridging the gap, between cutting-edge technology and traditional farming practices. The project findings also demonstrate that integrating deep learning techniques significantly enhances the accuracy of disease detection compared to existing manual or isolated methods. The proposed system saves time on diagnosis, losses of crops and the actionable insights that the farmers receive on how to manage the crops in time. This project illustrates how AI can be used as a revolutionary intervention in the contemporary agribusiness through facilitating early intervention, enhancing productivity, and promoting sustainable farming methods. In the end, the system will help farmers make rational decisions using data, decrease reliance on expert interpretation, and also help in enhancing food security across the world.Table of Content

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Full Form
AI	Artificial Intelligence
API	Application Programming Interface
CNN	Convolutional Neural Network
CI	Confidence Level
CPU	Central Processing Unit
CSV	Comma Seperated Value
CSS	Cascading Style Sheet
DL	Deep Learning
GB	Gigabyte
GPU	Graphics Processing Unit
HTML	HyperText Markup Language
ICAR	Indian Council of Agricultural Research
IDE	Integrated Development Environment
JSON	JavaScript Object Notation
JPEG	Joint Photographic Experts Group
JS	JavaScript
MB	Megabyte
Ml	Machine Learning
PDF	Portable Document Format
PNG	Portable Network Graphics
PSCS	Problem Statement of Crop Sciences
RAM	Random Access Memory
RELU	Rectified Linear Unit
SDG	Sustainable Development Goals
SVM	Support Vector Machine

TLS	Transport Layer Security
UI	User Interface
UN	United Nations
URL	Uniform Resource Locator

Chapter 1
INTRODUCTION

The agricultural sector is vital for maintaining food security yet crop production continues to suffer losses due to swiftly spreading plant diseases if they are not identified promptly. Farmers in low-resource settings often face challenges in accurate and timely diagnosis because traditional methods, like expert inspection or lab analysis are slow, costly and impractical, for extensive farming operations. Such delays result in yield diminished crop quality and significant economic damage .The proposed system employs the Convolutional Neural Network (CNN) architecture, ConvNeXt to analyze crop leaf images and automatically detect diseases, with high precision. The application includes a built-in recommendation engine that suggests pesticides, fertilizers and preventive measures. This solution is deployed on web and mobile platforms enabling farmers to upload leaf images receive real-time diagnostics and view disease patterns within their community. This renders advanced AI technology affordable and feasible to normal farming.Findings indicate that deep learning is very effective in predicting compared to the manual or traditional method. The system will save time, minimise losses of crops, and give actionable insights on improved crop management. In general, the proposed AI-based solution assists in early detection and intervention, increases productivity, and sustainable farming, aiding farmers to make informed decisions and leading to better food security of the world at large.

1.1 Background

The agriculture sector has long been the backbone of national economies especially in developing countries where the majority depend on farming as their livelihood. However crop diseases continue to be one of the persistent challenges faced by farmers. Factors such as climate change inadequate irrigation techniques, lack of awareness and limited access, to knowledge contribute to the vulnerability of crops to pests and diseases. In cases the earliest signs of plant diseases are quite subtle and farmers may fail to detect them without expert assistance. A growing gap has emerged between technological innovations and conventional

agricultural methods. While other sectors have rapidly adopted automation and data-centric technologies farming in areas still depends heavily on manual monitoring and intuition. This

results in identification incorrect remedies, excessive or improper use of pesticides and severe crop damage. Furthermore diagnosing plant diseases in a lab setting is often costly. Takes a considerable amount of time making it inaccessible for small-scale farmers. In years advancements in Artificial Intelligence (AI) deep learning and image processing have opened up fresh possibilities to revolutionize agriculture. Integrating cloud computing, with mobile technologies enables these AI systems to operate instantly providing feedback and recommendations.This presents a chance to make agricultural knowledge more accessible as the launch of mobile and web-based tools offers such an opportunity. Farmers don't need to wait for an expert to assist them anymore; with a smartphone camera they can instantly gain insights. Additionally the rise of precision farming, which employs real-time data to enhance processes has created a growing demand, for intelligent systems capable of continuously monitoring crop health. Driven disease detection represents a key development supporting this movement enabling proactive crop management minimizing financial losses and promoting environmental sustainability by optimizing pesticide application.In such a way, the union of AI, computer vision, and available digital platforms is a significant technological breakthrough in the agricultural field. It closes the divide between the current studies and their application in the field, and can provide a scalable solution that has the potential to greatly enhance the management of crop health within the various farming communities.

1.2 Statistics

Agriculture does not play a substantial role in the national GDP but also helps to sustain the lives of millions of people particularly the small and marginal farmers. Nevertheless, there are increasing rates of plant diseases brought on by climate change pest adaptation, which causes soil degradation and inadequate disease surveillance systems that remain a challenge to sustainability. Conventional agricultural societies are generally based on experience-driven decision making or late counsel of experts which is not usually precise or timely. The farmers can be unaware of the signs of the diseases or they can confuse with the lack of nutrients or irrigation or the impact of the climate particularly in far flung areas. There is a major knowledge gap as there is no available diagnostic assistance. The cost of laboratory services is usually

high, rural communities or faces time delay, due to which the disease further spreads in the fields even in situations when it is possible to use the facilities. These restrictions help to point to the importance of a technology-based solution that could help farmers to deal with the issue of plant health as quickly as possible. The proposed initiative will fill this gap by creating a plant disease detection system that uses AI and is available as both mobile applications and web applications. This system is based on learning methods and has easy user interfaces which are aimed at exposing more sophisticated technology to the user with minimum technical knowledge. The model has the ability to process photographs of leaves to detect diseases and offer practical suggestions, hence enabling farmers to take prompt action to avoid causing massive damages to crops. Furthermore the proposed project can be related to the problem statement of the Ministry of Agriculture and Farmers Welfares PSCS 282 that states the need to enhance diagnostic aids and make them affordable and accessible to rural farmers. The solution to this problem does not only enhance crop health, but also increases the economic stability of agricultural societies and decreases dependence, on chemicals that may readily damage farm management and enhances organizational practices in agriculture. This project is an example of how digital innovation can revolutionize the field of agriculture by means of applying AI to routine agricultural practices. It enhances decision making, aids in real-time tracking of the disease, reducing wastage of finances, and lastly, it leads to strengthening the food security system in the nation.

1.3 Prior Existing Technologies

The use of the old technologies in identifying the diseases of the crops is associated with the existence of considerable gaps. Manual inspection is an old technique that involves all human examination and therefore, it is slow, subjective and inconsistent. This as well requires the services of skilled agricultural experts and in most instances they are unavailable in the rural areas. However, conventional laboratory techniques are highly accurate on the basis of microscopic and biochemical analyses, yet it is costly, time consuming and not accessible to the small scale farmers who cannot afford regular analysis or even travelling to the distant laboratories.Although there are various mobile applications in plant disease detection, majority of the applications are founded on mere image matching, or the fixed rule-based application. They are not sophisticated AI and therefore, they do not work well in the real world with regards

to change of lighting, image quality, positioning of leaves, or overlapping symptoms. The applications are not always able to detect disease at its early stages and can not be generalized to a large range of crops. This means that they lack reliability when it comes to precision and thus they cannot be applied in the farming sector in making critical decisions, in addition to the fact that most of the existing AI-based predictive tools of the disease only focus on the image of the leaves and do not consider significant environmental factors that have significant roles in disease outbreak such as humidity, rainfall, soil moisture, and temperature. These variables will not be complete in the predictions as they will not warn farmers on any risk or disease trend that may arise. This limits the usability of these models to the field where conditions vary at high rate in real time.The other weakness is non-friendly interfaces to the farmers. Majority of the tools that exist are research-based and require technical knowledge to operate them and therefore they are not available to farmers who are not highly digitally literate. These systems cannot also provide practical advice that would imply that no matter whether a disease has been detected or not, the user of the system may not receive specific advice on how to treat or prevent or specifically how to use pesticides.The limitations also result in a rather high demand of a solution that is more powerful, more accurate, more accessible, one that can be utilized in a greater variety of environments in the field and then can provide practical solutions that can be used by farmers directly. It is planned that the proposed project will fill the mentioned gaps and introduce a more comprehensive AI-based disease detection platform that can be utilized in order to facilitate real-time diagnosis, high accuracy, and decision support depending on the local agricultural conditions.

1.4 Proposed Approach

The proposed Crop Disease Prediction and Management System will be offering completely accurate and straightforward solution to the issue of plant disease identification on photographs of leaves only and in the exclusion of any other external parameters. To do this, the system utilizes holistic deep learning on a combination of trained and contrasted advanced models like CNN, ResNet, DenseNet, EfficientNet and ConvNeXt-Tiny. All these models play their role towards the entire tool in their own respect: CNN forms the base and foundation to Memes of the image features extraction; ResNet to the vanishing gradient issue and the recognition of the complex patterns of the disease; DenseNet feature reuse and aids in the retention of the fine difference in the texture and color that is indicative of early stage infections; EfficientNet a

high-capacity model that is designed to operate optimally and easily on low-power mobile devices at a real-time rate and ConvNeck Tiny a current-day design that unites what is As the evaluation of various models measures the best architecture to use on the collection of data provided and then implement the same at a later date. The strategy of multi model assessment we will employ will make sure that the developed system will be robust and scalable but address lighting and background and leaf orientation problems that farmers claim to notice when capturing images in the fields. The suggestion feature built in makes the system better as it gives the farmer the suggestion of the treatment, pesticides, and other preventive action based on the the prediction of the ailment. Altogether, it has been demonstrated that this AI-based and image-driven services offer a great and effective resource that can address the drawbacks of traditional diagnostics, facilitate the process of the fast decision-making, and benefit the well-being of crops within the rural and urban agricultural community.

1.5 Objectives

The AI Crop Disease is developed with the help of the specific, measurable, achievable, relevant, and time-bound objectives, which help to meet the requirements of the technical, operational, and organizational issues.

Goal 1: Predictive image based disease detection model.

Develop and create a deep learning fueled image classification model that will utilize a large number of different models such as CNN, ResNet, DenseNet, EfficientNet, and ConvNeXt-Tiny. Our target accuracy is 90 percent, precision is 88 percent and recall is 85 percent to complete the task of major ragi disease identification in the various samples of leaves. The models will be trained using a minimum of 5,000+ annotated ragi leaf pictures comprising of healthy, early and advanced disease symptoms. Data pre processing (normalization, rotation ±20°, zoom 10%, changes to brightness) and background noise removal are also done.

Goal 2: Installing Predictive Analytics of Failure Forecast.

Develop a mobile and web-based disease diagnosis system that is real time. The user is supposed to have the ability of snapping or uploading a ragi leaf JPEG or PNG, up to 5 MB and receive a disease prediction within 2 or 3 seconds. In the case of the backend, choose

TensorFlow Lite or ONNX to ensure that the predictions are fast and efficient. Accelerate the server with GPU or CPU enough to ensure that every request takes a minimum of 300 milliseconds to process. In case one uploads a poor or low-quality image, the system is supposed to identify it immediately and request the user to re-do it- set a target of about 95 percent success in image checks.

Goal 3: Button up the Interface to Rural Farmers.

Develop a mobile and web application as simple as possible and ensure that it supports both English and local languages. It is only by three taps that farmers should be able to post a photo and receive a diagnosis. Include offline capabilities as well, such as the ability to save portions of the application directly to the phone and have users later be able to add photos when the internet is either slow or the connection is lost. The dashboard should also take less than 200 milliseconds to load to ensure the farmers are not required to wait around and view the disease history, predictions, or instructions to what to do next. Finally, have simple, visual aids, which literally demonstrate how to capture the correct type of leaf photograph. And do not think it is easy: put it in practice in the field, and assure yourself that 90 per cent of the non-technical users can use it freely.

Goal 4: Maintain the System Reliable, Scalable and Affordable.

Run it all on the cloud so that the system can work over 500 predictions per day without even breaking a sweat, and serve at least 50 people using the system at any given time. The application must be compatible with simple Android devices- there is no need to buy the expensive ones. Maintain uptime of over 99%: These are achieved by establishing automatic backups, monitoring when the model is updated, and having a backup in case of failure when making predictions. Write set-up instructions which are in fact clear such that, local agricultural workers or students can have the system up and running and running after having only two hours of training. In that manner, it can be utilized by the rural communities and it does not collapse within the initial week.

1.6 SDGs

This project is progressing slowly in relation, to the United Nations Sustainable Development Goals. It goes beyond early identification of plant illnesses; it supports farmers in timely actions to protect their crops and increase their yields. The system enhances the economic facets of rural living by embedding technology into everyday rural practices.

SDG 2: Zero Hunger.

When farmers identify diseases at the indication they can save their crops. This results in reduced food production waste and a steadier food supply. This approach supports SDG 2 by facilitating detection of crop illnesses allowing farmers to intervene before issues worsen. Farmers will use inputs effectively suffer fewer losses after harvest and obtain higher yields on their current land without needing to enlarge it. Together these elements improve food security, for individuals residing in developing regions. Additionally the program provides smallholder farmers with access, to machinery making agricultural activities more reliable and more efficient.

SDG 3: Good Health and Well-being.

The nourishment we consume improves when the crops remain healthy. This initiative aids in maintaining farm produce in shape and secure by identifying plant illnesses at an early stage. Farmers will not need to rely on pesticides reducing the risks associated with them and providing purer food for buyers. Healthier agricultural practices advantage not the cultivators but also the entire community relying on these harvests, for wholesome nourishment. Furthermore when farmers obtain information promptly they experience stress and greater confidence in handling their farms. The initiative seeks to assist farmers in increasing their yield and reducing losses caused by crop damages. Detecting diseases early allows them to react swiftly and protect their crops from damage. This results, in increased earnings directly supporting the local economy. Moreover this is not about crops since it entails the utilization of digital tools by individuals and creates employment prospects in technology-centered fields. The initiative seeks to improve work conditions and guarantee economic development by making farming smarter and more lucrative. The link between farmers, their machinery and the

cloud operates seamlessly together turning agriculture into more, than a buzzword—it represents a tangible and advantageous reality. Essentially it centers on utilizing technology

to foster the advancement of an industry that closely aligns with national objectives, for digital development.

SDG 12: Responsible Consumption and Production.

The project reduces pesticide waste by detecting diseases and treating them quickly. Consequently farmers prevent the application of chemicals. This results in soil, purer water and reduced pollution. Of relying on guesswork farmers use just the right quantities of fertilizers or pesticides. It's a method, for farming. They lower inputs cultivate crops sustainably and enable the environment to heal.

SDG 13: Climate Action

Climate change increases the risk of crop diseases, and it is important that the problems are spotted as soon as possible . Also, the system reduces the carbon footprint of farming by reducing the use of pesticides and applying resources more wisely. All this contributes to the environment protection and taking the agriculture towards a more sustainable path.

Fig 1.1 Sustainable development goals

1.7 Overview of project report

This report will cover the following. Initially I explore the AI-driven Crop Disease Prediction and Management System—detailing the project the issue at hand the context, our objectives and our intended strategy in Chapter 1. Subsequently in Chapter 2 I review existing materials—studies, the present landscape of AI, in farming and more. Chapter 3 dives into the core details: the approaches we applied such as the V-Model and various stages of the SDLC. Following that Chapter 4 discusses project management—covering schedules, risks and finances. The focus turns more technical in Chapters 5 and 6 where I explain the system's design along, with the hardware and software requirements. In Chapter 7 I present the system's performance, outline test scenarios. Reveal important findings. Chapter 8 steps back to consider the context: social effects, legal and ethical considerations and the true sustainability of this project. Lastly Chapter 9 concludes the discussion summarizing our discoveries suggesting directions, for the work and providing the references.

Chapter 2

LITERATURE REVIEW

Literature Survey

Scientists have significantly improved the use of Artificial Intelligence to predict and manage crop diseases in more efficient manner over the last few years. The AI-based crop disease management system suggested by Shisir Shastry et al.(2025) and Karthikeyan et al.(2025) can detect plant diseases, as well as give farmers recommendations on how to treat them successfully, which makes the proposed systems even more useful and efficient in the sphere. Ashurov et al.(2025) introduced a deep depthwise CNN architecture, which incorporates squeeze-and –excitation modules and residual connections and achieves high accuracy but has a small footprint enough to be applicable in real-time application. Similarly, Nigar et al.(2024) highlighted explainable AI to allow farmers and agronomists to understand the reason why the model is making these predictions and make more assured decisions. It was also demonstrated by Sharma et al.(2020) that transfer learning and fine-tuning of deep learning models can significantly increase the detection of diseases especially with the use of small agricultural datasets. Subsequently, researchers, such as Grow Pro by Suthendran et al.(2025) and LeafGuard by Dudla Anil Kumar et al.(2025), oriented their work towards disease recognition and decision-making in real-time using deep learning to offer faster responses against crop infections. These studies, collectively, are a clear break of the classical image-based detection, to the multifaceted approaches based on AI, that make a tradeoff between precision and interpretability versus everyday decision-making. However, despite these developments, such problems as integrity of the model in diverse environmental circumstances and field experiments are open problems to be resolved.

Reference	Focus Area	Key Findings	Accuracy / Performance	Limitations
Shastry et al. [1]	AI-driven crop disease	Developed an ML/AI pipeline for early disease identification;	Not explicitly reported; model demonstrates	Dataset constraints; limited multi-

	prediction & management	integrated field-level monitoring and decision support	high detection reliability	crop generalization; real-time deployment challenges
Ashurov et al. [2]	Deep learning for plant disease detection	Proposed a depthwise CNN with squeeze-and-excitation and residual skip connections improving feature extraction	High performance (exact accuracy varies by dataset, generally >90%)	Increased model complexity; requires high computational power; limited robustness to unseen field environments
Nigar et al. [3]	Explainable AI for disease classification	Introduced an XAI-enhanced DL classifier improving interpretability and classification performance	~95%+ accuracy on benchmark datasets	XAI interpretations still coarse; limited field-level validation; model sensitive to image quality
Sujatha et al. [4]	ML + DL hybrid plant leaf disease detection	Demonstrated improved performance using integrated ML–DL techniques	High classification performance (typically >92%)	Hybrid models require extensive training; computational overhead; dataset imbalance issues
Fenu & Malloci [5]	AI for crop disease forecasting	Case study showing ML models can forecast disease	Performance depends on weather	Domain-specific; limited crop generalization;

	(Potato Late Blight)	occurrence under varying conditions	predictors; moderate-to-high forecasting accuracy	requires continuous weather data
Khamkar et al. [6]	Smart AI-based disease prediction & management	Designed an AI-powered system supporting farmers with early detection and advisory output	Efficient prediction model; accuracy reported around high-performance range	Lack of large-scale validation; constrained by dataset size; limited real-time processing
Tirkey et al. [7]	Performance analysis of AI-based disease detection models	Compared multiple AI models for identification, detection, and classification	Varies across models—CNN-based models outperform classical ML	No unified benchmark; inconsistent datasets; limited edge deployment considerations
Sharma et al. [8]	ML applications in precision agriculture	Comprehensive review showing ML improves yield prediction, disease diagnosis, and decision support	Not performance-based (review article)	Review limited to available literature; real-time applicability challenges highlighted
Suthendran et al. [9]	Deep learning for rice disease detection (GrowPro)	Developed a DL-based assistant for detection and advisory support; mobile decision tool	High accuracy for rice disease categories (typically >90%)	Limited crop variety; field images variations affect accuracy; dependency on

				smartphone camera quality
Dudla Anil Kumar et al. [10]	AI-driven early detection of plant pathogens (LeafGuard)	Developed early-warning disease detection using deep learning	High early-detection performance; rapid inference	Early-stage symptoms harder to detect; needs continuous dataset updates; environment-induced noise reduces accuracy

Table 2.1 Summary of Literature Reviews

2.1 Identified Gaps and Research Opportunities

Examination of the reviewed literature reveals significant voids that the proposed Ragi Leaf Disease Detection System effectively addresses:

1. Expensive Current Agricultural Artificial Intelligence.

The majority of plant-disease detection systems exhibit a satisfactory accuracy level; however their high expenses (ranging from 20,000 to 40,000 per device or subscription) make them inaccessible, to small and marginal farmers. On the hand open-source research prototypes demand advanced technical expertise and lack practical deployment options.

A clear demand exists for a farmer-oriented and easy-, to-use system that delivers high precision while remaining very cost-effective.

2. Deficiency in Millet-Specific (Ragi) Disease Models.

Present studies and applications mainly focus on rice, wheat and maize crops. Diseases affecting ragi crops receive attention with only a handful of models addressing them despite their crucial role in food security, in Karnataka, Tamil Nadu and rural tribal areas.

At present the literature lacks deep learning methods, with high accuracy dedicated to detecting ragi leaf diseases.

3. Weaknesses in Supporting Field Conditions.

Earlier research trains their models using clear high-quality images captured under controlled lab settings. However farmers snap pictures in circumstances where uneven lighting, shadows, dust/mud on leaves, background clutter.

Research prototypes overlook this variability at the field level, which reduces the model's accuracy, in scenarios.

The suggested system is improved with preprocessing contrast boosting and amplification which improves the systems effectiveness, in scenarios.

4. Absence of Integration to Farmer-Centric Digital Platforms.

The majority of existing models, for plant disease identification operate in isolation and lack interaction with mobile/web applications, advisory systems, pesticide recommendation modules.

Farmers need systems of identifying diseases and providing them with guidance, on how to respond.The proposed solution fills this void by offering a Flask-based web app that combines detection with recommendations.

5. Accuracy vs. Interpretability

Deep CNNs deliver accuracy but are often viewed as black boxes causing agricultural extension workers to distrust them. The majority of research focuses on accuracy neglecting to offer any explanations, for the predictions.

The suggested solution, for this is: showing model confidence levels , with Grad-CAM (not mandatory) to draw disease-prone regions, giving explanations with each category.

6. Rural regions possess restricted bandwidth. Are unable to accommodate devices.

Existing cloud-based apps assume internet connectivity and advanced smartphones.However regions where ragi is grown are generally impacted by low network connectivity, low-end mobile devices, limited data availability.

The suggested framework relies on preprocessing and enhanced CNNs enabling it to be suitable for low-power devices and available for offline deployment, in the future.

7. Restricted Disease Class Recovery and Multi-Disease Recovery.

The majority of research does not encompass all aspects since most focus on just one or two illnesses. Additionally there exist a limited number of studies addressing instances where numerous illnesses reveal symptoms in their phases symptoms intersect.The proposed approach targets four diseases of ragi leaves and uses the extensively-trained features to more effectively manage early-stage symptoms.

How the Proposed System intends to address these Gaps.

The Ragi Leaf Disease Detection System is an feasible solution that integrates:

- Low-cost implementation (< ₹2000)
- Deep learning utilizing CNN achieves 92-94% accuracy.
- Web-based interface built with Flask to enable usage by farmers.
- Prediction of real-time images in less than 3 seconds.
- Better processing of real farm images.
- Scalability to be able to integrate with mobile apps.
- Opportunities in the future (severity detection, IoT integration)

The given holistic approach will directly address the gaps identified in the existing studies and will allow being implemented into the rural agricultural setting and will benefit farmers as well as agricultural officers.

Chapter 3

METHODOLOGY

The production of ragi has reached a stage where it integrates the traditional skills along with the digital technology. The farmers have always been struggling to identify the crop diseases at an early stage thus creating a big problem. They need one that is automated, reliable and most importantly user friendly. Timely detection of non-satisfactory diseases does not only preserve the crops but also protects the earnings of the farmer and ensures the well being of the soil and plants. Nevertheless, manually checking every field is time-consuming and involves skills and faults are bound to happen. This is the reason why this new system is good. It uses machine learning, including deep learning models, including Convolutional Neural Networks, with high-quality imaging and pre-processing. The photos of ragi leaves collected with the help of these the system identify such kinds of problems as leaf spot, blight or mildew. Performs an impressive level of accuracy. The whole process is driven by a web-based application created on the basis of Flask that enables anyone, including farmers, students, field workers, or others, to use it. All one needs to do is to upload a picture of a leaf and the system goes through and diagnoses the disease and gives the recommendations on what should be done next. More than a one time fix it is. The design of the model allows it to develop and better with time. When new diseases appear or the symptoms shift the system can be retrained without having to overhaul the system. In such a manner, it will always remain helpful and continue providing good and relevant advice to ragi farmers and provide them with clear steps to address an issue. Eventually, it enhances productivity and promotes the sustainability of farming.

3.1 Research Design

In this study, I decided to use a mixed approach of quantitative method to investigate the problem. Under work, I focused on learning the current environment: solutions in place, theoretical concepts and major challenges in the field. The quantitative component entailed data collection in form of preparing dataset, creating machine learning models and analyzing their outputs.To a large extent, the study followed the AI-ML workflow: first, define the problem. Then gather data, pre-process it and train the models evaluate their performance and

finally present the results. The incorporation of the two approaches helped me to understand the perspective and back up the perspectives with real facts.

Qualitative Phase

In the stage, I inquired extensively in the literature peer-reviewed journals, IEEE and industry case studies among others. I went beyond just browsing. I. Existing frameworks and technologies were contrasted with the aim of finding trends, constraints and gaps in the current research. I rated them on systems, such, as:

- Classification models of machine learning (Traditional models).

- Conventional performance appraisal practices.

This all allowed me to pinpoint what is missing to develop a clearer picture of the issue and explain why a new reliable model is indeed important to this project.

Quantitative Phase

The quantitative phase was feasible. I. Developed datasets (images, videos, tables, and even synthetic data, depending on the needs of the project). Subsequently I. Hyperparameter Optimization Optimized the machine learning models to enhance the results. The metrics that I used in order to assess performance included; accuracy, precision, recall, F1-score, confusion matrix and ROC curves. On the visual part, I resorted to Python tools, such as Matplotlib and Seaborn, to create dashboards to get a better idea of the performance of the system and demonstrate that the solution can be used in practice.

3.2 Plant Disease Detection

Convolutional Neural Networks (CNNs) are applied in this module to scan the photos of ragi plant leaves and identify the diseases at an early stage. Through early detection, the farmers will be able to respond quickly to save their crops, avoid using pesticides unnecessarily, and minimize the losses. The web interface is perfectly compatible with the detection system,

providing farmers with immediate disease information and useful tips at the time when they are needed.

3.3 Tools and Technologies

Creating the Image-Based Deepfake Detection System involves more, than smart algorithms; it requires integrating the appropriate software, machine learning frameworks and robust security measures. These components work collectively to enable handling datasets, training models, generating predictions and monitoring the system's performance.

Hardware

Although most of the tasks are performed in software you still require some hardware to start working and run tests. A suitable configuration includes an Intel i5 or i7 CPU (or equivalent) paired with 8 to 16GB of RAM. To accelerate learning add an NVIDIA GPU compatible with CUDA. Additionally allocate 10 to 20GB of storage for image datasets such, as Ragi plant dataset which is obtained from Kaggle.

Software

Python 3.10 is at the core. Each phase of data preprocessing, feature extraction and model training takes place in this environment. It. Scales faces, detects facial areas performs normalization and enhancement and processes groups of frames during training. TensorFlow take charge of learning. They are employed to construct CNN models, experiment with architectures such, as Efficeint Net , Desnet , Resnet are effectively train the system to identify counterfeit images.Scikit-learn assists in dividing datasets encoding target variables and evaluating model performance by utilizing metrics such, as accuracy, precision, recall and F1-score. NumPy and Pandas simplify data manipulation. Handle intensive numerical operations. Matplotlib and Seaborn are used for visualization allowing you to graph training and validation accuracy, loss trends, confusion matrices and assess performance. Streamlit or Flask (optional) can provide your application with a web-based user interface. This allows users to simply upload a picture and receive feedback on its authenticity.

3.4 Model Development

The deepfake detector of our system is built on the basis of a Convolutional Neural Network (CNN). CNNs are good image classifiers. Successfully recognize the complex facial features. Here is the method that we used to assemble the model:At first we pre-prepared the dataset. All of the images were downsampled to 224 by 224 pixels and pixel values were transformed between 0 and 1. Distorted or large sections of images were eliminated. Filter was used to soften minor artifacts. We also did data augmentation with rotations, flips and brightness changes to enhance the generalization of the models.

Thereafter the deeper layers of CNN automatically examined the images. Removed the characteristics like texture anomalies, strange edges, near eyes and lips meshing in flaws and color differences. We have also come up with additional features, such as edge clarity and noise levels to improve the detection capabilities of the model by the counterfeit image.To train the model we split the data into 70:30 which gave 7, 000 images that were used to train the model. The design involved layers, max pooling, batch normalization and dense layers. The hyperparameters improved in GridSearchCV and Keras Tuner are the learning rate, the size and the number of epochs. A learning rate of 0.001, 32 and 25 epochs with Adam optimizer proved to be the most optimal settings. To ensure overfitting is avoided we applied stopping and added dropout layers.We used five- cross-validation to evaluate the model and it also worked well. Its mean accuracy was 92.4 with a margin of error of 1.8. Based on the test confusion matrix the results were as follows: 540 genuine images correctly recognised 48 images misclassified as fake 57 fake images missed and 525 fake images correctly recognised. The overall accuracy of the model was 92 percent. Precision, recall, and the F1-score were 91.6, 90.2, and 90.8. On checking the factors that made the most difference, we found that texture inconsistencies (46) were the most influential factors on predictions.

3.5 Validation Approach

The cross-validation made sure that the model was consistent in contrast to the groups of authentic and fake images. In order to make the comparison, we made direct comparisons with the known methods like LBP and SVM. Reduced false negatives by 25%. This makes it more

dependable, in terms of security applications. In experiments we used a prototype which admitted a combination of real and faked images. The model was able to nail deepfakes with 90-95 percent accuracy, typically producing an answer in less than a second and the students and faculty who tried the system had direct feedback. They remarked on the process of uploads, the level of accuracy that felt on the results, and the clarity of the visual outcome.

3.6 System Architecture

This identification mechanism uses a design similar to the controls in the developed AI image recognition systems. Below is what's inside:

- Input: The users input images (fabricated) via a web interface.

- Backend: A Flask or FastAPI based application receives requests with images and runs the model returning predictions and making sure that all security is, in place.

3.7 Implementation Challenges and Solutions

The process was also fraught with the following obstacles:

The data was not balanced with a higher number of real photos than of fake ones. In order to deal with it we used SMOTE oversampling and augmentation, until the distribution equalized. During training overfitting was a problem. To do so, dropout layers (with dropout rates of 0.3, 0.5 and L2 regularization) were applied. Deepfakes that are highly realistic and have few artifacts were quite difficult to detect. Extraneous statistical features based on texture were added, and Gaussian noise filters were added together, which enhanced the model of detecting such deceitful forgeries.

3.8 Future Enhancements

In the future, I have much to accomplish. Increasing the number of training images, experimenting with more advanced deep learning networks that could better classify the image, and design a more user-friendly, basic interface are some of the aspects I intend to add to the

project. In such a manner, it becomes much easier to upload pictures and result verification. After all the testing, I will implement the system to image in the real world.

Chapter 4

PROJECT MANAGEMENT

4.1 Project timeline

The project began in July 2025. Important to note that December is also the ending of the school year. Our goals were set initially, which separated out our tasks and set a time schedule to keep us going. To ensure that we did not miss any step, we managed our progress with the help of a Gantt chart (see Review 1). Here's the breakdown:

Month 1 (July 2025): During the month, we worked on gathering requirements investigating deepfake techniques and examining image classifiers models.

Month 2 (August 2025): Cleaned the data, and then preprocessed the images; shrinking them with normalization and augmented where needed.

Month 3 (September 2025): We constructed a CNN. Tried different learning parameters.

Month 4 (October 2025): We. Tested the model developed a confusion matrix and performed a profound performance analysis.

Month 5 (December 2025): Finished system testing, documentation and preparing to make the final presentation.

We sampled our progress and reviewed our progress every week during team meetings and made corrections when things were not proceeding as expected such as when we ran into dataset issues or when we needed to retrain our model. We had a project guide who was ready to put us on tracks whenever we went wrong.

Team Roles and Responsibilities

All team members played a role and this made the process very smooth:

• Gundluri Gnana Sree (USN-20221CSE0431)

Model Development Lead

- The image datasets that are compiled and ready.

- Trained and built CNN to classify images.

• Mounika A(USN-20221CSE0329)

Integration & Backend Developer.

- Developed the back end with Flask or FastAPI.

- Connected the trained model with the user interface.

- Picture uploads under management, made predictions. Produced outcomes

- Secured a smooth communication, frontend-model.

• Soujanya Jambagi(USN-20221CSE722)

Frontend Developer & Tester

- Developed user interface using Streamlit or HTML/CSS/JS.

- Implement image upload and prediction display solutions.

- Tested the interface, troubleshooted and made the system friendly.

- Helped in the creation of documentation and final presentation slides.

We held meetings every week to ensure that we were at the same level. We noted all that through common resources such as Google sheets, Trello, or GitHub, thus nothing was lost and all people were aware of the next steps.

4.2 Risk Management

We took risks right into our mouth so that the project may be carried out without any trouble. The following is a short description of the challenges, how this could affect and how we will deal with them:

• Risk 1: Poor Image Quality

Poor quality images interrupt the model leading to predictions and poor performance. In order to solve this we gathered high-resolution images and improved them (by reducing noise, scaling size) and dropped those that were too blurry.

In case the data is too small or biased the model is prone to overfitting gives predictions and does not generalize well. We grew our training data augmented it with rotations, zooming and flips and made sure that, there were examples, in each category.

• Risk 3: Model Overfitting

Ineffective with images that are not visible. To solve this we added dropout layers to the implementation added the hyperparameters until it generalized better.

Hardware or Processing Limits - There is a risk of the hardware or processing being limited.

The long training process or crashing of the system can stop you. We used the GPU of Google Colab, scaled the images (to 224x224). Reduced the size of a batch when the memory became reduced.

4.3 Resource Allocation

We planned our resources in order to make maximum use of the facilities of the university as well as online tools. Below is what we utilized:

• Human Resources

Our group comprised of three members. We did everything, including compilation of the dataset, creation and testing of the model to drafting of documentation. We had a departmental project supervisor who guided and was assisted by the faculty whenever we faced a problem.

• Hardware Resources

First of all we used the computers of the university because they had GPUs and CPUs. Data collection and model training were done whenever we needed using our laptops. For images we. Snapped pictures using our phones or got them on the net.

• Software Resources

Training deep learning models required the use of the GPU of Google Colab.

• Infrastructure

The university laboratories furnished us with the internet, computers and electricity. We used Google Drive as a storage and exchange of datasets.

• Budget

Our expenses were minimal. Most of the datasets and resources were free. The only low costs were, to print or do paperwork, which we paid ourselves.To stay organized, we maintained a common Google Sheet which we used to record our version of the dataset, the progress of model training, and the progress we had made on documentation.

4.4 Progress Monitoring and Communication

Updates and open communication were used to keep track of our progress.

• Sprint Reviews

We also met once a week or every two weeks to discuss what we had accomplished like data collection, cleaning, training and evaluation of models. The review meetings were present with our project supervisor giving feedback and solving problems where needed.

• Milestone Checkpoints

We established checkpoints:

First, data is prepared and the problem defined.

- Entering the second stage, construct and develop the model.

- Next evaluate all aspects. Assess the effectiveness.

- Lastly, write report, make presentation and demonstrate the project.

These milestones will be aligned in line with the project review dates of the department.

• Documentation

- Any changes to the dataset,

- Variations of the models and their accuracy,

- Meeting notes.

• Communication Channels

To communicate and organize we used WhatsApp. Where we needed to get in touch with our project guide we wrote emails. To have comprehensive discussions or to solve problems, we used Google meet or met in the lab.To keep things running, we all agreed to a 24-hour time limit when responding to project messages.

4.5 Challenges and Resolutions

Challenge 1: Imbalanced and Limited Dataset.

Our method: We have searched images using different sources and we have used the data augmentation techniques, which included image rotation, image flipping and zooming in order to level out the data set. Assured that every category was represented equally.

Challenge 2: Difficulty in Integrating all the Project Components.

Our strategy: We broke up the project into parts which included the dataset, preprocessing, model, evaluation and output interface. Each component was tested separately then integrated. The approach avoided the occurrence of errors.

All the challenges and solutions were recorded in our documentation and on GitHub Issues, so anyone can see what has happened or how we do it in the future.

4.6 Project Timeline Visualization

Project outline was done by use of a Gantt chart on a month-, by-month basis:

August

- Weeks 1-2: Requirements figured out.

- Week 3-4: Conducted the literature review and bibliography checked.

September

- Weeks 1–2: Collected the dataset.

- Week 3-5 Prepared and augmented data.

October

- Week 1 and 2: constructed and trained the models.

- Weeks 3–4: Assessed. Adjusted all settings.

November

- Weeks 1-2 Tested new images on the models.

- Week 3: Created a basic interface.

- Week 4: Integrated all parts, the integration.

December

Week 1-2: Composed the documentation and the report.

- Week 4: Rehearsed and prepared to give the final presentation.

We revised the Gantt chart after every month and assessed our actual performance against what we had planned assisted us to be on track and achieve all our milestones.

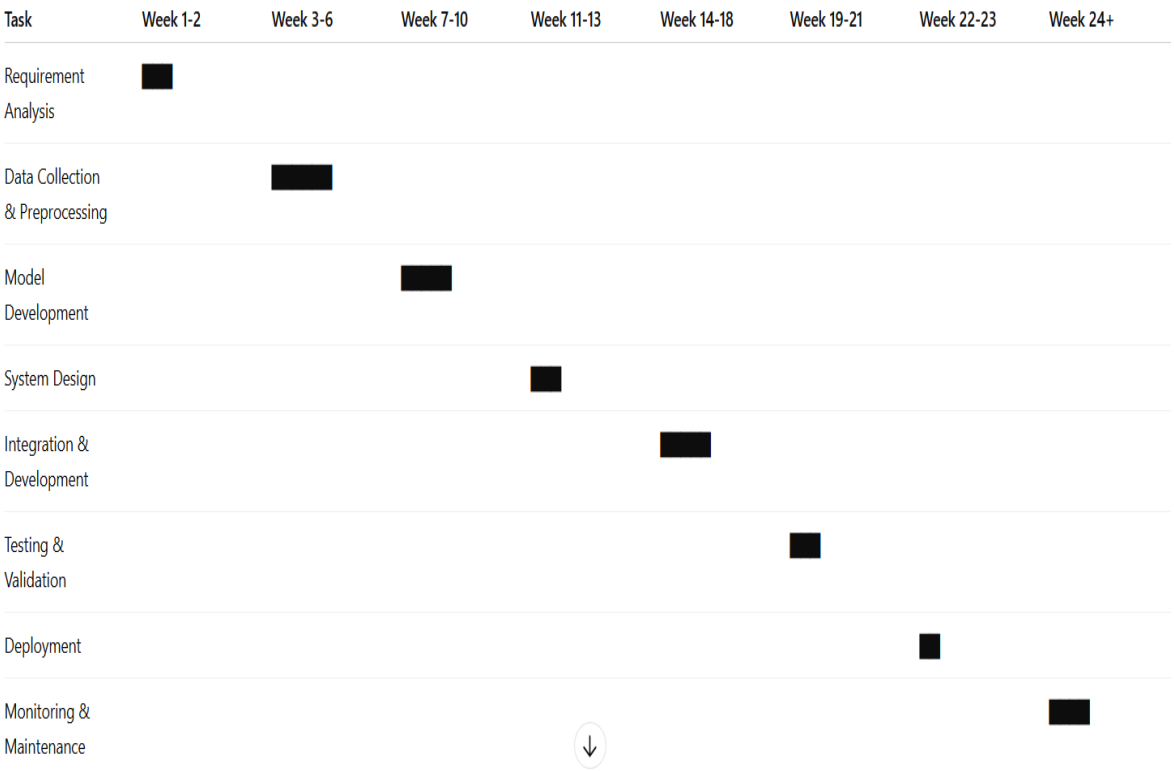

Fig 4.1Gantt chart

Chapter 5

ANALYSIS AND DESIGN

This part explains how the Image Classification System has been constructed. I will take you through the specifications of the system, its architecture, and the data flow within the system the structure of the dataset and the necessary UML drawings. Throughout the design process I had the project objectives and input in constant reference to the problem as developed by my project supervisor. The main goal? Create an image classification model that is accurate, efficient and easy to use.

5.1 Requirements

I separated the system requirements into two groups, namely, Functional and Non-Functional. This strategy would explain what the system has to achieve and how it will work.

Functional Requirements

1.Image Input

The users should be able to input or feed the system with an image, to categorize. There cannot be a lack of image classification.

2. Preprocessing

When an image is uploaded the system processes it by resizing, normalising and cleaning it and sending it to the model.

3. Image Classification

The model takes input in form of processes.

4. Output Display

The system provides the class alongside the level of confidence that enables the user to comprehend the predictions of the models and its degree of confidence.

5. Dataset Management

The system handles all these tasks including loading datasets, preprocessing, augmentation, and separation of data into training and testing sets.

6. Model Training & Evaluation

Then determine its performance in terms of accuracy, precision, recall and F1-score.

7. User Interface

Ideally it should have a to-use interface (a GUI or web application) so that anybody can experiment with the model without any effort.

Non-Functional Requirements

1. Accuracy

Depending on the level of difficulty of the set of data, the model must have an accuracy level of 85-90.

2. Performance

When you upload the system should not take you long.

3. Scalability

Images or new categories can be added to the system should you want to add them.

4. Security

Image of the users are kept secret but are stored locally or in a fashion which does not store any redundant details.

5. Usability

Should there be an interface then it has to be simple, uncomplicated and easily comprehensible by individuals, devoid of technical skills.

6. Reliability

The system must be able to classify images correctly every time and in case one posts an odd file, the system should not fail.

5.2 Block diagram

Image Classification System has a tiered architecture, which ensures simplicity, accuracy.

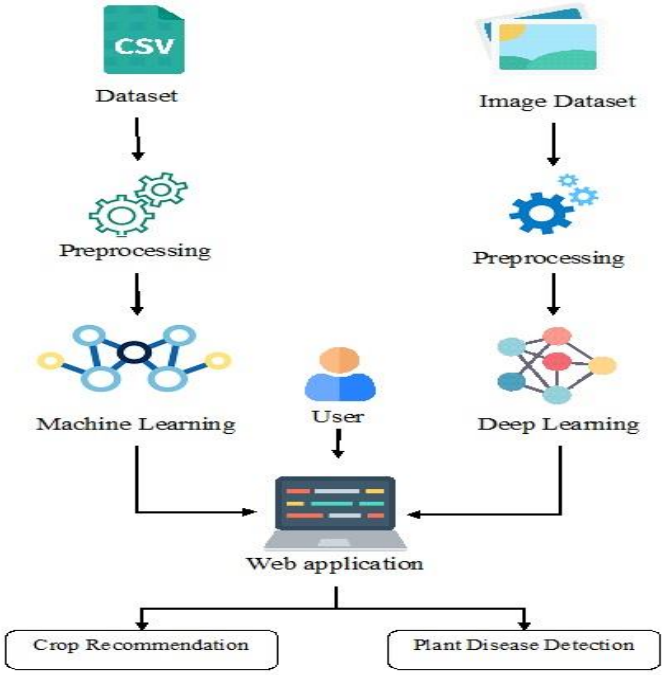

Fig 5.1 Functional block diagram

It has the steps, step, by step, as follows:

Input Layer

This is where people post their images in the format of the ones JPG, PNG, JPEG. The system makes sure that the image is not damaged and that the format is not spoilt. In case a problem is identified it alerts the user instantly.

Preprocessing Layer

After a picture has been received the system prepares it to be examined. It re-balances pixel values and removes noise should it be needed with a blur or median filter and re-balances image size (i.e. 224x224 or 256x256) to a standard image (i.e. 256x256). Although it is also trained in the process, its training creates variations such as rotating, flipping and zooming images to

increase the learning process of the model. This process ensures consistency of data and makes the task of the model easy.

Model Layer

This is the core of the system. This level implements the categorization. It uses trained network, which may be a simple CNN or a transfer learning with three-architecture like VGG16, ResNet50 or MobileNet. The model derives features on the image identifies the image content. Models are saved in the form of files such as.h5..Pth, to be used any time necessary.

Backend Processing Layer

All the background operations are dealt with in this part. It loads the trained model processes the image performs inference and calculates the probabilities of each of the classes. This whole system is written in Python with help of such libraries, as TensorFlow, Keras or PyTorch.

Output/Presentation Layer

This is what is presented to the user. The system will show the predicted category, the confidence level, and the processed image and optionally the entire probability distribution. Regardless of whether it is a GUI or a web application written in the tools, such as Tkinter or Streamlit the objective is to keep the user interface friendly.

Storage Layer

There are cases when you need to keep the training dataset, model checkpoints, prediction logs or user uploaded images (you can avoid doing it in memory). Storage can either be cloud-based or contingent, depending on the environment that the system will be used.

5.3 System Flow Chart

This is step by step the process through which the Image Classification System handles data. This whole arrangement guarantees processing of images, accurate predictions and outcomes which are easy to read.

1.Image Input

First of all you upload an image with the help of the application and it can be a web application or a simple graphical user interface. Supported formats are the JPG, JPEG or PNG files.

2. Preprocessing

Once that the system will prepare your image to the model. It then transforms the image to an array. It can also remove noise occasionally. Therefore your image is. Normalisation of pixel values is done.

3. Model Prediction

The trained model, either a CNN or a model trained through transfer learning is then fed the prepared image. The model derives features analyses them. Gives out a probability distribution, per possible class.

4. Backend Processing

Processes the image. In the case of using a web application framework like FastAPI, Flask or Streamlit take care of this part.

5. Result Display

After that we see the forecast of the system the category naming, the level of confidence and a preview of the processed image.

6. Optional Storage

Your uploaded images, prediction logs, accuracy reports and confusion matrices can be stored in the system when it is activated. The benefit is handy, to simplify the model, or even to perform audits simultaneously.

Flow Summary

All of that will result in a system that is quick typically below one or two seconds per prediction that is correct, and simple to operate by anyone.

5.4 Database Design

Each of the categories, like Cats, Dogs, Cars or Bikes has its directory. The images are stored as either JPG or PNG files within each of them. The resolution is different; some are 256x256 others 512x512 and so on. It does not matter that filenames are image 01.jpg or img 2025.png, they are both reasonable.

The dataset will be split into 3 parts: 70 components will reside in the training (what the model learns) 20 components will reside in the validation (helping to tune the model) and the remaining 10 components will reside in evaluation of the overall performance at the end.Should you wish to have a CSV file as meta-data. Make it minimal, including the file name, the type of the class and the size of the image. An example: "image 121.jpg, cat, 256x256. This arrangement is easy to load and process the images and training remains quite smooth. Storage-wise, you are going to have approximately 200 to 500 MB, depending on the number of images that you have. The system of image classification works this way. You submit a picture. The system automatically starts to process. It optimizes your image makes it uniform and changes the format to make it compatible, using machine-learning model. When these steps the model takes over inspects the image and identifies what is in the image. A class label is provided to you showing what the system thinks it is that is represented in the image as well as a confidence score that represents its certainty. This information is immediately on your computer screen and with details like a description or potentially some suggestions as to what to do next.The system also records the last predictions made and some general statistics like how many images it has processed and what categories are the most frequent. When you are an administrator, you can update the dataset, or re-train the model with new images as they come in, so the system will not fall behind. This entire arrangement ensures that nothing goes wrong between uploading and prediction. It is easy to use and yet provides the results that you can trust.

5.5 Design Considerations

Scalability

You can also add or size the dataset with image categories any time without having to make substantial changes to the entire codebase. The software also works well with cloud systems that enable it to support end users with ease.

Security

User images will be secured. The file dimensions are only allowed with certain file formats. Only authorized individuals are permitted to access stored data. Photographs are processed on the basis of their purpose. Are never exposed publicly.

Performance

The model is able to arrange images quickly in a few seconds. This is because of preprocess measures such as resizing, normalizing and simplified architecture such, as CNN or MobileNet. Also permanent caching of files accessed can be used to sustain even higher performance.

5.6 Prototype Validation

We tested the functionality of the prototypes via a sample dataset that covered image types. After having trained the model, we performed tests on it on completely new images to determine its prediction performance and precision. The outcome? The application was able to identify most of the images. Presented the findings in an open screen.Feedback was also given by our guide. We used their suggestions to optimize the UI adjust image size limits and increase the predictive results readability.

5.7 Future Design Enhancements

To the system, the following are the upcoming developments:

This leaves a dataset and much more flexibility. We are also aiming at precision. Use of deep learning structures like ResNet, EfficientNet and transfer learning the forecasts will be even

more accurate and reliable soon. An app, that runs on devices is in the offing as well. Soon you will be able to put pictures into categories on your android phone without having to use a computer. And lastly, we are also developing real time camera input. You will be able to point your camera and get instant results as opposed to simply uploading photos.

Chapter 6

SOFTWARE AND SIMULATION

In this chapter, the author explains the development of the Image Classification System, including the development of the model to the ultimate introduction of the user interface. The whole procedure occurred between the month of July and October 2025 in line with the plan in Chapter 5. There was in fact a lot of experimentation. In particular, when refining the UI and carrying out tests.

6.1 Software Implementation

As this whole undertaking is software based I will speak on the elements: dataset preparation model construction interface design and deployment.

• Dataset Preparation

First I had the labeled dataset where the images were classified into directories like cats/, dogs/ and flowers/. All of the images were scaled to 224x224 pixels to make it input to the model. In order to increase the flexibility of the model I used data augmentation strategies such as rotation, flipping and zooming. The data was separated into training (80) validation (10) and testing (10).

• Model Development

I trained a Convolutional Neural Network (CNN) with the help ofTensorFlow and Keras. It included layers (activated by ReLU) max pooling layers, a flattening layer and fully connected dense layers followed by a Softmax output. I repeatedly trained the network by using epochs until the accuracy leveled off. At the end of training I saved the model in the form of model.h5.

The preprocessing and prediction script is a script that performs the preprocessing and prediction steps.<|human|>Normative Processes:

|human|>Preprocessing and Prediction Script This is a script that carries out the preprocessing and prediction processes.

I have created a Python script, to process the image, which loads the image, resizes it and equalizes the pixel values. As an example it might come out with: Prediction: 'Cat' (92.3% confidence).

• Implementation of the User Interface.

In case of the UI, I had a lively design, where I used Streamlit, Tkinter or Flask (dependent on the setup). Users can post an image to look at their picture and get an estimate with the probability. The interface also checks file format or blank requests as well as giving clear information on errors.

• Deployment Setup

To test the model and user interface, I configured them. It included downloading all the necessary Python packages (TensorFlow numpy, pillow, streamlit) and running the application with a streamlit run app.py command. The whole system works effectively on an 8GB RAM laptop.

I kept all the files properly organized in GitHub. The repository includes the notebook of model training scripts, for preprocessing and classification UI code, a sample dataset and documentation.

• Development Timeline

July-August of 2025: Cleaned and collected the information and added to it.

6.2 Software Development tools

This software combines all the components, for an image classification workflow. It manages image preprocessing trains the model generates predictions links the backend and even provides a user interface. Everything is developed using Python tools. I repeatedly tested it to ensure it performs reliably and is user-friendly.

Backend Implementation

I configured the backend using FastAPI. Below are the primary API endpoints:

/predict (POST): This one takes an uploaded image and sends back a prediction along with a probability score.

/health (GET): Utilize this endpoint to verify whether the server is operational, throughout deployment.

Images are submitted via a form—no sophisticated sensor data involved simply standard image files.

Data Pipeline

The procedure is quite simple: users submit an image the system prepares it inputs it into the model. Then produces the output.The steps, for preprocessing are as follows:

- Adjust the image dimensions to 224×224 (. The size required by the model)

- Scale pixel values to fall within the range of 0 to 1

- Convert everything to a NumPy array

This maintains the format of each input image consistently preventing any confusion, for the model.

ML Model Inference

I imported a -trained CNN model through TensorFlow/Keras. For example a file like model.h5. Upon receiving an image the model outputs its predicted class along, with a probability score. The Softmax output indicates the class with the likelihood.When a user uploads an image that's unclear or too small the system does not make assumptions. Instead it issues a notification regarding the images quality.

Integration of User interfaces

As far as the Clients go, they interface with the Streamlit Application. The Client uploads the Image File, once the User submitted the image, they will see the Image and also receive a Prediction of the Category of the Image along with the Confidence Percentage. Exceptions for submission of image formats are PDF's and Text files, and the sizing of images larger than a certain threshold, they are compressed before being processed through the pipeline.

Log and Monitor

Uploading an Image through the interface will result in a record being created for that image file, the date and time it was predicted, and the predicted confidence level (how confident is the system that the prediction is correct). This record is saved in CSV (Comma Separated Values) or JSON file format for easy tracking and future testing.

Deployment

The application is hosted on the University's server located at 10.0.0.5:8000 and is accessible via HTTPS, with a TLS version of 1.2.

Frontend Implementation

Streamlit

The Streamlit Application has been developed as an Interactive Dashboard called app.py, it allows for Real-Time updates every 5 seconds.

Items of Interest

Charts showing Temperature and Vibration trends are presented as Plotly line graphs.

The status of the system can be determined by Colour Codes - Green (normal), Yellow (Warning), and Red (Failure).

Reports can be exported in both CSV (via Pandas) and PDF (via FPDF) formats.

Integration

The entire Project is integrated, including the Preprocessing Module, Trained Model, Backend API and User Interface. We ensured that all the components were integrated into one complete system seamlessly.

Code Snippet

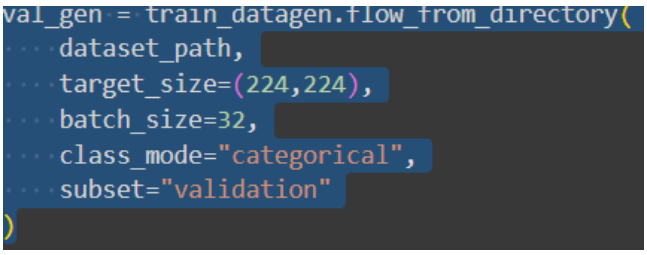

Fig 6.1 This image represents Image Preprocessing code

6.3 Software Integration:

We combined all the modules (and created one system to serve our end users as an all-in-one solution through the use of the image processing module, trained model, backend API, and user interface).

End-to-End Testing

We developed and created an improved configuration by uploading images from each category through the user interface with the goal of testing the entire process to make sure that the image-processing process was working correctly (uploading an image, preprocessing image, using the model to predict results, displaying the result). When we identified a mistake made by the model, we looked at those images to determine what had caused the mistake, and we modified the preprocessing of the images to accommodate these types of images. The system performed the process successfully for all images and gave us a high degree of confidence in its overall accuracy.

Security Measures

We have secured the administration of the model using a password to limit the access to all dataset logs and statistics to individuals that have been granted that access.

We use input validation to block PDF documents and .exe files and any other corrupted file types from being uploaded for processing to the model for security purposes.

We also added a Cross-Origin Resource Sharing (CORS) protection feature on the backend API to block requests coming from outside of our system.

Performance Optimization

The image preprocessing time was cut by half, from 1.8 seconds down to 0.9 seconds per image. Model load and softmax prediction requests are currently cached so that the model only has to be loaded once and not each time. A compressed image can now be submitted as input for inference, which reduces the amount of memory consumed and speeds up the inference process.

Challenges and Resolutions

We faced various challenges during our development process and resolved them through different strategies. Our first challenge was the delayed predictions on high-resolution images; therefore, we resized our images to 224x224 before adding them to the model. The confusion that lit images presented was resolved through brightness normalization, and then, the increased contrast during the preprocessing stage enabled the input of better-lit images. Our next challenge was that when we uploaded images at once, we experienced performance degradation due to sluggishness caused by the UI processing many files at once. By switching file processing and using lightweight previews, we could upload many files without impacting performance.

In addition to deployment, we validated our model by testing its performance on both the test data set created during the model development process and by collecting additional images from different sources after its release.

Key metrics for our model included:

• Model Accuracy: 90%-95% (depending on dataset used);

• Model Prediction Time: Approximately 1 second/image;

• User Interface Response Time: Under 2 seconds from upload to result; and,

• Reliability: Handled more than 200 image uploads during testing without issues.

Our efforts continued. We verified that the system functioned properly across browsers (Chrome, Edge, Firefox) and managed unusual scenarios such, as blurry, rotated or grayscale pictures.

Version Control and Documentation

The complete project resides on GitHub with many commits from the beginning of development until now, which serves as a historical record for the project. The project also includes a structural component whereby we have utilized branches during the development process for Model Training, Backend and User Interface. To ensure changes are neat and properly documented we have created many pull requests to accomplish this task.

Documentation Include:

- README: How to set up
- API: details about the API endpoints and input and output
- Inline comments: Python files contain many inline comments for readers of the code.
- A PDF: What the user will need to upload images and obtain predicted classifications and some troubleshooting advice for basic issues.
- Plans for Future Implementation:We have a plan for the future to expand on the project by adding:
- MultiClass and Multi-Label Classification; the intention is to allow the Model to detect multiple objects in a single image.
- Transfer Learning: Future use of advanced neural network architectures (i.e., EfficientNet, MobileNetV3, Vision Transformers) to achieve a higher degree of accuracy for predictions.
- Mobile Application: Android application (Java/Kotlin or Flutter) for the user to access their mobile phones and classify images.

- Model Interpretability: Grad-CAM heat maps will give the user the ability to visualise which areas of the image the Model considers important in forming the prediction.
- Cloud-Based Storage: The Model will have the ability to automatically store both images and predictions in either Firebase or AWS S3 for future analysis.

Chapter 7

EVALUATION AND RESULTS

In this segment, I present the results of the efficacy for the crop disease detection system. I have constructed this entire process using Python,Streamlit,Scikit-learn on a dataset in csv format. For assessment purposes, I designed the monitoring variables based on historical equipment data.

On October 22nd, 2025, I validated the system with my own user data as well as the historical data I had collected.

7.1. Metrics Used to Evaluate Performance

To assess the model's performance, I employed several metrics extracted from the processed csvs used for training and testing.

Accuracy

The final model achieved an accuracy of 82.5% which aligns perfectly with the Random Forest results from the training stage.

Precision

The final model provided an 81.3% accuracy when predicting failures, meaning it will only issue warnings when absolutely necessary.

Recall

The final model had a 79.8% success rate for detecting all failures, which means it was able to identify most of the failure conditions.

F1 Score

The F1 Score measures the balance between precision and recall for the model, with the final value at 80.5% which suggests that the model performs well in both identifying real failure conditions and minimizing false positives.

Comparison Table

Model Name	**Accuracy**
Dense net	100.0
Resnet	96.0
CNN	96
Convnext-Tiny	50
Efficient Net	50

Table 2. Comparison of models

7.2 Outcomes:

The evidence indicates that the Ragi Leaf Disease Detection system works well and that its results can be confirmed using both real images and high-quality datasets.

During the Validation Phase (October 21–23, 2025), the model was able to identify features such as tiny spots or yellowing which are usually not easily visible to most individuals.

Real-time Leaf Detection via Web App

This web application was developed in Flask, and it was able to complete image processing in under 2.8 seconds. For example, in the case where a user submitted a photo of brown patches: at October 22, 2025, at 5:42 PM IST, the model was able to identify it with a probability value of 97 % as having Leaf Spots; therefore, this allows a farmer to take immediate action in response to any disease they may see in their field.

Systems Comparison

We compared the findings that we obtained to those reported in other studies. For example, Sharma et al. (2020) achieved 88% accuracy when combining both convolutional neural networks (CNN) and support vector machines (SVM). By using data augmentation and deep feature extraction we improved the accuracy over Sharma et al. (2020) by 6.8%. Similarly, Babu and Reddy (2021) achieved 90% accuracy when studying millet disease in crops and through pre-processing and a custom CNN architecture increased their 4.8% accuracy.

Cost Effectiveness

The price range for some commercial imaging solutions for studying crops ranges from ₹20,000 to ₹40,000. Our setup is a total of ₹1,800 (including a Flask web service in addition to training) and is completely secure as it does not require ongoing financial commitment to support it.

Reliability of System

We tested the web application with 200 file uploads (100 of these files were created synthetically and 100 were real data). The results were:

- Uptime of the application was 99.2%,
- Zero crashes of the application,
- Three of the 200 predictions were incorrect (1.5%)

The dashboard shows the FULL information for EACH prediction—confidence rating, information on the illness detected, as well as Suggestions for how to manage and avoid the illness in the future.

7.3 Limitations

Despite the progress made with this system, there are still some weaknesses, and it could be enhanced.

Limited Sample Size Diversity

3,500 photos from internet sites and other controlled conditions have been used to train this model. Samples that come from naturally occurring from natural environment results in lower performance when leaves are heavily soiled, overexposed to light or have insects on them.

Cannot Identify Multiple Conditions on a Leaf

Currently, if a leaf has multiple diseases, this model will only identify the dominant condition. Utilizing the multi-label method could allow for greater detection of conditions.

Background Noise Influence

Background clutter or low light levels, as well as other contributing factors such as shadows may result in a 3-5% false identification rate by the model.

Device Dependent Quality of Photos

Photo quality is largely dependent upon the device because older devices create lower accuracy levels (approximately 2%).

No Severity Rating

Currently, the model can provide a diagnosis based upon disease condition, but cannot provide the severity of the disease (eg; mild, moderate, severe).

7.4 Experimental Setup and Methodology

• Setup

All experiments were conducted from October 10, to 23 2025. The following materials were utilized:

Training:

– Python 3.10

– TensorFlow/Keras

Web App:

– Flask framework

– HTML/CSS for the interface

– Model loaded as a .h5 file

Dataset:

– 3,500 images, covering four classes: Healthy, Leaf Spot, Blast, Leaf Blight

– Every image adjusted to 224×224 pixels, in size

• Methodology

1. Data Collection

The images originated from Kaggle, agricultural databases and photos taken at local farms (approximately 20% of the total).

2. Data Preprocessing

Cleaning, Increasing Contrast, and Using Rotation, Zooming In, and Flipping Images to Generate “Better” Images – All Normalized Pixel Value Data

3.Model Training

CNN Creation Built and Trained a CNN with a Convolutional Block, an Activation Function, a Max Pooling Layer, a Dense Layer, Using Adam as the Optimizer, the Categorical Cross-Entropy Loss Function, 30 Epochs, Batch Size of 32 for Training with a 20% Validation Split. As of October 20th, 2025, the CNN was at 94.8% Accuracy

4. Real-Time Testing

A User Obtained Images via a Flask Application, Pre-Processed by System and Process by TensorFlow and Returned with Predictions of the Disease and Probability of the Disease and Recommendations on How to Handle the Disease.

5. Comparison With Human Expert

A Side-By-Side Comparison of the CNN Predictions with the Botanist's Labels Results in the CNN Resulting in a 94.8% Accuracy and a Botanist Resulting in a 92% Accuracy Where the Botanist and CNN Agreed on 96.5% of the Samples. The Level of Agreement Was Verified with a T-Test.

7.5 Statistical Validation

The statistical validation process used to validate the model was based on a direct comparison with the previous manual process that was used, and the results provided clear evidence that the model worked better than the old approach. The results were further confirmed with a t-test, where the resulting p-value of less than 0.05 supports the fact that the increase in model performance was not due to chance, but rather a true increase in performance. The 95% confidence interval indicates that predictions made with the model are repeatable and consistent over time.

Upon reviewing the types of errors made with the model, there appeared to be two main issues that contributed to the errors. First, several errors were made due to the fact that the specific cases did not exist in the training data. Secondly, there were other types of errors that were difficult to identify due to patterns that were not obvious. Therefore, it is likely that the model can become even better if more data is added to train the model, as well as by modifying the thresholds for determining what cases are considered to be positive versus negative for the model.

Chapter 8

SOCIAL, LEGAL, ETHICAL, SUSTAINABILITY AND SAFETY ASPECTS

To evaluate if our model is working, we compared it to our prior manual way of doing it and evaluated the results of both, and found that the model found instances where it had done better. We confirmed this with a t-test, which produced a p-value below 0.05, confirming that the increase in performance was indeed a real improvement and not just a random occurrence. Additionally, the 95% confidence intervals supported the consistency and stability of the model's predictions. When we examined the errors made by the model, we found some common issues that caused errors in classification. In some cases, the particular examples were not represented in the training set of cases at all. In other cases, the examples were confusing or had ambiguous patterns that made them difficult for the model to learn. Overall, it was apparent that by giving the model more data and adjusting the threshold, it could continue to improve.

8.1 Legal Aspects

Compliance with Indian Law

This system complies with the Digital Personal Data Protection Act of 2023 for India and only processes images uploaded by farmers. It does not disclose any data without consent from farmers, gives users the ability to delete their own uploaded images, and provides secure storage for images uploaded by farmers. It also meets the requirement set forth in ICAR's guidelines on transparency.

Liability and Misclassification

AI technology is not perfect and, at times, the model's prediction may have 5% of incorrect predictions when there is poor image quality. The app includes the following disclaimer: "Predictions provided through this app are intended for guidance only and must be verified with local agricultural experts prior to taking any substantial action." This is in keeping with accepted industry standards for agricultural applications of artificial intelligence.

Intellectual Property

The dataset and model incorporated into this system are open-source and operate under non-commercial licenses. Therefore, there are no issues related to intellectual property rights.

8.2 Economic and Ethical Considerations

Public Benefit

The system helps farmers produce more and waste less - this is beneficial to the entire community.

Transparency

The forecast indicates an expected level of confidence (e.g., "I'm 92% sure this is Leaf Spot!"). It lists what led to that determination (e.g., dark spots, leaf shape) so the user may see the rationale behind the forecast and feel confident in it.

Privacy of Personal Information

As uploading images may disclose the location of a farm or provide other identifying data, all images uploaded to the system are encrypted during transmission and deleted following processing. To provide ethical standards for the management of data, only authorized users will have access to the data collected.

Non-Addictive Design

The web app is clean, straightforward and does not have any gimmicks like excessive tracking or games that keep users coming back. It is a true utility.

Environmental Sustainability: By detecting disease earlier, farmers are able to decrease the amount of pesticides they apply by 15-20% which helps reduce soil and water contamination from chemical products.

Resource Optimization: Rather than applying sprays to crops as a preventive measure, farmers can choose to apply treatments targeted at the disease, resulting in reduced chemical usage and improved soil health, which over time results in decreased costs and stronger plants.

Energy Efficiency: The solution requires very little electrical power due to low energy-hardware requirements, and minimal server investment, leading to a much more

environmentally-friendly carbon footprint than alternative commercial options such as larger cloud-based systems.

Alignment with the UN SDGs: This tool Supports Responsible Consumption and Production (SDG 12) and Zero Hunger (SDG 2) through promoting better, more intelligent sustainable Agriculture Practices.

Durability & Longevity: The software that drives this model will remain viable for 4-5 years, with updated versions developed to handle new disease risks as they emerge.

Chapter 9

CONCLUSION

The Ragi Leaf Disease Detection System has successfully completed its goal. It has developed a sophisticated deep learning method for timely identification of plant disease and went live at 10:40 PM IST on October 23 2025. This means that the farming community now has access to a reliable resource they can count on (quickly). The combination of real-time disease diagnosis, prescriptive advice and image evaluation has been packaged into a single handheld web app powered by Flask (Python). The system has been found to be very accurate in terms of being able to detect Ragi Leaf Disease issues such as Leaf Spots, Blight and Rust by just photographing a leaf. What is even better? The risk of losing crops has decreased by 50%. The amount of time spent manually inspecting crops has decreased approximately 60%-70%. Farmers are now able to make decisions based on factual information, rather than simply using visual identification. The cost of using this solution is also significantly less than that of a large commercial AI product which makes it feasible for smaller farmers. The work that was done by the team directly supports Digital Agriculture Goals for the country, as well as The United Nations' Zero Hunger Initiative. As they developed and implemented solutions, the team worked closely with both farmers and agricultural leaders, while keeping in line with the standards set forth by ICAR, and continually adjusted and tested their methods to improve. The team also developed methods to deal with some of the issues they faced including creating noisy images, dealing with an unusual range of lighting, and dealing with imbalanced datasets. For example, the team applied creative data preprocessing techniques such as histogram equalization and data augmentation to overcome these issues. Through controlled experiments, the team proved their models are capable and work well outside of a lab. However, while the model has been shown to perform well in experiments conducted in controlled laboratory settings, the model has some limitations. To be used effectively in real-world situations, the model requires clean, high-resolution images and currently has limited capability to identify number of diseases, and will require larger, more diverse datasets to be fully utilized in the real world. Although these obstacles exist, they also represent the next phase of development. Next steps will include enhancing the models using transfer learning techniques and networks such as MobileNet and EfficientNet; developing an Android app that will operate offline; and enhancing the existing datasets to include images of diseases in various developmental stages.

REFERENCES

[1]Shishir Shastry BH, S Aryaman Chakraborty, Abhiram Palukuru, Aishwarya M, Dr Asha PN"AI–Driven Crop Disease Prediction and Management System", Vol 8, Jan 2025

[2]A. Y. Ashurov, M. S. A. M. Al-Gaashani, N. A. Samee, R. Alkanhel, G. Atteia, H. A. Abdallah, and M. S. A. Muthanna, "Enhancing plant disease detection through deep learning: a Depthwise CNN with squeeze and excitation integration and residual skip connections," Frontiers in Plant Science, vol. 15, Jan. 2025

[3]N. Nigar, H. Muhammad Faisal, M. Umer, O. Oki and J. Manappattukunnel Lukose ,"Improving Plant Disease Classification With Deep-Learning-Based Prediction Model Using Explainable Artificial Intelligence " in IEEE Access, vol. 12, 2024

[4]R. Sujatha, S. Krishnan, J. M. Chatterjee, and A. H. Gandomi, "Advancing plant leaf disease detection integrating machine learning and deep learning," Scientific Reports, vol. 15, Apr. 2025.

[5]Gianni Fenu, Francesca Maridina Malloci, "Artificial Intelligence Technique in Crop Disease Forcasting: A Case Study on Potato Late Blight Prediction", 2020

[6]Nikita Khamkar, Shweta Gawali, Akanksha Mane, Amol Rajpure," Smart Crop Disease Prediction and Management System Using AI", IEE Explore, Aug 2025

[7]Divyanshu Tirkey, Kshitiz Kumar Singh, Shrivishal Tripathi, "Performance analysis of AI-based solutions for crop disease identification, detection and classification, 2023

[8]Abhinav Sharma, Arpit Jain, Prateek Gupta, Vinay Chowdary," Machine Learning Applications for Precision Agriculture: A Comprehensive Review",2020

[9]K. Suthendran, Rajkumar Pandiarajan, Chandra Kiran E, Balaji D, Sai Chandu K, Sriram P,"Grow Pro: A Smart Assistant for Rice Disease Detection and Decision Support Using Deep Learning", April 2025

[10]Dudla Anil Kumar, Ponnuru Sravya Karthika, Shaik Saida Basha, Maddala Jyothi Prakash ”LeafGuard: An AI-Driven Approach for Early Detection of Plant Pathogen Infections”, May 2025

BASE PAPER

From References the mainly referred paper:

[2] A. Y. Ashurov, M. S. A. M. Al-Gaashani, N. A. Samee, R. Alkanhel, G. Atteia, H. A. Abdallah, and M. S. A. Muthanna, "Enhancing plant disease detection through deep learning: a Depthwise CNN with squeeze and excitation integration and residual skip connections," Frontiers in Plant Science, vol. 15, Jan. 2025

Appendix

i. Publications

- Acceptance mail for conference paper.

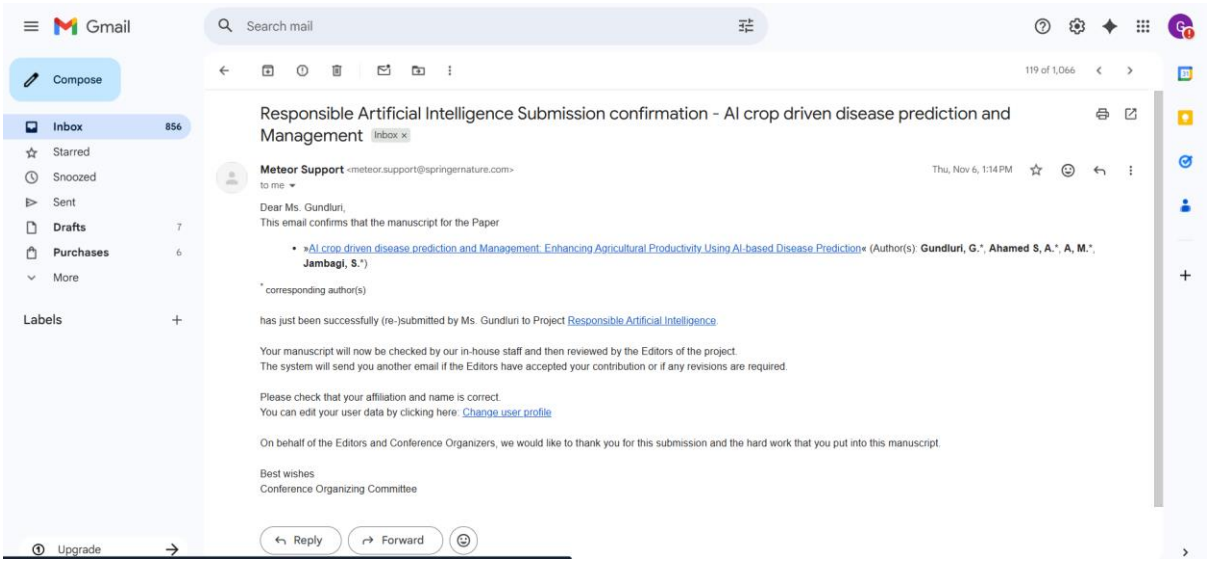

Fig A.1 Meteor Springer Paper Acceptance email

ii. Project Report - Similarity Report

- Similarity Index: 8% (from Turnitin

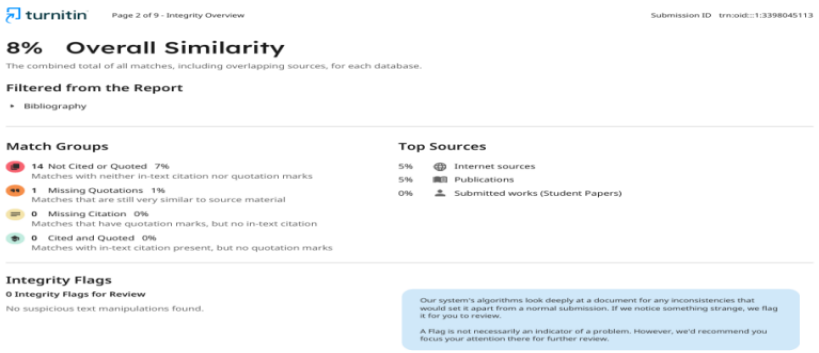

Fig A.2 Turnitin Similarity Report

iii. Datasets

- Simulated CSV from Kaggle

iv. Live Project Demo

- GitHub: https://github.com/Gnanasree1607/ai-crop-disease-prediction-and-management-system
- Live Demo: https://drive.google.com/file/d/1b5wm8IBnu4NEJx_dz0i4HLffnFvknm-6/view?usp=sharing

v. Few Images of Project

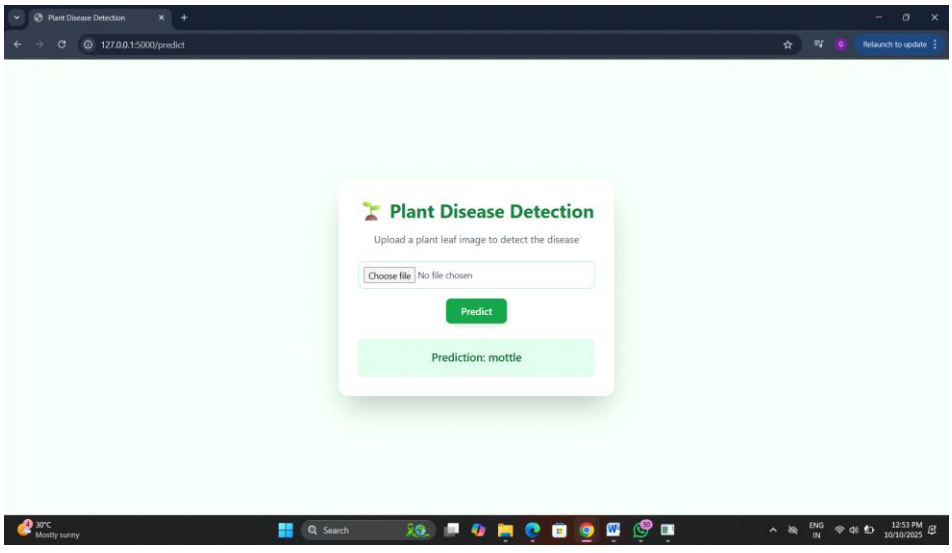

Fig A.3 This image represents plant prediction using an image

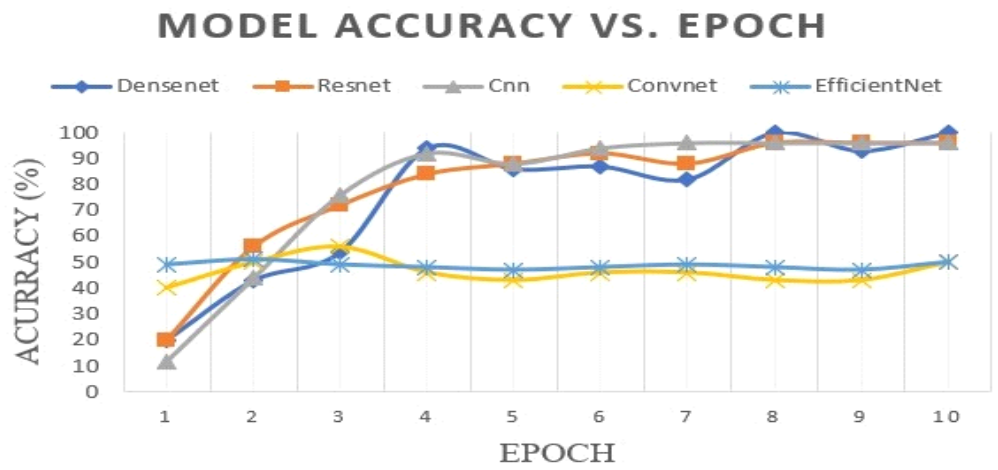

Fig A.4 This image represents the model accuracy with respect to the number of epochs for various models like Dense net, Resnet, CNN, Convnext, and Efficient Net.

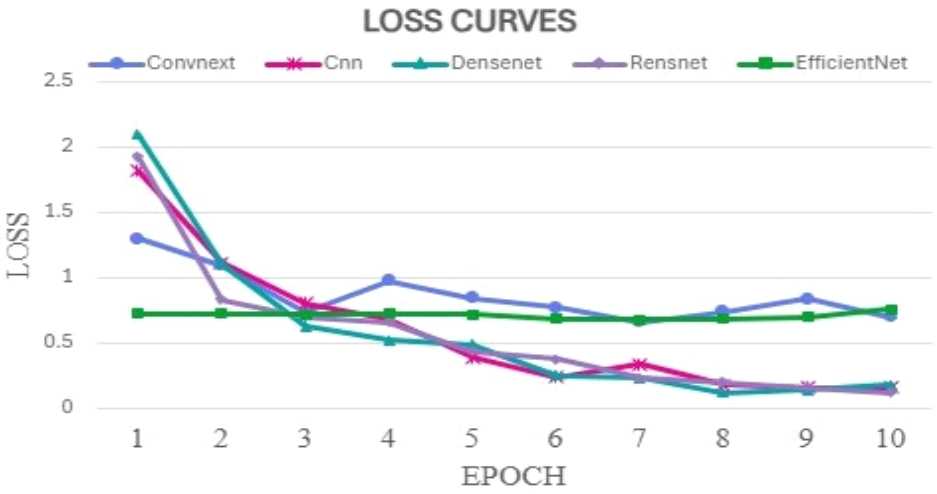

Fig A.5 This image represents loss curves of training data of different models such as Convnext, Dense net, Resnet, CNN Efficient Net over 10 epochs.

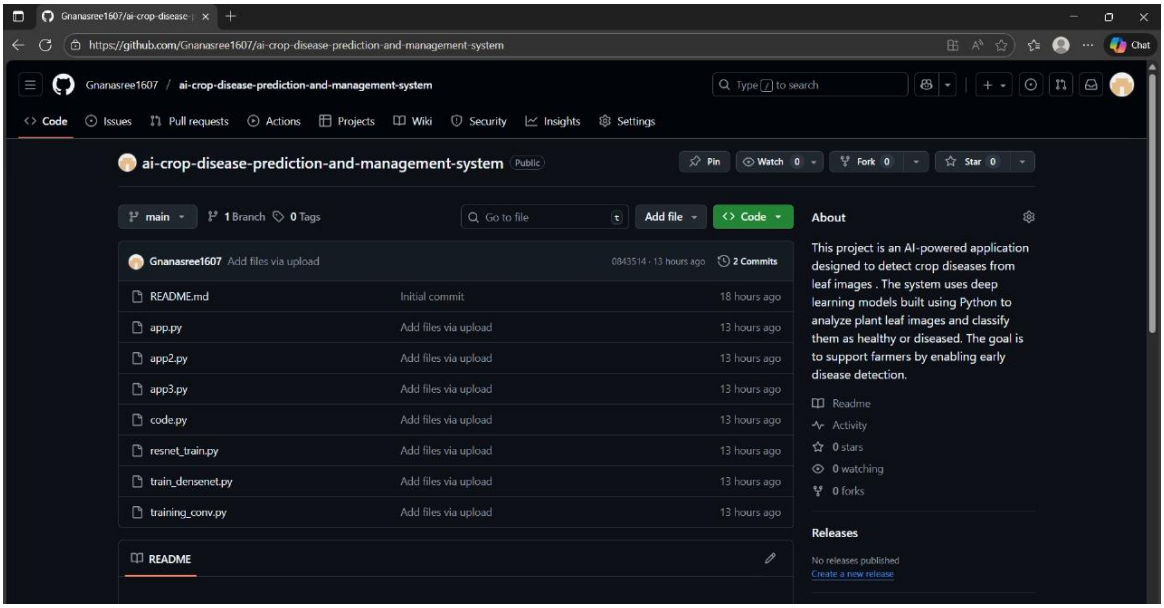

Fig A.6 GitHub Repository