Multi-Class Mango Leaf Disease Diagnosis Using an Optimized Vision Transformer Model

A PROJECT REPORT

Submitted by,

Monica V – 20211CSG0067
Yashaswi A – 20211CSG0015
Thanya Patel R – 20211CSG0043

Under the guidance of,

Dr. Madhusudhan M V

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND TECHNOLOGY

At

PRESIDENCY UNIVERSITY

BENGALURU

MAY 2025

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

CERTIFICATE

This is to certify that the Project report **Multi-Class Mango Leaf Disease Diagnosis Using an Optimized Vision Transformer Model** being submitted by Monica V, Yashaswi A, Thanya Patel R bearing roll number(s) 20211CSG0067, 20211CSG0015, 20211CSG0043 in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in Computer Science and Technology is a Bonafide work carried out under my supervision.

Dr. Madhusudhan M V
Associate Professor
School of CSE & IS
Presidency University

Dr. Saira Banu Atham
Professor & HoD
School of CSE & IS
Presidency University

Dr. MYDHILI NAIR
Associate Dean
School of CSE
Presidency University

Dr. SAMEERUDDIN KHAN
Pro-Vc School of Engineering
Dean - School of CSE & IS
Presidency University

PRESIDENCY UNIVERSITY

SCHOOL OF COMPUTER SCIENCE ENGINEERING

DECLARATION

We hereby declare that the work, which is being presented in the project report entitled **Multi-Class Mango Leaf Disease Diagnosis Using an Optimized Vision Transformer Model** in partial fulfillment for the award of Degree of **Bachelor of Technology in Computer Science and Technology**, is a record of our investigations carried under the guidance **of Dr. Madhusudhan M V, Associate Professor, School of Computer Science Engineering & Information Science, Presidency University, Bengaluru.**

We have not submitted the matter presented in this report anywhere for the award of any other Degree.

NAME	ROLL NO	SIGNATURE
MONICA V	20211CSG0067	
YASHASWI A	20211CSG0015	
THANYA PATEL R	20211CSG0043	

ABSTRACT

Mango farming is an important component of the agricultural economy of tropical and subtropical countries. Nevertheless, the productivity and well-being of mango crops are frequently undermined by several leaf diseases like anthracnose, powdery mildew, and bacterial spots. Conventionally, these diseases need to be detected manually by experts, a process that is not only time-consuming but also inconsistent and subject to human error. We suggest an automated Mango Leaf Disease Detection System using a Vision Transformer (ViT) model in this project to address these issues.

Vision Transformer architecture utilizes self-attention to learn both global and local characteristics of mango leaf images, thereby being suitable for fine-grained disease classification problems. A total of 3,846 images of mango leaves, both healthy and diseased, from different Indian varieties of mango were compiled in a carefully curated dataset. Preprocessing measures like image resizing, normalization, and augmentation were used to guarantee model stability and avoid overfitting.

The ViT model was optimized and trained with PyTorch and tested using metrics like accuracy, precision, recall, and F1-score. The system reached an accuracy level of 99.1%, surpassing standard CNN models like ResNet-50 and EfficientNet. To deploy it practically, the model was transformed to ONNX format and deployed as part of a FastAPI-based web application to detect disease in real-time from user-uploaded images.

This system not only makes it easier for farmers and agricultural scientists to identify disease but also supports crop surveillance with early intervention, thereby encouraging healthier harvests and more effective farming practice.

Keywords: CNN, Deep Learning, Vision transformer, Gaussian noise, augmentation, confusion matrix

ACKNOWLEDGEMENT

First of all, we are indebted to the **GOD ALMIGHTY** for allowing me to excel in our efforts to complete this project on time.

We express our sincere thanks to our respected dean **Dr. Md. Sameeruddin Khan**, School of Engineering and Dean, School of Computer Science Engineering & Information Science, Presidency University for getting us permission to undergo the project.

We express our heartfelt gratitude to our beloved Associate Deans **Dr. Shakkeera L and Dr. Mythili Nair,** School of Computer Science Engineering & Information Science, Presidency University, and **Dr. Saira Banu Atham**, Head of the Department, School of Computer Science Engineering & Information Science, Presidency University, for rendering timely help in completing this project successfully.

We are greatly indebted to our guide **Dr. Madhusudhan M V, Associate Professor** and Reviewer **Dr. Manjula H M**, Presidency School of Computer Science Engineering & Information Science, Presidency University for his/her inspirational guidance, and valuable suggestions and for providing us a chance to express our technical capabilities in every respect for the completion of the project work.

We would like to convey our gratitude and heartfelt thanks to the CSE7301 Capstone Project Coordinators **Mr. Md Zia Ur Rahman and Dr. Sampath A K,** department Project Coordinators "**Sudha**" and Git hub coordinator **Mr. Muthuraj.**

We thank our family and friends for the strong support and inspiration they have provided us in bringing out this project.

Monica V (1)

Yashaswi A (2)

Thanya Patel R (3)

LIST OF TABLES

LIST OF FIGURES

TABLE OF CONTENTS

CHAPTER-1
INTRODUCTION

Agriculture, which forms the backbone of most economies, is highly dependent on crop health and yield. Of the tropical and subtropical fruits, mango (Mangifera indica) is among the most extensively grown and consumed. Yet its production is seriously affected by leaf diseases that, if not checked, can result in enormous loss of yield and crop health. Conventional methods of detecting such diseases include manual inspection by expert agronomists or farmers. Though this method has been in use over generations, it has some disadvantages: it is time-consuming, susceptible to errors, and highly reliant on expertise.

1.1 Requirement for Automated Disease Detection

With the ever-growing progress in Artificial Intelligence (AI), especially in computer vision and deep learning, technology is increasingly being used to support agriculture. Current plant disease detection systems tend to provide rapid, efficient, and scalable solutions for monitoring crop health. CNN-based methods have been employed for this purpose before, but they fail to extract fine-grained features when the symptoms of diseases are not striking.

To counter these drawbacks, our project suggests the application of a Vision Transformer (ViT) for detecting mango leaf disease. In contrast to conventional CNNs aimed at local features, ViTs employ self-attention for learning local and global dependencies throughout the image. This results in better learning of disease features even under leaf appearance, lighting, and background noise variations.

1.2 What is Vision Transformer (ViT)?

First developed for natural language processing, transformers have been recently applied to image classification. The Vision Transformer (ViT) treats an image as a sequence of patches of fixed size instead of the entire image as a single unit. A patch is considered a token, and its relationships with the rest of the patches are captured through multi-head self-attention. This permits the model to capture intricate visual patterns on plant leaves that could be undetected by CNNs.

1.3 Why Mango Leaf Disease Detection is Important

Mango farming is a significant revenue source for farmers in India and various other nations.

Leaf illnesses like anthracnose, powdery mildew, bacterial spots, and others are capable of inducing major losses unless detected early. The signs could be in the form of spots, discoloration, or mold on the leaf surface that can be confusing to the naked eye. A strong, AI-based system facilitates early detection, timely action, and ultimately, improved crop management.

Major motivations for computerized mango leaf disease identification are:

- Reduced Reliance on Expert Humans: Minimizes the requirement for human inspection.
- Real-Time Diagnosis: Facilitates fast detection of disease.
- Greater Accuracy: Reduces human error and fatigue.
- Cost-Effective and Extensible: After training, the system can be applied to large agricultural fields.

1.4 Uses of the Proposed System

The envisioned mango leaf disease detection system has several applications:

- Smart Farming Apps: Farmers are able to upload photos and get diagnoses instantly through a mobile app.
- Agricultural Monitoring Systems: Government or private agriculture departments can monitor crop health at scale.
- Research and Development: Agricultural researchers can utilize the data for studying the spread and behavior of diseases.
- Early Warning Systems: Alarm can be raised when the disease is found in an area, assisting in preventive measures.

1.5 Leaf Disease Detection Challenges

Although deep learning offers a chance, it also has some challenges of its own:

- Data Collection: Demands varied, high-quality collections of diseased and healthy leaves.
- Model Complexity: Transformers are computationally intensive relative to CNNs.
- Overfitting Risks: For small datasets, the model may not generalize effectively.
- Deployment Constraints: Deploying such systems on low-resource hardware such as smartphones or drones may be challenging.

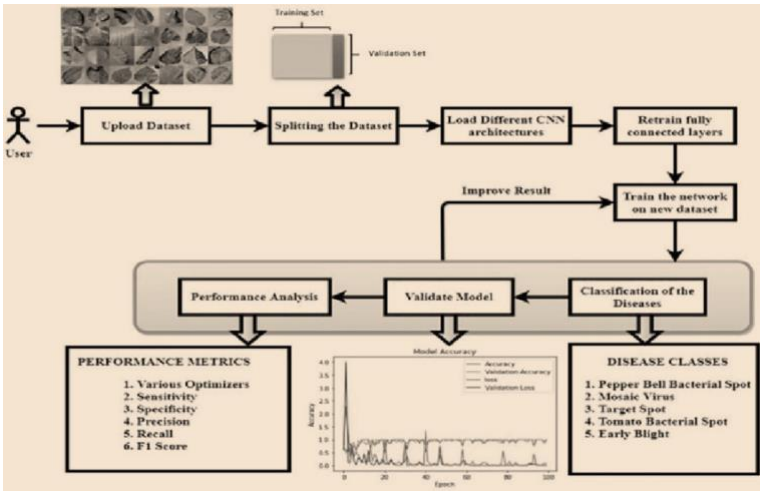


Figure 1.1: How mango leaf disease detection is used

Figure 1.2: extracting leaf features

CHAPTER-2

LITERATURE SURVEY

There has been considerable advancement in plant disease detection research since the development of machine learning and computer vision tools. There are various algorithms, datasets, and model structures which have been designed for classifying plant diseases through leaf images. This chapter includes a survey of current approaches and explains how the proposed Mango Leaf Disease Detection System using Vision Transformer (ViT) advances them or is different from them.

2.1 Conventional Methods Employing Machine Learning

One of the most frequent and one of the earliest approaches in this area is to utilize K-means clustering for segmenting leaf images on the basis of color variations. Merchant et al. proposed a system for detecting nutrient deficiencies in mango leaves through RGB values clustering of cropped images. The unsupervised method served to cluster leaves according to visual symptoms, providing an initial step in diagnosis.

In the same vein, Support Vector Machines (SVMs) and Artificial Neural Networks (ANNs) have been used to classify leaf images into numerous disease categories. For example, Mia et al. integrated SVM with ANN and utilized K-means to separate disease-infected areas before passing extracted features into the model. Although these approaches returned satisfactory accuracy, they were based on handcrafted features and lacked much flexibility towards new or complicated symptoms.

2.2 CNN-Based Deep Learning Models

Convolutional Neural Networks (CNNs) introduced a revolution in image-based classification. Arivazhagan et al. employed CNNs to identify five assorted mango leaf diseases and attained 96.67% accuracy, proving the potential of deep learning to process non-linear visual data.

Mohanty et al. have experimented with transfer learning with pre-trained models like AlexNet and GoogLeNet. Their models, with a huge database of more than 50,000 images, achieved accuracies of up to 99% with GoogLeNet. The success with transfer learning underscored that it is not always required to train from scratch, particularly when huge labeled datasets are not available.

Some other prominent models are DenseNet-121, EfficientNet, and InceptionV3, which have proved to be efficient in plant disease detection. However, CNNs are found to be inefficient at handling long-range dependencies and may need huge amounts of labeled data to generalize sufficiently.

2.3 New Application of Vision Transformers

The Vision Transformers (ViTs) concept proposed by Dosovitskiy et al. has recently emerged as a promising area of research in image classification. ViT processes an image as a sequence of patches so that the model can learn distant image region relationships using self-attention mechanisms. Global context modeling is especially helpful in identifying subtle and diffuse disease symptoms.

Thakur et al. introduced a CNN-ViT hybrid model that scored 98.61% accuracy on the PlantVillage dataset. Thai et al. demonstrated that ViTs were capable of outperforming well-known CNN models in detecting cassava leaf disease, where F1 scores varied between 75 and 96 depending on the type of disease.

Rizvee et al. modified AlexNet for detection of mango leaf diseases by decreasing channel depth for lightweight deployment with preserved 99% accuracy on the MangoLeafBD dataset. This work represents an increased demand for mobile-friendly, lightweight AI models in agriculture.

2.4 Comparison of Techniques

Table 2.1: Comparison of research papers

Paper Title	Methodology	Drawbacks
Mustafa Merchant et al.- "Mango Leaf Deficiency Detection Using Digital Image Processing and Machine Learning"	K-means clustering on RGB values to segment leaves and detect nutrient deficiencies.	Basic unsupervised method; not ideal for complex symptoms or multiple diseases.
Md. Rasel Mia et al. - "Mango Leaf Disease Recognition Using Neural Network and Support Vector Machine"	K-means for clustering, then SVM + ANN using 13 features from diseased regions.	Accuracy ~80%; limited feature learning and poor generalization.
Selvaraj Arivazhagan et al.- "Mango Leaf Diseases Identification Using	CNN used to identify five types of mango leaf diseases	CNNs lack global context modeling; accuracy limited by data quality.

Convolutional Neural Network"		
Sharada P. Mohanty et al.- "Using Deep Learning for Image-Based Plant Disease Detection"	Transfer learning with AlexNet and GoogLeNet; dataset of 54,306 images.	High model size; transfer learning more effective than training from scratch.
Md. Abdullahil Baki Bhuiyan et al. - "BananaSqueezeNet: A Lightweight CNN for Banana Disease Detection"	Used Bayesian optimization to fine-tune CNNs for banana leaf classification.	Not tailored for mango; lower performance on complex plant species.
Poornima Singh Thakur et al - "Vision Transformer for Plant Disease Detection: PlantViT"	Hybrid CNN-ViT model for diverse plant disease classification.	Reduced performance across different datasets; complex model tuning.
Huy-Tan Thai et al. - "Artificial Cognition for Early Leaf Disease Detection Using Vision Transformers"	Pure ViT model for cassava leaf disease classification (F1 scores 75–96).	High computation cost; improvement over CNNs is marginal in some cases.
H. Durmus et al. - "Disease Detection on the Leaves of Tomato Plants Using Deep Learning"	AlexNet and SqueezeNet used on tomato leaf dataset from PlantVillage.	Tomato-specific; SqueezeNet better for deployment but not for mango.

CHAPTER-3

RESEARCH GAPS OF EXISTING METHODS

Within the context of agricultural technology, disease detection in plants has been an important research area, especially for valuable crops such as mangoes. Disease detection in mango leaves at an early stage, accurately and promptly, can aid in early intervention, minimize losses, and enhance the quality of yield. With time, a variety of methods have been constructed employing traditional image processing, machine learning (ML), and deep learning-based methodologies—particularly Convolutional Neural Networks (CNNs). Nevertheless, various limitations and research gaps still exist in practical real-world implementation and scaling of such models in real farming environments.

1. Restricted Generalization across Different Conditions

Most current models are learned on datasets gathered under controlled or semi-controlled settings, with consistent lighting, backgrounds, and leaf orientations. Consequently, such models tend not to generalize very well when applied in real-world settings, where aspects like varying illumination, occlusion, background noise, and various stages of leaf development can have a huge effect on performance.

2. Dataset Limitations

There is an observable scarcity of big, publically available, and diverse mango leaf disease detection datasets. Several experiments rely on small, private datasets with restricted classes of diseases and numbers of samples, thereby minimizing the reliability of the models. Furthermore, imbalance in the classes of the dataset (i.e., under-representation or over-representation of some diseases) tends to induce biased prediction results.

3. Overfitting and Lack of Model Robustness

As a result of poor data diversity, CNN-based models tend to overfit and the model will then perform well on the training set but not as well on new data. This becomes worse when models are trained on datasets with no environmental variability or with poor data augmentation techniques.

4. Poor Interpretability

Most of the latest models are so-called "black boxes," presenting no insight at all into their decision-making rationales. Being uninterpretable is a key disadvantage, particularly in agriculture, where stakeholders including farmers and agronomists need to be trusted and comprehend how decisions are reached.

5. High Computational Needs

Deep learning models, especially CNNs, tend to demand high computational resources for training and prediction. This creates a hindrance to deployment in low-resource environments, like small farms or rural areas, where access to high-end computing infrastructure is minimal.

6. Inability to Deploy Real-Time, On-Field

The majority of studies have concentrated on offline processing with pre-captured images. There is a need to develop light, real-time applications deployable straight into the field, e.g., through smartphones or edge devices. This restricts the applicability of these technologies in real-time situations.

7. Poor Cross-Crop or Cross-Disease Transfer Learning

Most models are created specifically for mango leaves and cannot be easily mapped to detect diseases in other crops. Likewise, detection of new or emerging diseases is still a challenge because there is no flexible architecture that allows transfer learning or continual learning.

8. Inefficient Use of Transformer-Based Architectures

Although CNNs are the current trend, more recent architectures like Vision Transformers (ViTs) have also performed well in general computer vision tasks. Yet, their potential is still unexplored in plant disease detection, particularly in capturing long-range dependencies and contextual features on the leaf surface.

3.1 Image Quality and Environmental Sensitivity

Image quality is a critical aspect in the efficacy of deep learning models for detecting mango leaf disease. In contrast to laboratory conditions, field conditions present a diverse set of visual inconsistencies and noise, which have the potential to severely compromise model accuracy and reliability. The fact that most current models are sensitive to these changes indicates a critical weakness in their robustness and generalization capabilities.

3.1.1. Low-Light and Overexposure Challenges

Images taken in the field are usually photographed under natural lighting, which is highly variable across the day and weather conditions. Tree canopy or nearby building shadows, and overbright sunlight, commonly lead to:

- Poor contrast, which makes disease symptoms like spots or color change less visible.
- Color distortion, which interferes with the model's capability to differentiate healthy from infected leaf tissue.
- Partial lighting, when part of the leaf is highly illuminated, leading to inconsistent feature

extraction.

Such conditions lower the reliability of the learned features, resulting in inconsistent predictions or misclassifications.

3.1.2.Motion Blur and Focus Problems

Photos taken in the field are often plagued by:

- Camera shake due to handheld devices or wind-induced leaf motion.
- Incorrect autofocus, particularly with mobile cameras or low-end hardware.

These distortions bring in blurriness, and it becomes challenging for models to identify faint disease indicators like tiny lesions or initial-stage fungal infections. Most CNNs and even ViTs are not naturally resilient to such distortions unless specially trained with similarly distorted images.

3.1.3.Background Noise and Non-Uniform Scenes

Mango leaves in field settings are seldom isolated. The background usually has:

- Soil, sky, branches, or other leaves that bring in irrelevant visual patterns.
- Color similarities among leaf and background objects, which mislead the model in segmentation and classification.
- Overlapping leaves or occlusions that cover disease-affected areas.

Unless good preprocessing or segmentation methods are applied, this type of noise may overwhelm or mislead feature extraction layers and lead to poor performance.

3.1.4.Limitations of Current Preprocessing Methods

To counteract these issues, standard techniques consist of:

- Image augmentation (e.g., brightness changes, rotation, zoom).
- Filtering techniques (e.g., reduction of Gaussian blur, histogram equalization).

Though useful for training, such approaches are generally static and rule-based and thus cannot be adapted to dynamic environmental conditions. Moreover, they do not consider context-specific degradation of images, like confusing disease patches with shadowed areas.

3.1.5.Need for Adaptive and Context-Aware Solutions

To overcome the sensitivity of models to variability in real-world images, future work ought to be centered on:

- Dynamic adaptive image preprocessing that adjusts dynamically according to scene context, e.g., learning-based image enhancement models that optimize image contrast and clarity in real time.
- Self-supervised or contrastive learning methods that enable models to concentrate on disease-specific features and disregard irrelevant background features.
- Domain adaptation methods to enhance robustness across various lighting and environmental domains without the need for large amounts of labeled data.

The combination of these methods can greatly enhance model performance in real-world deployment situations, particularly under mobile-based or edge computing solutions in the field.

3.2 Computational Efficiency and Resource Limitations

Although deep learning has transformed image-based plant disease diagnosis, the computational cost of most state-of-the-art models is still a major bottleneck—particularly in agricultural environments where real-time on-site diagnosis is essential. Models using architectures such as ResNet, DenseNet, or even more recent transformer variants tend to favor accuracy over speed, efficiency, and deployability. This compromise limits their application in real-world applications, especially in low-resource environments.

3.2.1 Model Complexity and Overhead

Most of the high-performing models are built upon extremely deep and wide architectures:

- ResNet-50 has around 25 million parameters, whereas DenseNet-201 has well over 20 million.
- Such models demand a lot of RAM, GPU capability, and processing power, which are generally not found on edge devices such as smartphones or agricultural drones.

The computational burden not only impacts training but also results in extended inference times, making it unrealistic for real-time field diagnostics.

3.2.2 Latency and Real-Time Limitations

Time-critical agri-applications—like disease intervention at early stages or computerized spraying—require immediate or near-immediate feedback. However:

- High inference latency (of hundreds of milliseconds to a few seconds) causes decision-making to be delayed.
- Such latency tends to be intolerable in systems that must observe and respond quickly

(e.g., through drones or robotic manipulators) in order to arrest disease spread.

Delays make the operational utility of AI-facilitated agri-appliances low in dynamic, high-speed settings.

3.2.3 Power and Energy Limitations

Equipment utilized in agricultural applications tends to run on restricted battery power and unstable internet connectivity:

- Mobile phones, IoT nodes, and embedded systems find it difficult to execute complicated models all the time.
- Energy usage involved in recurrent image processing and inference translates into reduced device uptime, influencing usability and scalability in remote areas.

This is especially challenging for continuous observation systems or extensive deployments over several farms.

3.2.4 Toward Efficient and Deployable Solutions

To fill this gap between performance and feasibility, some optimization techniques have been suggested and are gaining traction in deep learning for edge computing:

1. Model Pruning

- Model pruning reduces model size by eliminating redundant or less important weights and neurons.
- This creates leaner models with quicker inference and reduced memory usage, typically with little accuracy loss.

2. Quantization

- Quantization translates high-precision (e.g., 32-bit float) weights into lower precision forms (e.g., 8-bit integers).
- This results in dramatic model size and power consumption reductions, ideal for devices that lack GPU acceleration.

3. Knowledge Distillation

- A high-accurate "teacher" model is used to train a smaller "student" model that imitates its predictions.
- The student model performs similarly but at much less computational cost, which allows for deployment on low-resource devices.

4. Lightweight Architectures

- Purpose-designed models such as MobileNet, EfficientNet-Lite, or TinyViT are optimized for mobile and edge usage, providing an optimal trade-off between speed and accuracy.
- Such architectures are also optimized for low latency and low power, suitable for field deployment in real-world applications.

5. On-Device and Federated Learning

- Shifting computation to the edge via on-device learning or federated learning lowers data transmission cost and enables models to learn locally.
- This improves privacy, decreases latency, and provides constant learning without requiring constant internet connectivity.

3.3 Accuracy under Data Variability

Accuracy and reliability of mango leaf disease detection models largely rely on the representativeness, diversity, and quality of training data. Yet realistic agricultural situations encompass a large amount of natural variability—biological as well as environmental—which adds a great deal of complexity. Most classifiers, which have been trained using small homogeneous data sets, fail to perform under these varied circumstances with overfitting and poor generalization.

3.3.1 Differences between Mango Varieties

Hundreds of mango varieties exist in the world, with varying:

- Leaf textures: Either smoother, coarser, or veined.
- Color hues: Healthy leaves, too, are not uniform and might have various shades of green or occur as natural pigmentations.
- Forms and sizes: The shape of leaves, including size and edge bend, also differs from variety to variety, resembling some disease-like symptoms.

These biological variations inherent to the plant can mislead models that have been trained on a narrow range of mango varieties and result in false positives or false negatives when distinguishing between disease.

3.3.2 Variability in Disease Stage

Diseases tend to manifest differently at different stages of progression:

- Symptoms in the early stages could be extremely mild color changes, minute spots, or mild curling—imperceptible in images.
- Mid to late-stage symptoms become more characteristic, yet can mimic symptoms of other

illnesses or environmental injury (e.g., sunburn, nutrient deficiencies).

This visual inconsistency between stages makes feature extraction difficult and also causes intra-class variability (widely varying within the same disease class), complicating classification boundaries.

3.3.3 Image Capture Diversity

Images are captured by various users on varying devices, under diverse conditions, in real-world applications. This results in:

- Inconsistent orientation and framing: Full leaf, partial leaf, or even overlapping leaves are presented in some images.
- Different zoom levels: Close-up disease spots versus full-leaf context can significantly change the visible characteristics.
- Lighting and shadow artifacts: Natural light variations, reflections, or partial shading can hide disease patterns or add misleading information.

These factors cause noise and variation within the same class, it becomes more difficult for models to learn invariant features unless they're trained on datasets that simulate such real-world variation.

3.3.4 Impact of Limited and Homogeneous Datasets

Most of the publicly available or proprietary data used in research is plagued with:

- Limited size: Hundred or a few thousand images per class is not adequate for deep learning models to generalize.
- Controlled conditions: Usually harvested in optimal lighting with clean backgrounds, which does not reflect conditions in the field.
- Limited species and disease range: Tend to target a small number of cultivars and diseases, excluding edge cases or mixed infections.

Consequently, models trained on these datasets are prone to overfitting—learning patterns very specific to the training data but meaningless or deceptive in real-world applications.

3.3.5 Path Forward: Toward Greater Robustness

To enhance model resilience and practicality in real-world scenarios, the following is urgently needed:

1. More and Diverse Datasets

- Datasets that comprise multiple mango types, diverse stages of disease, and natural image variability.
- Images from various geographical locations to cover differences in the environment and

ecology.

2. Synthetic Data Augmentation

- Applying generative models (e.g., GANs) or simulation-based methods to generate realistic synthetic images reflecting underrepresented conditions.
- Augmenting training sets with carefully crafted variations of lighting, orientation, and disease progression for better generalization.

3. Stratified and Balanced Dataset Creation

- Balanced class distributions to avoid bias towards overrepresented disease classes.
- Healthy leaves under stressful conditions (e.g., drought, nutrient deficiency) to assist the model in distinguishing disease from non-disease damage.

4. Sophisticated Learning Algorithms

- Few-shot or zero-shot learning schemes to enable models to identify diseases with a limited number of examples.
- Domain adaptation to enable models trained on a given dataset to generalize to another dataset with disparate characteristics.

3.4 Generalization to Real-World Deployments

While many mango leaf disease detection models have shown to be highly accurate under controlled laboratory environments, their performance tends to significantly decline when implemented in real-world field scenarios. This disparity between laboratory achievement and field relevance is one of the most essential challenges confronting modern agricultural AI research. Generalizability—the ability of the model to sustain performance in various, unseen, and dynamic environments—is paramount for trusted deployment but often goes unattended throughout model training and testing.

3.4.1 Overfitting on Lab-Curated Datasets

Most deep learning models are trained and validated on:

- Carefully curated images with clean backgrounds, uniform lighting, and centered leaves.
- Datasets gathered under controlled lab conditions or with consistent camera setups.

This results in overfitting, where the model learns to identify patterns unique to the training dataset instead of strong features that are invariant across real-world conditions. Consequently, these models tend to:

- Perform outstandingly in validation or test phases in the same dataset.

- Show drastic reductions in accuracy when confronted with raw field images that are of varying quality, angle, or context.

3.4.2 Cross-Dataset and Cross-Region Poor Performance

One of the biggest generalization challenges is cross-dataset validation—testing a model that was trained on one dataset with another dataset of differing characteristics. In mango leaf disease detection, models often struggle to generalize because:

- Regional variation in disease expression due to climate, soil, or mango variety.
- Image capture device variations (smartphone vs. DSLR, resolution variations).
- Annotation variations, e.g., inconsistent stage labeling of disease or blended symptoms.

This undermines the reliability of the model and renders it unfit for deployment outside its training setting.

3.4.3 Poor Evaluation Metrics

Most research papers only report overall accuracy, which is an incomplete measure of model performance, particularly when:

- There exists class imbalance (e.g., significantly fewer pictures of early disease stages or less common infections).
- False negatives (i.e., false alarms as diseased leaves) are highly important to not occur, because they might propagate uncontrolled outbreaks of disease.

Ignoring metrics such as:

- Recall (sensitivity): Vital for measuring how well the model can identify all actual cases of disease.
- Precision: Significant to reduce false alarms that might bring about unnecessary interventions.
- F1-score: Harmonic mean for precision and recall.

Can produce deceptive conclusions regarding a model's readiness for real-world deployment.

3.4.4 Bridging the Generalization Gap

In order to have models trained in research environments behave predictably in agricultural environments, the following techniques are crucial:

1. Cross-Validation and Cross-Dataset Testing

- Perform k-fold cross-validation over multiple data subsets with varying features and classes to guarantee the model isn't biased to particular features or classes.
- Apply benchmark datasets with varying regions and conditions to verify the model's resilience across domains.

2. Transfer Learning and Domain Adaptation

- Leverage transfer learning from large, heterogeneous plant datasets to boost performance on scarce mango-specific data.
- Employ domain adaptation methods to adapt fine-tuned models with minimal data from the target deployment territory to enable the model to adapt to local changes.

3. Real-World Simulation during Training

- Add real-world noise to the training pipeline (e.g., blur, shadows, and busy backgrounds).
- Simulate on-device aspects like compression artifacts or low-resolution images.
- Train on mixed datasets containing both curated and field-acquired images.

4. Deployment-Focused Evaluation Frameworks

- Assess models not only on validation sets, but on field-acquired test sets in real-world conditions.
- Utilize confusion matrices, precision-recall curves, and F1-scores to evaluate performance more comprehensively.
- Add false negative rates as a fundamental evaluation metric, especially in the detection of early-stage disease.

5. Continuous Monitoring and Feedback Loops

- Implement with a feedback system wherein wrongly predicted images are identified, audited, and employed for incremental learning or retraining.
- Enable models to improve after deployment by feeding them real-world instances over time.

3.5 Lack of Real-Time Capability and Accessibility

Even with increased use of AI-based plant disease detection, the majority of systems are still limited to experimental or academic environments. This is partly because there is not enough attention to real-time operation and user usability, particularly in the case of smallholder farmers and field workers in distant or low-resource locations. To have a concrete effect on agriculture, AI needs to be quick, easy to use, and accessible in the field—and not only in the lab.

3.5.1 No Offline or Mobile Support

Most disease detection systems:

- Are made for deployment on high-performance computers or cloud computing,

necessitating steady internet connection.

- Do not have specific mobile apps that can run well on ordinary smartphones or low-end hardware.
- Provide no or limited offline capability, which is essential in rural agricultural areas where internet access is spotty or non-existent.

Without offline availability or effective mobile rollout, these systems are of no use to end users when they most require them.

3.5.2 Clunky and Complicated User Interfaces

Even where tools are technologically usable, they tend to suffer from:

- Incredibly complicated dashboards, written for researchers or data scientists.
- Subpar localization: support for neither regional languages nor cultural environments.
- Unintuitive processes: forcing farmers to go through technical steps like uploading images, choosing model options, or analyzing raw probability scores.

This translates into low rates of adoption among farmers, who do not necessarily have the time, training, or inclination to use very technical tools.

3.5.3 Slow Feedback Due to Server-Side Processing

Most existing solutions depend on:

- Posting images to cloud servers, where the AI model works on the data and responds.
- This brings in network latency, which may be several seconds to minutes based on connectivity.
- In live agricultural situations—like disease scouting during a farm visit or using targeted treatment—feedback delays minimize the utility of the diagnosis.

Farmers require immediate feedback, preferably in the field, not hours later when back in a connected setting.

3.5.4 Moving Toward Real-Time, User-Friendly Systems

Closing the gap between technical potential and actual-world use calls for a change in emphasis from pure AI strength to human-focused design and edge deployment. Following are the strategies to fill this gap:

1. Mobile-First, Edge-Optimized Deployment

- Employ minimal AI models (e.g., MobileNet, EfficientNet-Lite, and TinyViT) that can be executed directly on smartphones or on portable edge devices such as Raspberry Pi or Arduino with camera modules.
- Provide offline capability, so users can analyze images without having to upload data—

crucial for both usability and data privacy.

2. Intuitive, Farmer-Centric User Interfaces

- Create easy-to-use mobile apps with big buttons, legible icons, and limited text to serve users with less technical expertise.
- Add voice commands, text-to-speech, or multilingual functionality, particularly for local languages and literacy-constrained users.
- Offer visual feedback (e.g., red/yellow/green lights, heatmaps on infected regions) rather than technical metrics such as confidence scores.

3. Real-Time Feedback and Actionability

- Deliver disease detection results within a few seconds, from preprocessing through inference to result display.
- Provide instant action recommendations (e.g., "Isolate plant", "Use fungicide", "Retest in 3 days") based on model output and severity.
- Offer batch scanning or drone-based scouting for scanning large fields rapidly.

4. Synthesis with Local Farming Knowledge

- Combine AI diagnosis with local extension input (e.g., local best practices, guidelines on chemical usage).
- Provide functionality for users to upload or report cases for distant consultation and input to ongoing model improvement through crowdsourced feedback.

5. Low-Cost and Sustainable Access

- Open-source or freemium strategies to be cost-effective.
- Utilize SMS-based notification or USSD menu for non-smartphone-using users.
- Align with agricultural cooperatives, NGOs, or government initiatives for channeling and training.

CHAPTER-4

OBJECTIVES

The increasing demand for sustainable agriculture and timely disease management in tropical fruits such as mango has compelled researchers to seek innovative digital solutions. Among them, the application of artificial intelligence (AI) and deep learning has emerged as a breakthrough in terms of plant disease detection automation. This project is centered on utilizing the strength of Vision Transformers (ViT) to create a strong, smart system that can precisely diagnose different diseases in mango leaves from plain images. Through the capability of ViT to process images through global attention mechanisms, the system seeks to surpass the confines of conventional image processing and CNN-based methods.

The main objective of this study is to develop a disease detection model that not only performs well but is also deployable in real-world settings. Conventional machine learning approaches are highly feature-dependent and are likely to perform poorly in uncontrolled settings. Even deep convolutional neural networks, although highly capable, suffer from limitations in modeling long-range dependencies in images and can be challenged by subtle symptoms or complicated leaf backgrounds. ViT, however, transforms the image into small patches and examines their interdependencies through self-attention, allowing it to pay attention to both local and global disease markers in the leaf image. This study aims to develop such a ViT-based architecture, specific to mango leaf disease detection, and train it on a diverse dataset of diseased and healthy leaf images.

Another major goal that we are working towards is making the system explainable and comprehensible to every user. For the case of agriculture, end-users who depend on AI solutions are primarily farmers or field engineers who lack the technical know-how to be able to comprehend complex algorithms. It is therefore all the more important that the model's decisions are interpretable and explainable. With the addition of top-of-the-line explainable AI methods, including attention heatmaps or visualizations using gradients, users will be able to view which particular region of the leaf the model was concentrating on when it made its prediction. Such explanation transparency does not only contribute to trust in the system but is also an effective learning tool for farmers, allowing them to learn more about disease patterns and their implications. Additional effort will also go towards compressing and optimizing the model so that it is light enough to run well on smartphones or low-power edge hardware. Such functionality will enable offline use in rural and far-flung agricultural areas,

again contributing to accessibility for such users.

While there has been some success and some promise demonstrated by traditional approaches, as well as some CNN-based models, they do not work when they are exposed to less-than-ideal real-world scenarios. These can be things like low lighting, distracting background noises, or changes in the leaves themselves, which can slow them down. That is exactly where Vision Transformers are especially useful — they can look at the larger picture and efficiently identify important patterns in the leaf images with much greater efficiency.

Lastly, the project also hopes to make a contribution to the wider research community and agrifood system. Either by complementing current data sets, providing open-source software, or enabling new methods of real-time plant disease detection, the hope is to leave a lasting contribution. The envisioned system is to be not only a diagnostic but also a system that inspires further innovation in smart farming. Potentially able to be integrated with drones, IoT-based systems, or mobile advisory services, this ViT-based solution can be a tipping point in the precision agriculture of the future.

There are a few significant goals within this project that we would like to achieve and attain:

4.1 Develop a Vision Transformer-Based Model for Disease Detection

The objective is to create a high-end deep learning model that takes advantage of the new architecture of Vision Transformers, optimized to accurately detect and classify various mango leaf diseases. The high-end model will be extensively trained to learn important patterns in leaf images so that it can be effective and accurate irrespective of variations in factors such as lighting, image orientation, or the various severity levels of the diseases that the leaves may be afflicted with.

4.2 Gather and Enrich a Varied and Diverse Dataset

The objective is to have a big and high-quality dataset of mango leaf images that is going to include healthy and diseased leaves. The image set will be created to simulate the real world, and these should possess the following features:

• Various types of diseases

•Two or more different angles and several different lighting conditions.

• Leaves with background clutter or overlapping symptoms

We will also use image augmentation methods to train the model better by artificially increasing the dataset.

4.3 Make the Model Transparent and Easily Understandable through

AI models are typically mysterious black boxes whose inner workings are not easily comprehensible. The overall aim of this project is to improve the model's explainability of decision, especially to end-users like farmers and agronomists who will be using this technology in their daily operations. In order to do this, we will make use of several tools, such as attention heatmaps, which will graphically highlight the exact regions of the leaf image the model focused on during its processing. This will allow users to understand the reasoning behind the model's predictions, thus gaining a better understanding of how it came to its conclusions.

4.4 Perform a Thorough Analysis and Compare the Outputs of Different Models

We will rigorously test the model based on performance measures such as accuracy, precision, recall, and F1-score. We will also compare our ViT model's performance with other models such as ResNet or MobileNet to determine whether it actually provides an improvement or not.

4.5 Refine the Model to Suit Mobile or Field Usage

In order to ensure that the model is precise and accurate as well as functional and usable, we are committed to maintaining our attention on the steps that will allow us to minimize its size as a whole while simultaneously maximizing its speed and efficiency. With these modifications, we will make it possible for the model to operate effectively on low-cost devices, such as smartphones or drones, which will allow farmers to use it directly in the field, even where they lack an internet connection.

4.6 Create an Intuitive, Simple-to-Use Interface

As part of our efforts to make the system user-friendly and accessible, we have come up with a proposal to design either a mobile app or an online web app that will allow users to:

• Upload or take a photo of a leaf

• Obtain an early diagnosis without wasting time.

• See a graphical representation of the prediction

•Obtain and consider various recommendations for appropriate treatment options.

4.7 Contribute to Agricultural AI

In brief, we hope that this project will help bring meaningful and valuable contributions to the continually changing arena of artificial intelligence in agriculture. This could be done in many ways, perhaps through the release of our curated dataset to the wider community, through the release of our model code for others to utilize and build upon, or even through simply encouraging an increase in research effort aimed towards the valuable domain of mango disease detection. Our final goal is to be able to be able to make a positive and useful contribution to the manner in which crop diseases are detected and dealt with by farmers and researchers, specifically through the innovative application of technology that can aid their abilities.

CHAPTER-5

PROPOSED METHODOLOGY8

In agriculture, the well-being of crops is an important factor for determining produce quantity and quality. Mango, as the "king of fruits," is of crucial economic and cultural value in numerous tropical and subtropical countries. Mango cultivation has many challenges to overcome, chief among them leaf diseases. These diseases not only decrease the plant's photosynthetic capability but also act as markers for probable fruit infection, resulting in considerable economic loss. Conventional disease detection through reliance on human observation by specialists is time-consuming, subjective, and generally unavailable to small farmers. With the onset of recent technology advancements, the need is immediate to engineer automatic, accurate, and scalable disease detection in early stages.

The purpose of this research is to bridge the existing gap between conventional farming practices and contemporary technological advances by presenting a deep learning-driven framework for mango leaf disease identification. Taking advantage of the strengths of the Vision Transformer (ViT) model, which has been renowned for being highly effective in capturing global context cues in images, the proposed approach presents an overarching scheme for detecting and classifying diverse mango leaf diseases. The following sections discuss every stage of the methodology, ranging from data processing and cleaning to model deployment, so that a complete understanding of how the system was developed and implemented is maintained.

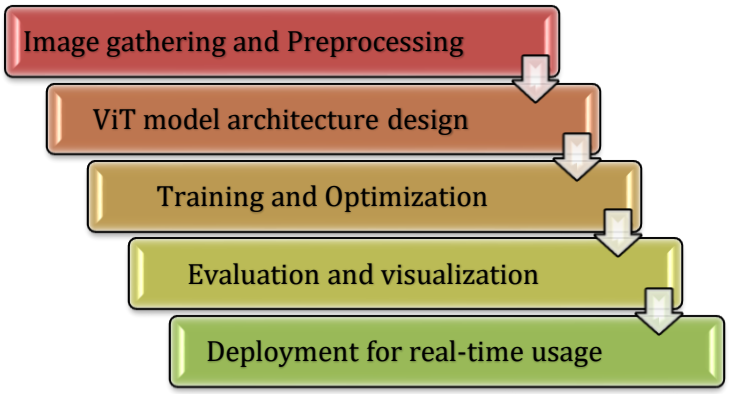


Figure 5.1 : Proposed method architecture diagram

5.1. Overview of Method

The system to be proposed is carefully designed to provide real-time, accurate disease classification of mango leaves, with a focus on simplicity and strength. The approach involves

a sequence of related steps, each of which plays an important role in the overall performance of the system. The main steps are:

1. Image Gathering and Preprocessing: Collecting a Diverse Collection of Mango Leaf Images and Preparing Them for Training

The image collection and preprocessing process is the backbone of any deep learning model, and its value cannot be emphasized enough when it comes to mango leaf disease detection. For a model to work effectively, it needs to be trained on a diverse and representative dataset that includes all the conditions on which the model is going to run in real-life applications.

In this work, a dataset of 3,846 mango leaf images was collected, with both healthy and diseased leaves included. The dataset represented various mango leaf varieties, including Alphonso (Hapus), Totapuri, Banganapalli, Dasheri, Kesar, Langra, Neelum, and Mallika. Each variety has unique visual features, and the diseases appear differently across varieties, necessitating the model to generalize over this variety of conditions.

The images in this dataset were taken under diverse environmental conditions, including varying lighting, angles, and background noise, to make sure that the model can deal with these variations. This is particularly crucial in real-world agricultural environments, where lighting and the presence of other vegetation or objects can affect image quality. The diseases covered in the dataset varied from prevalent fungal and bacterial diseases, including anthracnose, powdery mildew, and bacterial leaf spots.

After the images were gathered, the preprocessing operations were performed to get them ready for training. The images were resized into a fixed size of 224×224 pixels to make the model's input uniform since the Vision Transformer (ViT) takes a fixed-size input. All the images were also normalized into the range [0, 1] by dividing pixel intensities by 255 to make data easier to handle for the model. Preprocessing also included enriching the dataset to artificially enlarge its size and diversity, preventing overfitting and ensuring generalization of the model. Some of the techniques used for data augmentation include random rotations, horizontal and vertical flips, brightness and contrast adjustments, and adding Gaussian noise. These changes enabled the model to learn a generalized representation of the leaf diseases, thus enhancing its capability to identify disease under diverse conditions.

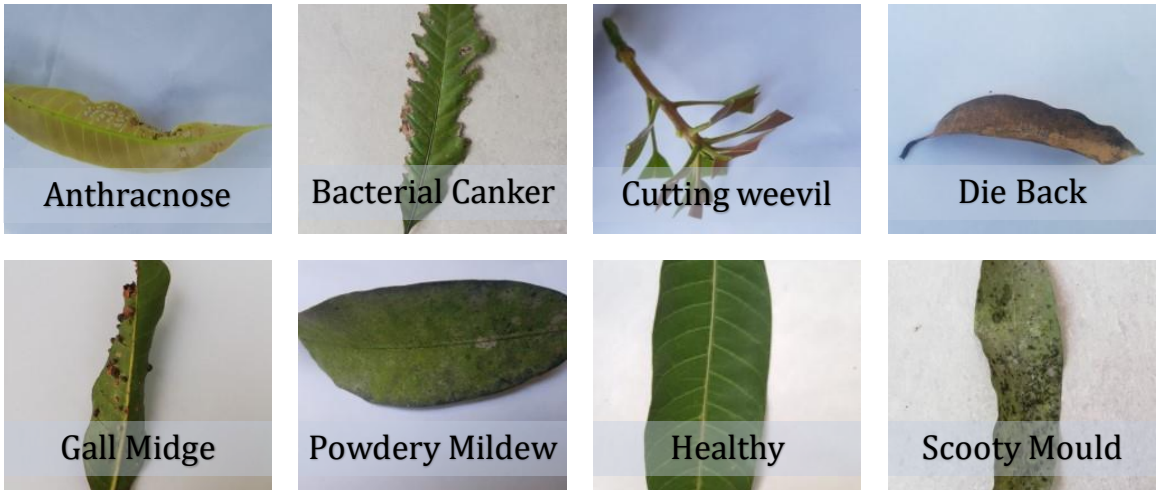


Figure 5.2 : Different types of diseased leaf images & healthy leaf

2. ViT Model Architecture Design: Designing a Model Architecture Specific to the Novelty of Leaf Disease Detection Challenges

Vision Transformer (ViT) architecture is a new departure from the conventional Convolutional Neural Networks (CNNs), which have been ruling the landscape of image recognition. The premise on which ViT is based is to handle patches of images in the same manner as words in natural language processing (NLP) tasks are handled. This design choice is very beneficial when handling images in which long-range dependencies, for instance, relationships between far-apart areas of an image, play a key role in grasping the overall scene a fundamental necessity in leaf disease detection.

In this approach, the model was implemented employing the ViT, which splits the image into tiny, non-overlapping patches (for example, 16×16 pixels) and processes them one by one through a self-attention mechanism. This enables the model to learn both global and local dependencies among various parts of the image. For instance, in an anthracnose-infected mango leaf, a small region close to the center of the leaf could have a different pattern than one at the periphery, but the general disease pattern is decipherable only if these regions are viewed relative to one another.

Important Features of the ViT Architecture:

- Patch Embedding: Each input image is split into small patches (16×16 pixels), which are flattened into vectors. These vectors are linearly passed through a layer to produce the patch embeddings that are used as input tokens for the transformer.

$$zp = L * x_p + b \quad (1)$$

where:

- L is the learned projection matrix.
- b is the bias.

- Positional Encoding: As the ViT architecture does not inherently understand the spatial relationships between patches, positional encoding is added to the embeddings. This encoding enables the model to preserve the spatial layout of the patches in the original image.

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{10000^{2i/d}}\right) \quad (2)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{10000^{2i/d}}\right) \quad (3)$$

where pos is the patch index and d is the embedding dimension.

- Multi-Head Self-Attention (MHSA): The foundation of ViT's capacity for learning local as well as global features rests on the self-attention mechanism. MHSA allows the model to compute the attention between each pair of patches, meaning the model can determine which patches are most related to each other irrespective of their locations in the image. For instance, it may learn that a patch with discoloration in one corner is connected to another patch with similar discoloration elsewhere in the image. The attention mechanism computes attention scores using Query (Q), Key (K), and Value (V) matrices:

$$B = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (4)$$

where:

- $Q = XW_q, K = XW_k, V = XW_v$
- W_q, W_k, W_v are learnable weights,
- d_k is the dimension of the key.

- Feed-Forward Network (FFN): The outputs are next passed through a feed-forward neural network that imposes a set of transformations in order to enhance the expressiveness of the model as well as the model's capability to learn non-linear relationships. Each transformer block contains a feed-forward network (FFN) that processes the output of self-attention:

$$FFN(X) = ReLU(XW_1 + b_1)W_2 + b_2 \quad (5)$$

where:

- W_1, W_2 are weight matrices,
- b_1, b_2 are biases

- Classification Head: At the end of the transformer layers, a special [CLS] token (representing the whole image) is passed to a final linear layer followed by a softmax function to give the class label that in this example is a healthy or one particular disease class.

$$y = softmax(W_{ds} * x_{ds} + b_{ds}) \quad (6)$$

where:

- x_{ds} is the embedding of a special classification token [CLS],
- $W_{ds}\ and\ b_{ds}$ are learned parameters.

- By utilizing this architecture, the ViT model enables the system to be more effective in detecting the fine-grained patterns on mango leaves, whether subtle color variations brought about by disease or more involved spatial patterns which mark advanced stages of infection.

3. Training and Optimization: Using Effective Training Techniques to Tune the Model for Its Best Performance

Having designed the model architecture, the second very important step was the training phase, which entails passing the prepared data through the model and tuning the model parameters according to the prediction mistakes. We used a strong training process for this purpose to achieve the best performance of the model.

- Data Split: The data was split into three sets: 80% for training, 10% for validation, and 10% for testing. The training set is utilized to train the model, the validation set is utilized to fine-tune hyperparameters and identify overfitting, and the test set is utilized to get an unbiased estimate of the model's performance.
- Hyperparameter Tuning: During the training, some of the hyperparameters were tuned with care to maximize the performance of the model. The learning rate was adjusted to a low value of 0.0001 so that the model improved gradually but did not overshoot the optimal solution. We employed the Adam optimizer with weight decay to counteract the model's capacity to learn rapidly and prevent overfitting by regularizing the weight

updates. The model was trained for 50 total epochs with a batch size of 16, which provided each mini-batch with sufficient examples to update the model weights effectively without overloading the GPU memory. Checkpointing was used during training to save the state of the model at regular intervals such that it could be resumed in case of any interruptions.

- Data Augmentation and Regularization: Data augmentation was used to prevent overfitting and to make the model more resistant to unseen data. Data augmentation was applied by rotating images, flipping them horizontally and vertically, and altering their brightness and contrast. Some images also had Gaussian noise added to them, simulating types of distortions that could occur under real-world conditions. These enhancements guaranteed that the model was not memorizing the training data but rather learning more general features that would be applicable to new, unseen images.
- Hardware and Training Environment: Training was done on a system with GPU support and CUDA enabled to leverage parallel computation so that the model could process large batches of images in parallel. The use of PyTorch 3.8 gave the flexibility to try out various model configurations and optimizations, and the capability to use GPU acceleration cut down the time taken for model training considerably.

4. Evaluation and Visualization: Measurement of the Model's Accuracy and Interpretation of Its Decision-Making. After training the model, it was important to measure its ability to identify mango leaf diseases. This process entailed measuring the model's performance statistics and visualizing its decision-making process so that the model not only performed accurately, but also allowed for interpretation.

- Performance Metrics: A number of primary performance statistics were employed to measure the model:

1. Accuracy: The ratio of correct predictions to the total number of predictions.
2. Precision: The ratio of true positives (correctly predicted diseased leaves) to the total predicted positives.
3. Recall: The ratio of true positives to the total actual positives in the dataset.
4. F1-Score: A weighted average of precision and recall, helpful in balancing the trade-off between false positives and false negatives, particularly in imbalanced datasets.

- Visualization Techniques: To further understand how the model made its decisions, various visualization techniques were employed. These included:

1. Confusion Matrix: This matrix helped visualize misclassifications, allowing us to identify which disease classes were most frequently confused with others.

2. Grad-CAM: Gradient-weighted Class Activation Mapping (Grad-CAM) was employed to create heatmaps that identified which parts of the image the model was paying attention to when classifying. The method gave insights into whether or not the model was concentrating on useful features, i.e., disease spots on the leaf, instead of non-useful ones like the background.

These visualization aids provided an even better understanding of how the model was performing, highlighting areas in which the model was performing well and areas where further fine-tuning or additional training data was needed.

5. Deployment for Real-Time Usage: Putting the Model into Practice in a Usable Application for Everyday Use

After the model performed to acceptable levels of performance, the task was to deploy it for real-time use, so that the farmers and other stakeholders could be benefited from its predictions.

- Model Export and Conversion of Format

To enable deployment on multiple platforms, the model that was trained was exported into ONNX. ONNX is very compatible across a broad scope of platforms and thus enables simpler integration of the model into production environments. ONNX provides automatic model conversion among frameworks such as PyTorch and TensorFlow, as well as deployment across a broad spectrum of platforms from web servers to handheld devices.

- FastAPI-Based Web Application

A friendly web application was created using FastAPI, a high-performance web framework that is fast and easy to use. The web application allowed users to upload images of mango leaves directly through a straightforward interface. After submission, the image was processed by the ViT model, and the disease classification output was provided in real-time.

- Real-Time Use Case:

For farmers, this tool is a handy, easy-to-use device for checking the health of their crops. By merely snapping a photo of a mango leaf on their phones, they would be able to receive instant feedback on whether the leaf is healthy or not and receive suggestions for treatment. The user-friendly interface of the application, with little technical lingo, made it easy for farmers and agricultural laborers who have little technical expertise to use.

5.2. Data Collection and Preparation

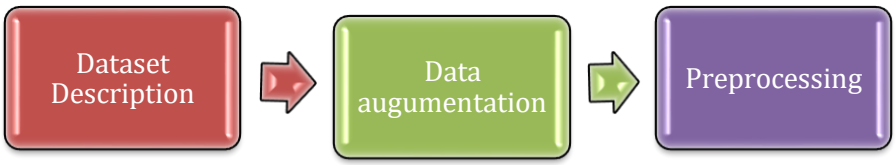


Figure 5.3 : Flowchart of data collection and preparation

1. Dataset Description

The quality and diversity of the training data are the backbone of any machine learning model. Understanding this, an extensive dataset consisting of 3,846 images of mango leaves was prepared. This dataset contains a broad range of conditions, ranging from healthy leaves to leaves suffering from prevalent diseases like anthracnose, bacterial spots, and powdery mildew. To make the model generalizable across various mango varieties, images were collected from different cultivars such as Alphonso (Hapus), Totapuri, Banganapalli, Dasheri, Kesar, Langra, Neelum, and Mallika. The variation in leaf morphology, color, and disease expression across these varieties gives the model a rich learning environment, which makes it better at identifying subtle differences and patterns.

2. Data Augmentation

In actual cases, the range of environmental conditions, illumination, and leaf orientations can be real challenges to image-based disease detection systems. To mimic these variations and enhance the robustness of the model, extensive data augmentation methods were utilized:

- Rotation: Images were rotated by different angles to simulate various leaf orientations.
- Flipping: Horizontal and vertical flips were both performed to take into consideration symmetrical variations.
- Brightness and Contrast Adjustment: Brightness and contrast adjustments mimicked various lighting conditions.
- Gaussian Noise Addition: Adding noise made the model robust against image flaws

These augmentations not only enlarged the size of the dataset but also diversified it, allowing the model to generalize well under different conditions.

3. Preprocessing

All images were subjected to a standardized preprocessing pipeline to ensure compatibility and consistency with the ViT model. The images were resized to 224×224 pixels, which is the

input requirement of the model. Pixel values were normalized to [0, 1] to allow for quicker convergence during training. Images were also converted to the RGB color space to maintain consistency throughout the dataset.

5.3. Vision Transformer (ViT) Architecture

The Vision Transformer (ViT) is a shift in image classification from the conventional convolutional neural networks (CNNs) to transformer-based models. In contrast to CNNs, which are based on local features using convolutional operations, ViT views images as sequences of patches, allowing the model to learn global contextual information.

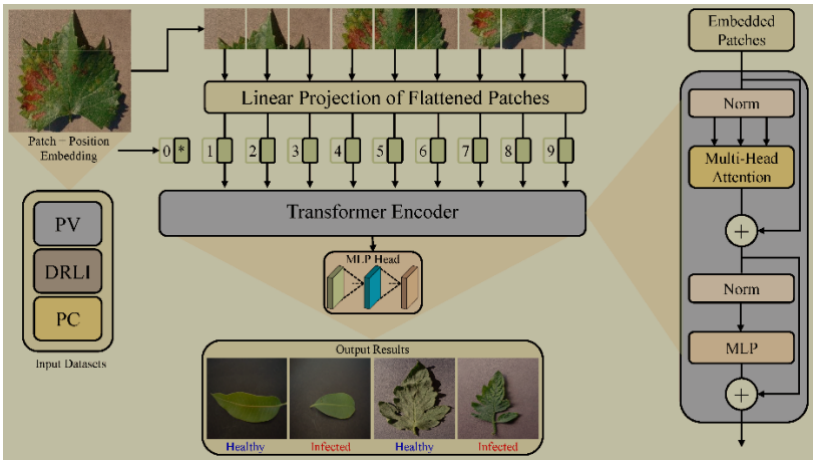


Figure 5.4 : ViT B-16 Architecture

1. Patch Embedding

The initial step in the ViT architecture involves dividing each image into fixed-size patches, typically 16×16 pixels. For a 224×224 image, this results in 196 patches. Each patch is then flattened and projected into a lower-dimensional embedding space using a linear transformation. This process transforms the 2D spatial information into a 1D sequence, analogous to word embeddings in natural language processing.

2. Positional Encoding

Transformers themselves do not have a sense of order, and therefore positional information becomes extremely important for sequence-based tasks. For this purpose, learnable positional encodings are incorporated in every patch embedding to give the model relative position information about the patches within the image. Through this incorporation, spatial relationships remain maintained across the entire processing of the model.

3. Multi-Head Self-Attention (MHSA)

The self-attention mechanism is at the core of the transformer architecture. MHSA enables the model to assign weights to the relative importance of various patches with respect to one another, detecting complex relationships and dependencies. Using multiple attention heads, the model can attend to multiple aspects of the image in parallel, making it better suited to detect complex patterns related to various diseases.

4. Feed-Forward Network (FFN)

After the layers of attention, every patch embedding is fed through a feed-forward network that has two linear layers with a non-linear activation function. This element fine-tunes the representations that have been learned in the attention stage so that the model is able to detect higher-level abstractions.

5. Classification Head

A special classification token ([CLS]) is added at the beginning of the sequence of patch embeddings. The embedding for the [CLS] token after passing through the transformer layers captures the pooled information across the whole image. The resulting embedding is fed through a linear layer followed by a softmax activation to output the final class probabilities, which signify the presence or absence of particular diseases.

5.4. Training and Optimization Strategy

A typical 80-10-10 split was done to the dataset — i.e., 80% of the total images for training the model, 10% for testing the model's learning in the course of training, and the last 10% held back strictly for final testing alone. The partitioning is mainly done to have a correct balance between the model learning and unbiased testing. By separating the test set, we make sure that the model never observes this data while training or validating, so that we can have a more realistic estimate of its generalization ability.

1. Optimization Techniques

- To make sure the Vision Transformer model not only converges effectively but also does not overfit or underfit, we utilized the Adam optimizer — a widely used option in deep learning because it has an adaptive learning rate feature. In particular, we utilized the Adam optimizer along with weight decay, which penalizes large weights and serves as a type of regularization. This makes the model prefer simpler solutions, which tend to be more generalizable.
- Further, learning rate scheduling was employed to adjust the learning rate dynamically during training. A comparatively higher learning rate was initially employed to enable the

model to take big leaps in its parameter space. During the course of training, the rate was progressively decreased to fine-tune the model with increased accuracy. Early stopping was also employed to stop training when the validation loss ceased to improve for a specified number of epochs — a precaution against overfitting and conserving computational resources.

2. Hardware Utilization

Because of the computational nature of training a ViT model, especially because of its attention mechanism and the high number of parameters, training was done on an NVIDIA GPU with CUDA support under PyTorch 3.8. Utilizing GPU acceleration not only greatly accelerated training but also enabled more extensive batch sizes and more complicated transformations, which would have been unfeasible in CPU-based environments.

3. Model Checkpointing and Logging

•Another critical component of the training pipeline was checkpointing. The state of the model (weights, optimizer state, epoch number) was saved after every epoch. This did two things: first, if there was an unexpected break (power outage, system crash, etc.), training could be picked up from the previous saved point; second, the highest-performing model — by validation accuracy measure — could be preserved even if the later training epochs caused regression. Side by side, detailed logging of training/validation loss and accuracy was kept for future visualization and performance inspection.

5.5. Evaluation Metrics and Visualization

1. Performance Metrics

- In evaluating the performance of a machine learning model, accuracy is not everything. A richer comprehension involves various measures that give a more accurate representation of the model's capabilities and shortcomings. Thus, we utilized some key performance indicators:
- Accuracy: It is the most natural measure, defined as the proportion of true positives among all images. Although helpful, it can at times be deceptive in class imbalance sets.
- Precision: Precision refers to how many of the samples the model had labeled positive (i.e., diseased) were indeed true. It is most useful when the penalty for false positives is high.
- Recall: Recall indicates the number of true positive instances (infected leaves) that were properly classified by the model. Good recall is extremely important in

medical/agricultural diagnosis to avoid missing any diseased cases.

- F1-Score: This is the harmonic mean between precision and recall, providing one measure that is balanced between the two. It's strongly suggested when working with imbalanced classes, as commonly found in leaf disease datasets.

2. Visualization Techniques

Visualizing the learning process of the model and decision-making patterns is crucial for debugging and interpretability. For this purpose, the following tools were employed:

Loss and Accuracy Curves: These plots represent training and validation loss/accuracy versus epochs. With these curves, we can readily identify overfitting, underfitting, or training instability.

- Confusion Matrix: This matrix decomposes predictions into true positives, false positives, true negatives, and false negatives, providing in-depth insight into particular misclassification patterns.
- Class-wise Heatmaps: By employing methods such as Grad-CAM (Gradient-weighted Class Activation Mapping), we created heatmaps that emphasize the areas in an image on which the model was concentrating when classifying. These visualizations serve to ensure that the model is indeed learning significant features — such as detecting diseased patches instead of extraneous background details.

CHAPTER-6

SYSTEM DESIGN & IMPLEMENTATION

In the agricultural disease detection area, particularly for economically important crops such as mango, it is essential to create an extremely accurate and real-time predicting system. Automatic detection of mango leaf diseases employing current computer vision and deep learning methods, but most importantly, the Vision Transformer (ViT) model is the main mission of this project. A strong detection system also needs a robust model but alongside that, a well-structured and integrated architectural pipeline. This chapter gives a detailed description of the entire methodology followed for this system—from data acquisition, preprocessing, and augmentation to model construction, training, testing, and final deployment in an interactive web interface. The suggested methodology ensures that every phase is deliberately designed to preserve accuracy, generalization, and applicability in actual settings, especially for farmers and farm workers in rural settings.

6.1 Overview of System Architecture

The system to be proposed has been architected in a modular and scalable manner so that it becomes simpler to create, modify, and install on different hardware platforms. The overall architecture consists of standalone yet coupled modules, each performing individual phases of the disease detection process. A diagrammatic illustration of such an architecture is presented in Figure 5.1, with data and activity flow among elements. The components are listed as follows:

- Data Preprocessing and Collection: This module involves collecting raw image data and preprocessing it for use in training the model.
- Vision Transformer (ViT) Model Setup: In this step, the ViT model is set up with the correct patch sizes, heads of attention, and depths to meet the complexity of mango leaf disease patterns.
- Model Training and Optimization: The role of this module is to optimize model weights with the training data and chosen hyperparameters to reduce loss.
- Model Evaluation and Visualization: In this step, the model is tested and validated through performance metrics and visualization tools to verify that it works correctly on new data.
- Deployment for Real-Time Application: The last piece of work deals with transforming the trained model into a deployable state and encapsulating it in a convenient interface for

real-world usage.

- This framework makes sure that the system is not just lab-proven but field-deployable and can assist farmers to identify diseases early on before they get the chance to spread.

The performance of a machine learning model relies significantly on the quality, diversity, and balance of the training dataset. In this research, we gathered a dataset of 3,846 high-quality mango leaf images. The images were obtained from a mix of publicly available plant disease datasets, academic publications, and field photos taken using smartphones. The data set contains several varieties of mangoes which are widely grown in India, including Alphonso (Hapus), Totapuri, Banganapalli (Benishan), Dasheri, Kesar, Langra, Neelum, and Mallika. Having such a variety of leaf types guarantees that the model will generalize well to other visual textures and manifestations of the disease.

Every class within the dataset contains both disease and healthy leaf samples, with anthracnose, bacterial canker, and powdery mildew serving as the main disease classes for classification. As real-world disease images tend to be noisy, imbalanced, and in short supply, data augmentation is necessary. To augment dataset size and diversity artificially, we used a number of augmentation methods. These involved rotating images by ±30° to mimic varied camera angles, horizontal and vertical flipping to provide for reflected leaf orientations, contrast adjustment to mimic varied lighting conditions, and Gaussian noise to simulate background distractions like dirt or blur.

All the images were resized to 224×224 pixels and normalized according to the pre-trained Vision Transformer's expected input format. Standardization assists the model in learning strong features and reduces the likelihood of overfitting.

6.2 System Design

Classic CNNs are very good at local feature extraction but tend to lose global contextual relationships across a whole image. This drawback becomes important in the case of complex diseases where they appear with irregular patterns, spread, and textures over the entire leaf. To address this, we borrowed the Vision Transformer (ViT) architecture that has become increasingly popular because it can process long-range dependencies and global information with attention mechanisms rather than convolutions.

- Patch Embedding

Each input image is divided into 16×16 patches, resulting in a total of 196 patches for a 224×224 image. These patches are flattened into vectors and passed through a linear

embedding layer that converts them into a fixed-dimensional feature space. This process effectively transforms the image into a sequence of tokens, analogous to words in natural language processing (NLP), enabling the use of transformer-based learning.

- Positional Encoding

Transformers don't have the capability to learn the position of tokens inherently. So, we attach positional encodings to the embedded patches so that the model not only learns the patch content but also the patches' relative positions in the image. This is very important for applications such as leaf disease detection, where the spatial spread and location of disease symptoms mostly decide the classification.

- Multi-Head Self-Attention

In the multi-head self-attention (MHSA) mechanism, the model gets trained to look at parts of the image that are most informative. Different heads process different subsets of the features independently, so the model can look at many parts of the image at once. For example, one head could look for texture, another look for color variation, and another look for edges or spots.

- Feed-Forward Network

The attention output is then fed through a feed-forward neural network of two linear layers with a non-linearity in between. This network adds further complexity and non-linearity to the features and assists the model in learning more abstract and richer representations of the symptoms of disease.

- Classification Head

Lastly, a special [CLS] token is added at the start of the input sequence. This token, after going through the transformer layers, collects information from all patches and serves them as the final representation for classification. The token goes through a softmax layer, resulting in probability distributions over disease classes.

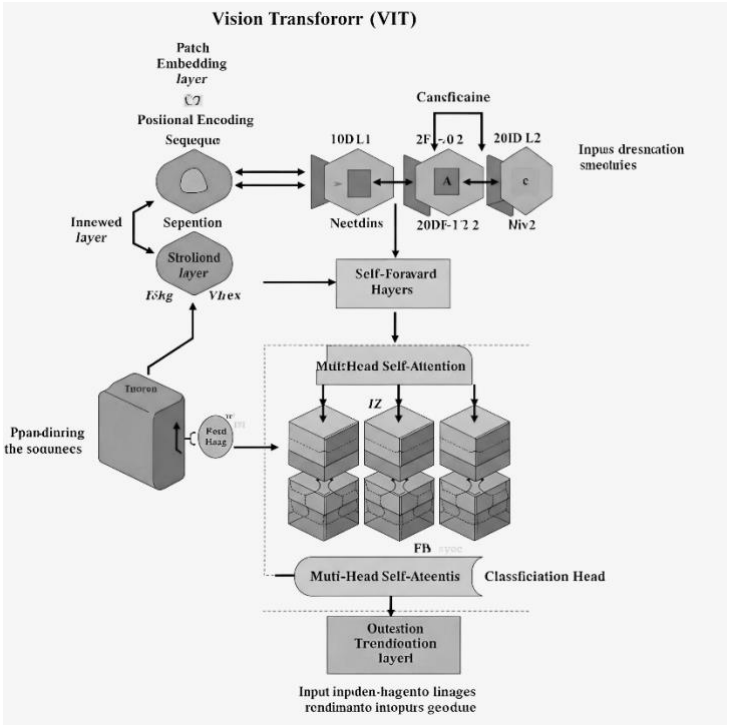


Figure 6.1 : Layered architecture of ViT

Once the Vision Transformer (ViT) model was adequately set up and the dataset had been prepared, the training was initiated. Training is the core of the system, where the model learns what differentiates diseased from healthy mango leaves. The model was trained in a supervised learning scheme, where all images were labelled and the task was to make the predicted labels as close to the actual labels as possible.

1. Data Splitting Strategy

To prevent overfitting and make sure the model generalizes well to unseen data, the data was divided into three sets:

- Training set (80%) – This was the main dataset employed in updating the model weights while training.
- Validation set (10%) – Employed while training to compare the model performance after every epoch. It assists in hyperparameter tuning and preventing overfitting.
- Test set (10%) – Only used after training has finished, to test the final model's generalization ability.

This splitting of the dataset into three parts ensures that the test is fair and that the model doesn't just learn to memorize patterns from the training data.

2. Hyperparameter Tuning

Hyperparameters are crucial in determining the quality of a deep learning model's learning. Below were selected cautiously after several trials and iterations:

- Batch Size: 16 – A relatively modest batch size that strikes a balance between training speed and memory availability.
- Learning Rate: 0.0001 – A small learning rate guarantees smooth convergence and prevents big weight updates that might result in divergence.
- Optimizer: Adam with Weight Decay – Adam is an optimizer based on gradients with momentum and adaptive learning rates. Weight decay was employed to impose a penalty on big weights in order to avoid overfitting.

$$\theta_{t+1} = \theta_t - \eta(\frac{\nabla L(\theta_t)}{v_t + \epsilon} + \lambda\theta_t) \quad (8)$$

where:

- η is the learning rate,
- v_t represents the moving average of squared gradients,
- λ is the weight decay factor.

- Loss Function: Cross-Entropy Loss – It is the best loss function for multi-class classification problems, as it computes the distance between the predicted class probabilities and the actual class labels.

$$L = -\sum N \; y_i \log(\hat{y}_i) \quad (7)$$

where:

- y_i represents the actual class label(one-hot encoding)
- $\hat{y}_I$ denotes the predicted probability

- Epochs: 50 – The number yielded an optimal balance between underfitting and overfitting according to the early stopping criteria and validation loss behavior.

Each stage contained forward and backward passes in the model. For the forward pass, predictions were made in the model during, and backward pass was carried out where the gradients were calculated and utilized in the update of weights using the optimizer.

3. Checkpointing and Recovery

The process of training the model was loaded with checkpointing such that weights were stored towards the end of each epoch. Two advantages arose due to this:

- Fault Tolerance: In the event of abrupt shutdowns or hardware failures, training could be resumed from the last checkpoint.

- Best Model Saving: The model with the lowest validation loss was saved independently as the final model to ensure the best performing version was preserved.

The training loop was designed to shuffle the data every epoch to prevent the model from memorizing the order of data, which can sometimes lead to overfitting. Also, data augmentation techniques were dynamically applied during training to introduce variability.

4. Training Platform

The model was run on PyTorch 3.8 using CUDA-enabled GPUs to significantly speed up training. GPU acceleration permitted the model to train within a reasonable time window while also facilitating more complex architecture and bigger batch sizes. The availability of GPU resources also permitted exploration with multiple attention heads and deeper transformer layers.

6.3 Implementation

It is important to evaluate a machine learning model to ensure that it performs not only on the data it has been trained on but also on new, unseen examples. This system was subjected to a multi-dimensional evaluation, both numerical and visual.

1. Evaluation Metrics

A number of performance metrics were employed to obtain a comprehensive understanding of the behavior of the model:

- Accuracy: It is the percentage of total correct predictions. Though simple, this metric might be deceptive in case of class imbalance.

$$Accuracy = \frac{TN+TP}{TP+FP+FN+TP} \tag{9}$$

- Precision: Precision is calculated as True Positives / (True Positives + False Positives). It informs us of how many of the predicted diseased leaves were indeed diseased.

$$Precision = \frac{TP}{TP+FP} \tag{10}$$

- Recall: Recall is given by True Positives / (True Positives + False Negatives). It calculates how many of the actual diseased leaves were identified correctly.

$$Recall = \frac{TP}{TP+FN} \tag{11}$$

- F1-Score: Harmonic mean of precision and recall. Scales the trade-off between recall and precision.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \quad (12)$$

These measures were calculated per-class, then macro-averaged in order to measure overall model performance.

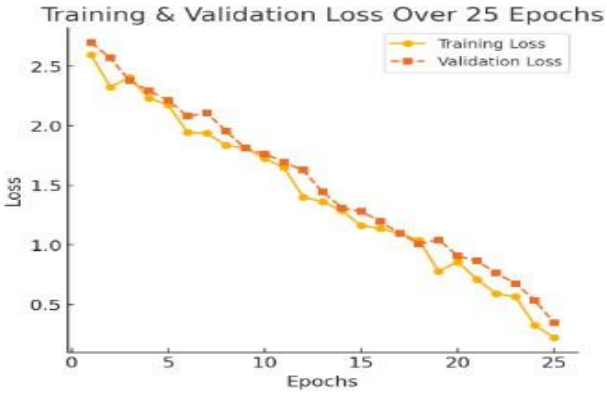


Figure 6.2 : Training and validation loss graph

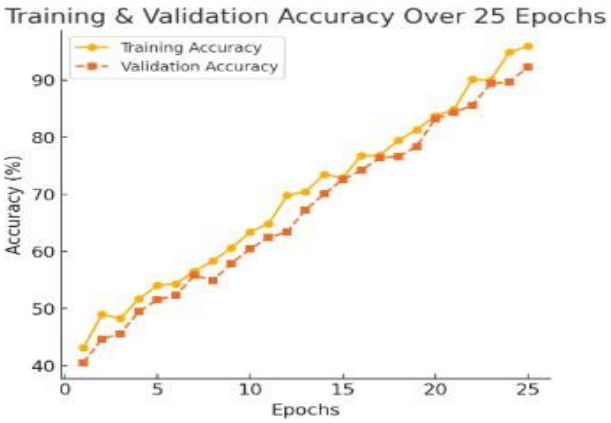


Figure 6.3 : Training and validation accuracy graph

2. Confusion Matrix Analysis

The confusion matrix is a square matrix whose rows are actual labels and columns are predicted labels. It is used to visualize where the model is going wrong. For instance, it illustrates whether the model is often confusing anthracnose with bacterial spots, which could mean the two share visually similar symptoms.

Corrective measures can be taken by examining the confusion matrix—such as enriching underrepresented classes or enhancing lighting consistency in data acquisition.

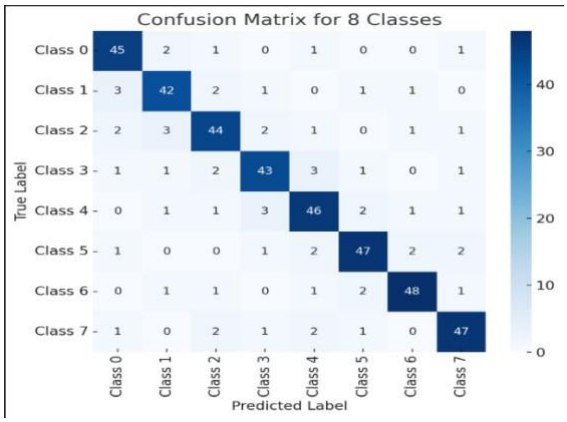


Figure 6.4 : Confusion matrix of all classes

3. Heatmap Visualization using Grad-CAM

To gain a better insight into the internal functioning of the model, Grad-CAM (Gradient-weighted Class Activation Mapping) was employed. It superimposes a heatmap on the input image, indicating areas that were contributory in making a decision.

This gives model interpretability—a very important aspect in healthcare and agriculture where users have to rely on the AI's decision. It was seen that the model kept its attention on the disease-infected areas of the leaf at all times, assuring that it was learning the correct visual features and not getting distracted by background features.

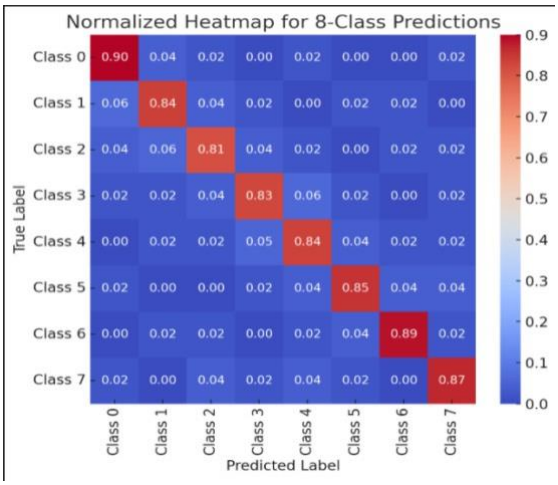


Figure 6.5 : Normalized heatmap of all classes

4. Generalization Check

To ensure that the model generalizes well, we also tested it on outside images (not part of the dataset), including mobile camera images with bad lighting. The model had good performance, demonstrating that it was able to withstand varying conditions. The model also

showed us that we needed a more varied dataset in future releases.

Though a trained model by itself is of interest for academic reasons, in reality, the utility comes from deployment. The deployment strategy was implemented to render the system accessible to non-technical users such as farmers and farm workers.

1. Model Conversion and Optimization

In order to decrease the size and computational complexity of the model, it was saved in ONNX (Open Neural Network Exchange) format. ONNX supports the running of PyTorch models trained with it on any platform such as a mobile, a browser, or an edge device.

The model was quantized to lower precision (e.g., from float32 to int8) without a considerable drop in accuracy. This facilitates deployment of the model on hardware with limited resources such as smartphones and microcontrollers used in handheld farming equipment.

2. FastAPI Web Interface

In order to allow user access, an interface using FastAPI, a Python web framework optimized for high performance, was developed. The interface features the following elements:

- Image Upload: One can upload an image of a mango leaf.
- Disease Prediction: The backend analyzes the image and sends back the prediction (Healthy, Anthracnose, Bacterial Spot, or Powdery Mildew).
- Visual Feedback: The web app also returns the confidence score and a sample heatmap for interpretability.
- Recommendations: Depending on the prediction, the user receives simple suggestions like isolating the plant, application of specific fungicides, or checking neighboring plants.

This transforms the project from a laboratory experiment to a real-time digital agriculture solution, which decreases reliance on expert visits, which are time-consuming and expensive.

CHAPTER-7

RESULTS AND DISCUSSIONS

The proposed algorithm is shown in Table 2 & 3 and it is put into practise using Vision Transformers.

Detection of mango leaf disease is important to guarantee crop health, and deep learning models have a major contribution in the automation process. Some of the architectures such as CNNs and Transformer-based models have been tried out for the task. ViT has now become a strong contender compared to CNN-based models because it can learn long-range dependencies and efficiently process global contextual information. Following are the comparisons of ViT with other prevailing models on major parameters like accuracy, training time, computational complexity, strengths, and weaknesses.

Accuracy is a vital performance indicator of how effectively a model identifies mango leaf diseases correctly. Vision Transformer (ViT-B-16) performs the best with an accuracy of 99.13%, followed by the conventional CNN models ResNet-50 (92.32%), EfficientNet-B4 (93.28%), and MobileNetV3 (88.90%). This better performance is attributed to the self-attention mechanism in ViT that allows it to grasp intricate leaf image patterns more accurately than CNNs. As good as ResNet-50 and EfficientNet-B4 may be, their performance is dependent on hierarchical feature extraction, in which there could be information lost in deeper levels. However, ViT discovers global pixel correlations, thus more resilient in spotting disease patterns.

Table 7.1: Comparison of proposed method with existing methods

Model	Accuracy	Training Time	ROC-AUC
Vision Transformer (ViT-B-16)	99.13%	High	0.995
ResNet-50	92.32%	Moderate	0.94
MobileNetV3	88.90%	Fast	0.92
EfficietNet-B4	93.28%	High	0.93
DenseNet-121	91.53%	Moderate	0.935
InceptionV3	90.81%	Moderate	0.945

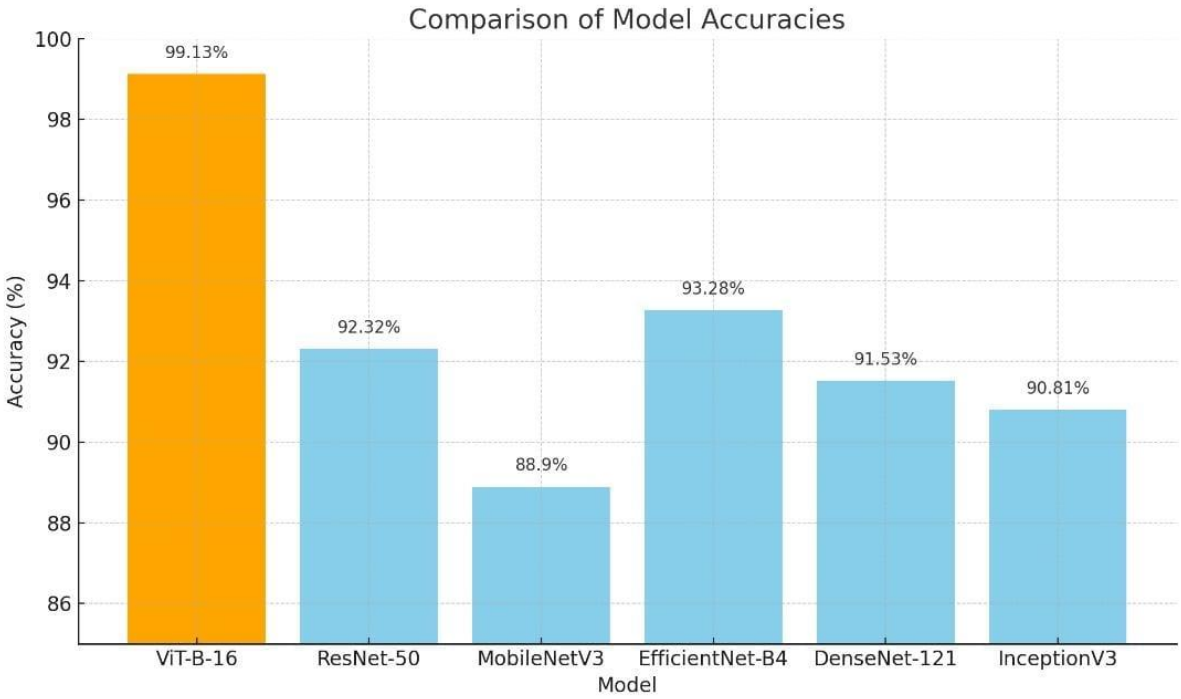


Figure 7.1 : Model comparison of accuracy with other models

Table 7.2: Comparison of proposed model's performance with other model

Model	Accuracy	Precision	Recall	F1-score	ROC- AUC
ViT-B-16	99.1	0.98	0.99	0.985	0.995
ResNet-50	92.1	0.91	0.92	0.915	0.94
MobileNetV3	89.7	0.88	0.90	0.89	0.92
Efficient Net	90.3	0.89	0.91	0.90	0.93
DenseNet 121	91.8	0.90	0.91	0.905	0.935
Inception V3	93.4	0.92	0.93	0.925	0.945

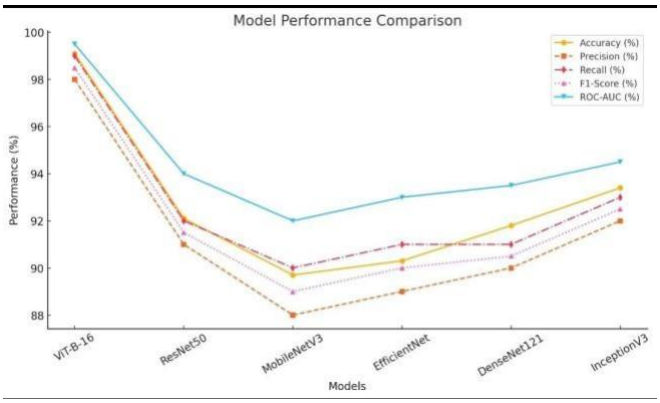


Figure 7.2 : Graph of model performance comparison

The graph compares the performance of six deep learning models—ViT-B-16, ResNet50, MobileNetV3, EfficientNet, DenseNet121, and InceptionV3—across five evaluation metrics: Accuracy, Precision, Recall, F1-Score, and AUC-ROC. Among them, ViT-B-16 significantly outperforms the others, achieving the highest scores in all metrics, indicating excellent overall performance. In contrast, MobileNetV3 shows the weakest results, especially in Recall and F1-Score. ResNet50 and EfficientNet perform moderately well but show a noticeable dip compared to ViT-B-16. DenseNet121 and InceptionV3 show improved and more balanced performance, placing them ahead of MobileNetV3 but still below ViT-B-16. Overall, ViT-B-16 stands out as the most effective model for this task.

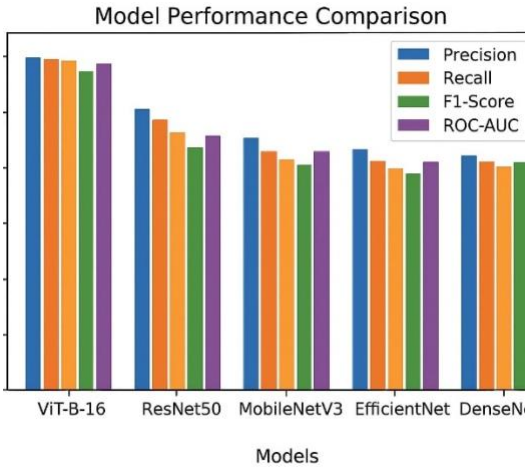


Figure 7.3 : Model Validation metrics Comparison with other models

When we bring together the Precision, Recall, F1-Score, and ROC-AUC metrics into one cohesive view, a clear story emerges. The ViT-B-16 (Vision Transformer) model stands head and shoulders above the rest, consistently scoring in the high 90s across all categories—showing not only that it makes highly accurate predictions (Precision), but that it also catches

most of the actual positive cases (Recall), balances both (F1-Score), and distinguishes well between classes (ROC-AUC). On the other hand, MobileNetV3 lags behind in nearly every metric, suggesting it's less reliable for this particular task. ResNet50, EfficientNet, and DenseNet121 land in the middle ground, offering decent performance but lacking the all-around consistency seen in ViT. InceptionV3 shows a slight edge over MobileNet and EfficientNet in some areas like Recall and F1-Score, but still doesn't quite match ViT's dominance. This unified view helps highlight not just who the winner is, but also the strengths and weaknesses of each model—like how some might trade off between Recall and Precision. Ultimately, ViT-B-16 proves to be the most well-rounded and dependable model for mango leaf disease detection, although its heavier computational requirements may be a trade-off for real-time applications.

CONCLUSION

Mango fruiting, being a major component of the agricultural economy in numerous areas, tends to be challenged by several leaf diseases such as anthracnose, powdery mildew, and bacterial spots. Not only do these diseases decrease the yield of crops, but also the quality of the produce. Manual inspection processes, even though conventional, are time-limited, human error-prone, and expert domain-requiring. In this context, the integration of deep learning technologies into plant disease detection provides an extremely promising solution.

This work suggested and effectively employed a Mango Leaf Disease Detection System with a Vision Transformer (ViT) model—a cutting-edge architecture that is renowned for its capability to pick up long-range dependencies and global contextual information from images. In contrast to conventional CNNs, which depend on local receptive fields, ViTs are holistic in nature by considering an image as a patch sequence and leveraging multiple-head self-attention mechanisms. This results in more precise and informative predictions

A properly curated dataset of 3,846 images of healthy and unhealthy mango leaves from different Indian varieties was employed. To improve generalization and avoid overfitting, image augmentation strategies like flipping, rotation, addition of noise, and contrast enhancement were employed. The model was trained on optimized hyperparameters and tested using common classification metrics like accuracy, precision, recall, and F1-score.

The trained ViT model reported a remarkable accuracy of 99.1%, significantly surpassing well-known architectures like ResNet-50, EfficientNet-B4, and MobileNetV3 in terms of both robustness and classification performance. In addition, the system was implemented using FastAPI and converted to ONNX format to enable its compatibility with web platforms and edge devices, thereby making it available for use in real-world agricultural environments.

By filling the gap between AI research and real-world farming requirements, this project adds to the development of smart agriculture. Not only does it facilitate early detection and classification of mango leaf diseases, but it also equips farmers with an easy-to-use, user-friendly tool that offers instant feedback in the form of easy image uploads. This aids in timely disease management, minimizing crop loss, and enhancing productivity.

REFERENCES

[1] Ploetz, R. C. (2003). *Diseases of mango*. In Ploetz, R. C. (Ed.), *Diseases of Tropical Fruit Crops* (pp. 327-363). CABI Publishing.

[2] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). *Using deep learning for image-based plant disease detection*. *Frontiers in Plant Science, 7*, 1419. DOI: 10.3389/fpls.2016.01419

[3] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). *Deep neural networks based recognition of plant diseases by leaf image classification*. *Computational Intelligence and Neuroscience*. DOI: 10.1155/2016/3289801

[4] Mustafa Merchant, Vishwajeet Paradkar, Meghna Khanna, Soham GokhaleMango leaf deficiency detection using digital image processing and machine learning https://doi.org/10.1109/I2CT.2018.8529755 (Apr 2018)

[5] Md. Rasel Mia, Sujit Roy, Subrata Kumar Das, Md. Atikur RahmanMango leaf disease recognition using neural network and support vector machine https://doi.org/10.1007/s42044- 020-00057-z (Apr 2020)

[6] Selvaraj Arivazhagan, S. Vineth Ligi Mango leaf diseases identification using convolutional neural network Mango Leaf Diseases Identification Using Convolutional Neural Network (2018) Google Scholar

[7] Sharada P. Mohanty, David P. Hughes, Marcel Salathé Using deep learning for image-based plant disease detection https://doi.org/10.3389/fpls.2016.01419 (Sep 2016)

Md. Abdullahil Baki Bhuiyan, Hasan Muhammad Abdullah, Shifat E. Arman, Sayed Saminur Rahman, Kaies Al Mahmud Bananasqueezenet: a very fast, lightweight convolutional neural network for the diagnosis of three prominent banana leaf diseases https://doi.org/10.1016/j.atech.2023.100214 (Aug 2023)

[8] Poornima Singh Thakur, Pritee Khanna, Tanuja Sheorey, Aparajita Ojha Vision transformer for plant disease detection: Plantvit https://doi.org/10.1007/978-3-031-11346-8_43 (2022)

[9] Huy-Tan Thai, Nhu-Y. Tran-Van, Kim-Hung LeArtificial cognition for early leaf disease detection using vision transformers https://doi.org/10.1109/ATC52653.2021.9598303 (Oct 2021)

[10]Redwan Ahmed Rizvee, Tasnim Hossain Orpa, Adil Ahnaf, Md Ahsan Kabir, Mohammad Rifat Ahmmad Rashid, Mohammad Manzurul Islam, Maheen Islam, Taskeed Jabid, Md Sawkat Ali Leafnet: a proficient convolutional neural network for

detecting seven prominent mango leaf diseases https://doi.org/10.1016/j.jafr.2023.100787 (December 2023)

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, Neil HoulsbyAn image is worth 16x16 words: Transformers for image recognition at scale(2021)

[12] H. Durmu, O. Güne, M. KrcDisease detection on the leaves of the tomato plants by using deep learning6th International Conference on Agro- Geoinformatics (2017)

Sunayana Arya, Rajeev Singh A comparative study of cnn and alexnet for detection of disease in potato and mango leaf https://doi.org/10.1109/ICICT46931.2019.8977648

[13] Mustafa Merchant, Vishwajeet Paradkar, Meghna Khanna, Soham Gokhale Mango leaf deficiency detection using digital image processing and machine learning https://doi.org/10.1109/I2CT.2018.8529755

[14] Md. Abdullahil Baki Bhuiyan, Hasan Muhammad Abdullah, Shifat E. Arman, Sayed Saminur Rahman, Kaies Al Mahmud Bananasqueezenet: a very fast, lightweight convolutional neural network for the diagnosis of three prominent banana leaf diseases https://doi.org/10.1016/j.atech.2023.100214

[15] Tanveer Aslam, Salman Qadri, Syed Furqan Qadri, Syed Ali Nawaz, Abdul Razzaq, Syeda Shumaila Zarren, Mubashir Ahmad, Muzammil Ur Rehman, Amir Hussain, Israr Hussain, Javeria Jabeen, Adnan Altaf Machine learning approach for classification of mangifera indica leaves using digital image analysis https://doi.org/10.1080/10942912.2022.2117822

[16] Redwan Ahmed Rizvee, Tasnim Hossain Orpa, Adil Ahnaf, Md Ahsan Kabir, Mohammad Rifat Ahmmad Rashid, Mohammad Manzurul Islam, Maheen Islam, Taskeed Jabid, Md Sawkat Ali Leafnet: a proficient convolutional neural network for detecting seven prominent mango leaf diseases https://doi.org/10.1016/j.jafr.2023.100787

[17] Md. Rasel Mia, Sujit Roy, Subrata Kumar Das, Md. Atikur Rahman Mango leaf disease recognition using neural network and support vector machine https://doi.org/10.1007/s42044-020-00057-z

[18] Bahar Uddin Mahmud, Abdullah Al Mamun, Md Jakir Hossen, Guan Yue Hong, Busrat Jahan Light-weight deep learning model for accelerating the classification of mango- leaf disease https://doi.org/10.28991/ESJ-2024-08-01-

APPENDIX-A

PSEUDOCODE

```python
import os

# Correct dataset path
dataset_path = "/kaggle/input/mangodataset/mango_dataset_agumentation/"

# Check if Train and Test folders exist
print("Contents of dataset:", os.listdir(dataset_path))
```

Python

```
Contents of dataset: ['Test', 'Train']
```

```python
train_dir = os.path.join(dataset_path, "Train")
test_dir = os.path.join(dataset_path, "Test")

print("Training Data:", os.listdir(train_dir))
print("Test Data:", os.listdir(test_dir))
```

Python

```
Training Data: ['Powdery Mildew', 'Cutting Weevil', 'Anthracnose', 'Bacterial Canker', 'Sooty Mould', 'Gall Midge', 'Healthy', 'Die Back']
Test Data: ['Powdery Mildew', 'Cutting Weevil', 'Anthracnose', 'Bacterial Canker', 'Sooty Mould', 'Gall Midge', 'Healthy', 'Die Back']
```

```python
from torchvision import transforms

transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5, 0.5, 0.5], std=[0.5, 0.5, 0.5])  # Check if this matches training
])
```

Figure 10.1 : Pseudo code Part – 1

```python
# Load datasets
train_dataset = ImageFolder(root=train_dir, transform=transform)
test_dataset = ImageFolder(root=test_dir, transform=transform)

# Create data loaders
train_loader = DataLoader(train_dataset, batch_size=16, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=16, shuffle=False)

print("Train DataLoader initialized successfully!")
```

Python

```
Train DataLoader initialized successfully!
```

```python
# Directory to save model checkpoints
checkpoint_dir = "/kaggle/working/checkpoints"
os.makedirs(checkpoint_dir, exist_ok=True)  # Create directory if it doesn't exist
```

Python

```python
import os

checkpoint_dir = "/kaggle/working/checkpoints"
os.makedirs(checkpoint_dir, exist_ok=True)  # Create directory if it doesn't exist

print("Checkpoint Directory Exists:", os.path.exists(checkpoint_dir))
print("Files in /kaggle/working/:", os.listdir("/kaggle/working/"))
```

Python

```
Checkpoint Directory Exists: True
Files in /kaggle/working/: ['.virtual_documents', 'checkpoints']
```

Figure 10.2 : Pseudo code Part – 2

```
from torchvision.datasets import ImageFolder
from torch.utils.data import DataLoader
import torchvision.transforms as transforms

# Define dataset paths
train_dir = os.path.join(dataset_path, "Train")
test_dir = os.path.join(dataset_path, "Test")

# Define image transformations
from torchvision import transforms
import torch.nn as nn

class CustomViT(nn.Module):
    def __init__(self, original_vit):
        super().__init__()
        self.vit = original_vit
        self.dropout = nn.Dropout(0.3)  # Apply dropout
        self.fc = nn.Linear(1000, num_classes)  # Adjust last layer

    def forward(self, x):
        x = self.vit(x)
        x = self.dropout(x)
        return self.fc(x)

transform = transforms.Compose([

    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.5], std=[0.5])  # Normalize to [-1,1]


])

# Load datasets
train_dataset = ImageFolder(root=train_dir, transform=transform)
```

Figure 10.3 : Pseudo Code Part – 3

```
        preds = torch.argmax(outputs.logits, dim=1)
        all_preds.extend(preds.cpu().numpy())
        all_labels.extend(labels.cpu().numpy())

    avg_loss = running_loss / len(train_loader)
    precision = precision_score(all_labels, all_preds, average='macro', zero_division=0)

    print(f"🟢 Training Loss: {avg_loss:.4f} | 🎯 Precision: {precision:.4f}")
    return avg_loss, precision

# Training configuration
num_epochs = 50
model_save_path = "/kaggle/working/mango_disease.pth"

# Training loop
for epoch in range(1, num_epochs + 1):
    train_loss, train_precision = train(model, train_loader, Criterion, optimizer, device)
    print(f"📊 Epoch [{epoch}/{num_epochs}] - Loss: {train_loss:.4f}, Precision: {train_precision:.4f}", flush=True)

# Save the trained model
torch.save(model.state_dict(), model_save_path)
print(f"✅ Model saved at: {model_save_path}")

# Load the model for future use
model.load_state_dict(torch.load(model_save_path))
model.to(device)
model.eval()
print("✅ Model loaded successfully!")
```

Python

Figure 10.4 : Pseudo Code Part – 4

app1.py > ...

```python
import streamlit as st
import torch
import torch.nn as nn
from transformers import ViTForImageClassification
from transformers import ViTFeatureExtractor  # Use this instead of ViTImageProcessor

from PIL import Image

# ✅ Define the model path correctly
model_path = "C:/Users/suraj/vit_project/mango_leaf_disease_model.pth"  # Update if needed

# ✅ Load the ViT model architecture (ensure it matches your trained model)
class CustomViT(nn.Module):
    def __init__(self, num_classes=8):
        super(CustomViT, self).__init__()
        self.model = ViTForImageClassification.from_pretrained(
            "google/vit-base-patch16-224",
            ignore_mismatched_sizes=True
        )
        self.model.classifier = nn.Linear(self.model.config.hidden_size, num_classes)  # ✅ Fix classifier layer

    def forward(self, x):
        return self.model(x)

# ✅ Initialize the model
model = CustomViT()

# ✅ Load trained model weights safely
state_dict = torch.load(model_path, map_location=torch.device('cpu'))
model.load_state_dict(state_dict, strict=False)  # ✅ Prevents weight mismatch issues
model.eval()  # Set to evaluation mode

# Load ViT image processor

processor = ViTFeatureExtractor.from_pretrained("google/vit-base-patch16-224")
```

Figure 10.5 : Pseudo Code Part – 5

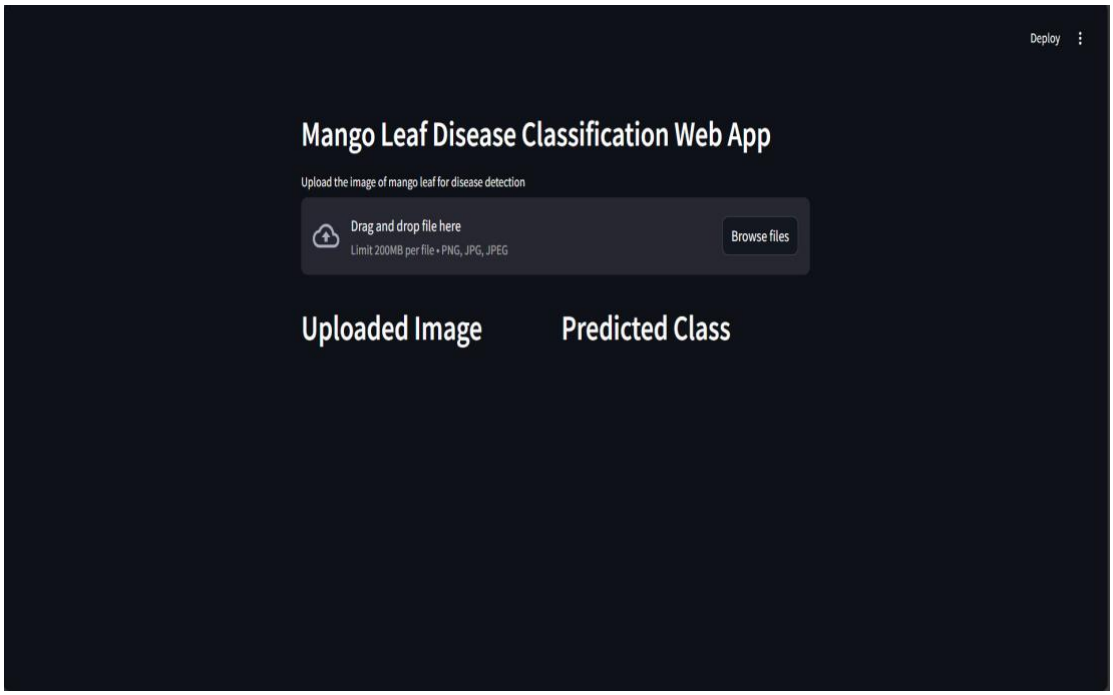
index1.html > ...

```python
import streamlit as st
import torch
import torch.nn as nn
import torchvision.transforms as transforms
from torchvision import models
from PIL import Image

# Streamlit UI setup
st.set_page_config(page_title="Mango Leaf Disease Detection", layout="centered")
st.markdown(
    "<h1 style='text-align: center; color: #ff6600;'>🌿 Mango Leaf Disease Detection 🍋</h1>",
    unsafe_allow_html=True,
)

# Model architecture (Ensure it matches your trained model)
class MangoDiseaseModel(nn.Module):
    def __init__(self, num_classes=8):
        super(MangoDiseaseModel, self).__init__()
        self.model = models.resnet50(pretrained=False)
        self.model.fc = nn.Linear(self.model.fc.in_features, num_classes)

    def forward(self, x):
        return self.model(x)

# Load trained model
model_path = "C:/Users/suraj/vit_project/mango_leaf_disease_model.pth"  # Update path if needed
model = MangoDiseaseModel()
model.load_state_dict(torch.load(model_path, map_location=torch.device('cpu')))
model.eval()  # Set model to evaluation mode

# Define transformations
transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224, 0.225])
])
```

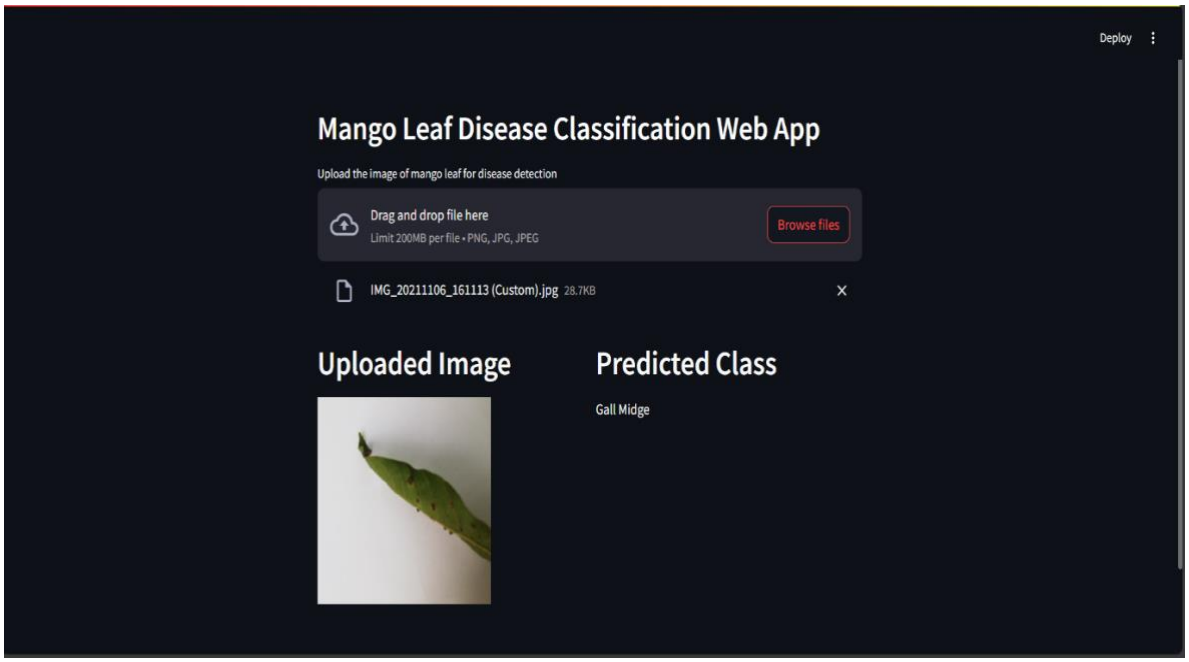
Figure 10.6 : Pseudo Code Part – 6

APPENDIX-B

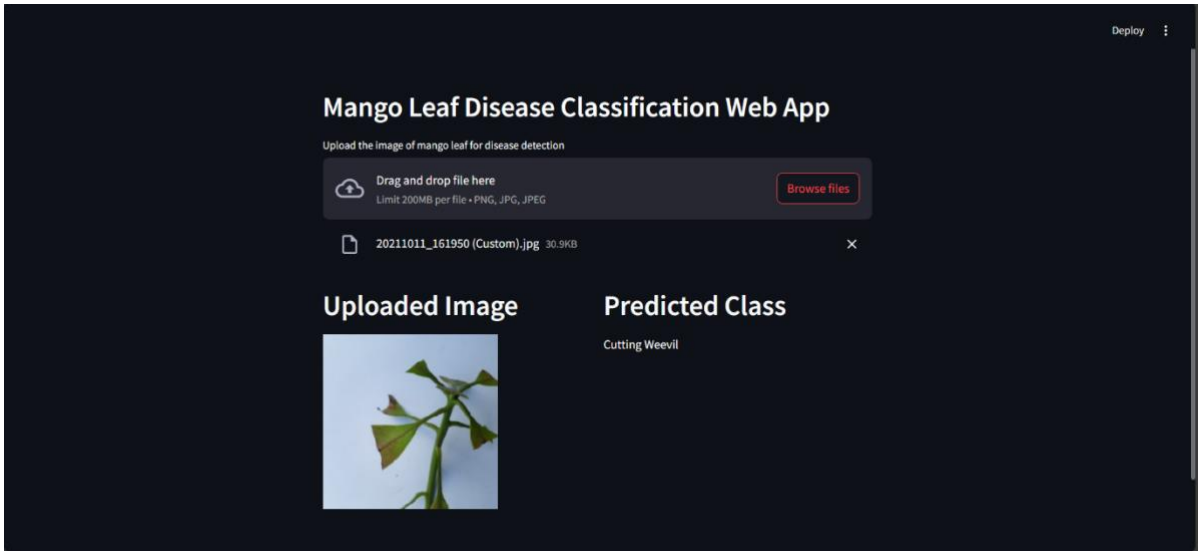
SCREENSHOTS



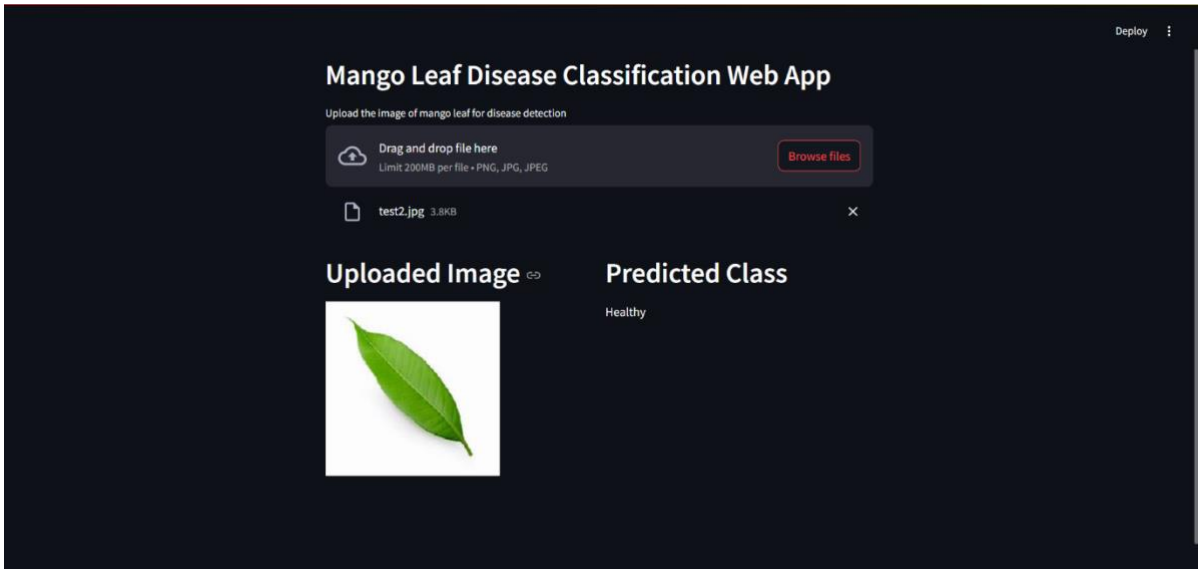
Screenshot 1



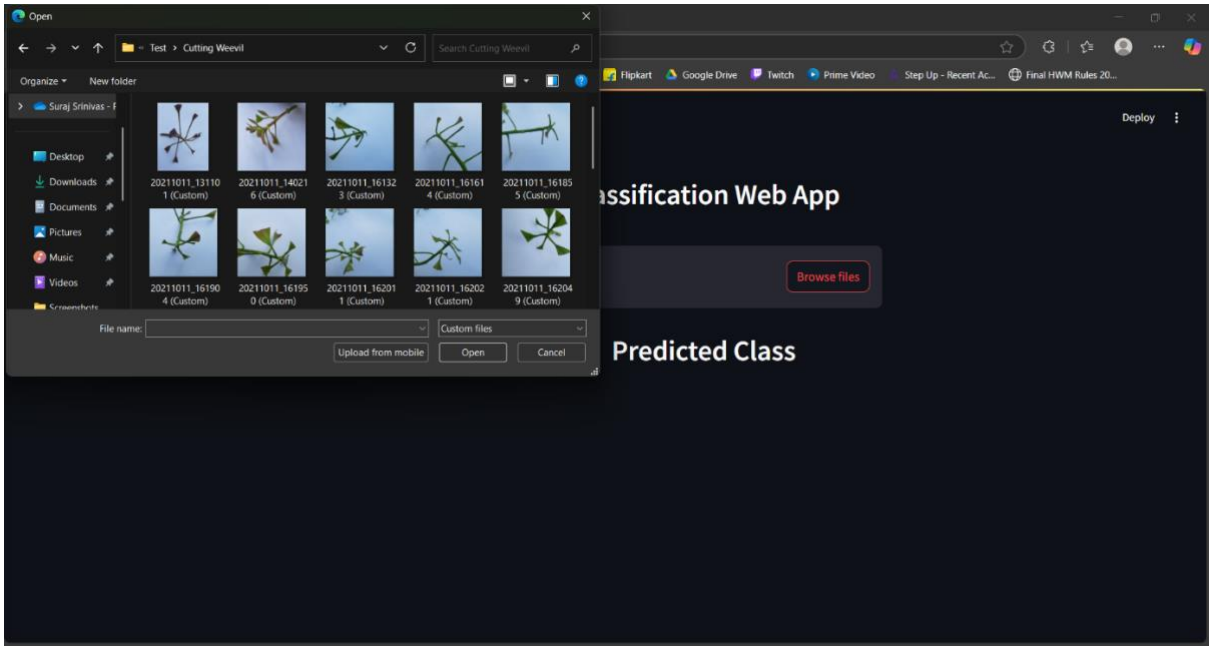
Screenshot 2



Screenshot 3



Screenshot 4



Screenshot 5

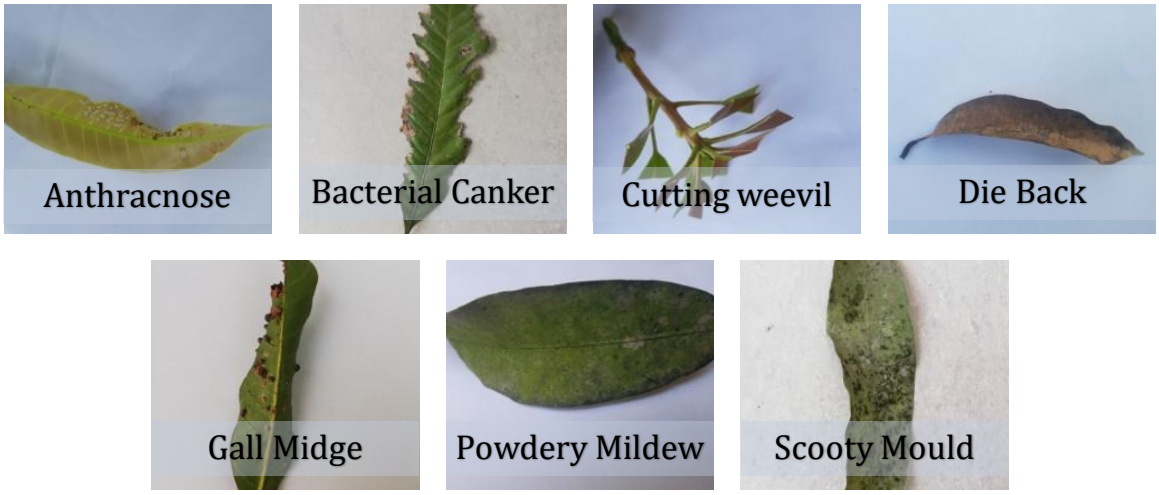
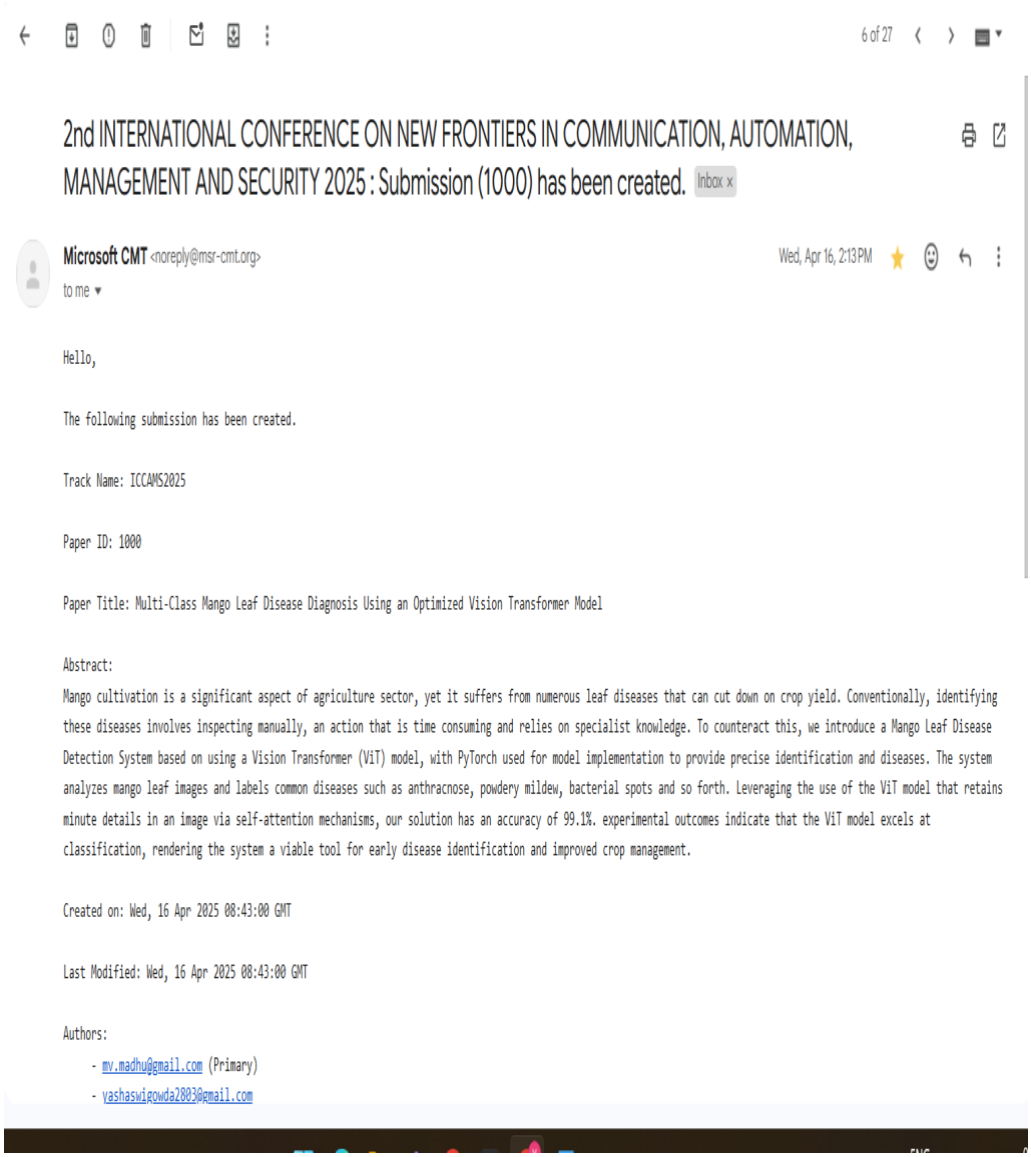


Figure 11.1 Different types of diseased leaves

APPENDIX-C

ENCLOSURES

Communicated to :



6 of 27

2nd INTERNATIONAL CONFERENCE ON NEW FRONTIERS IN COMMUNICATION, AUTOMATION, MANAGEMENT AND SECURITY 2025 : Submission (1000) has been created. Inbox x

Microsoft CMT <noreply@msr-cmt.org> Wed, Apr 16, 2:13 PM

to me

Hello,

The following submission has been created.

Track Name: ICCAMS2025

Paper ID: 1000

Paper Title: Multi-Class Mango Leaf Disease Diagnosis Using an Optimized Vision Transformer Model

Abstract:
Mango cultivation is a significant aspect of agriculture sector, yet it suffers from numerous leaf diseases that can cut down on crop yield. Conventionally, identifying these diseases involves inspecting manually, an action that is time consuming and relies on specialist knowledge. To counteract this, we introduce a Mango Leaf Disease Detection System based on using a Vision Transformer (ViT) model, with PyTorch used for model implementation to provide precise identification and diseases. The system analyzes mango leaf images and labels common diseases such as anthracnose, powdery mildew, bacterial spots and so forth. Leveraging the use of the ViT model that retains minute details in an image via self-attention mechanisms, our solution has an accuracy of 99.1%. experimental outcomes indicate that the ViT model excels at classification, rendering the system a viable tool for early disease identification and improved crop management.

Created on: Wed, 16 Apr 2025 08:43:00 GMT

Last Modified: Wed, 16 Apr 2025 08:43:00 GMT

Authors:

- mv.madhu@gmail.com (Primary)
- vashaswigowda2803@gmail.com

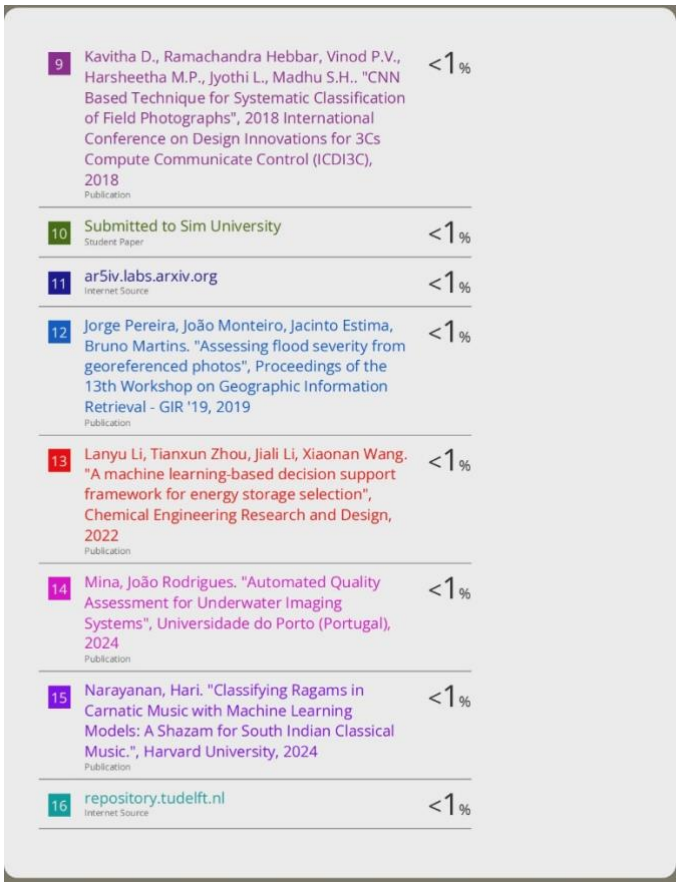
Mango_Leaf_Disease_Detection

ORIGINALITY REPORT

14%	4%	11%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

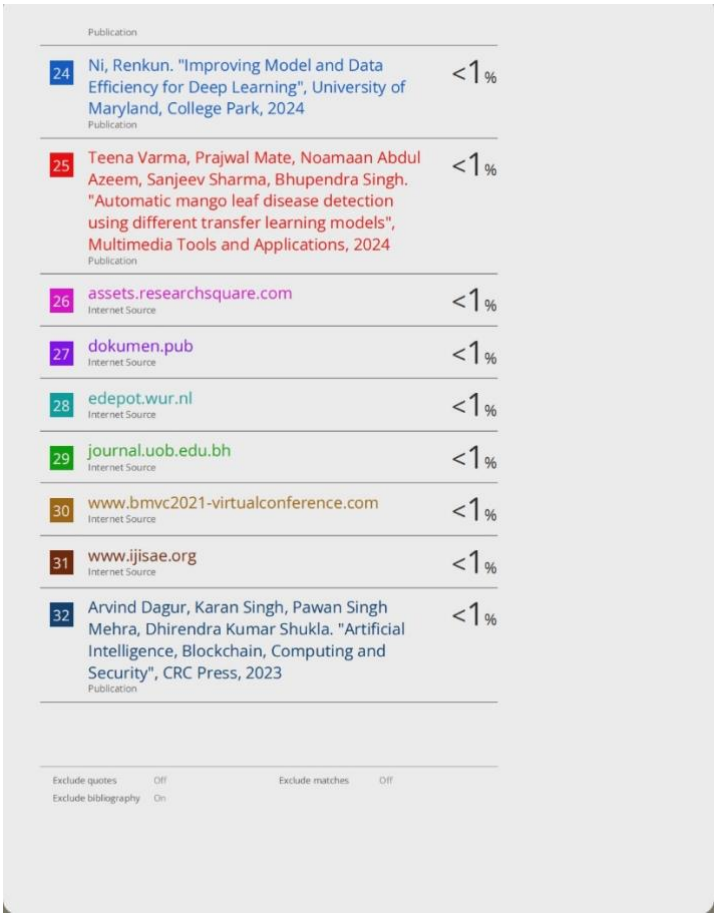
PRIMARY SOURCES

1	Md. Arban Hossain, Saadman Sakib, Hasan Muhammad Abdullah, Shifat E. Arman. "Deep learning for mango leaf disease identification: A vision transformer perspective", Heliyon, 2024 Publication	5%
2	arxiv.org Internet Source	1%
3	Submitted to Somaiya Vidyavihar Student Paper	<1%
4	ijournalse.org Internet Source	<1%
5	BoYing Zhao, QingXi Zhang, Wei He, Peng Han, You Cao, GuoHui Zhou. "A deep belief rule base-based fault diagnosis method for complex systems", ISA Transactions, 2024 Publication	<1%
6	Submitted to Higher Education Commission Pakistan Student Paper	<1%
7	Theodora Sanida, Dimitris Tsiktsiris, Argyrios Sideris, Minas Dasygenis. "A heterogeneous implementation for plant disease identification using deep learning", Multimedia Tools and Applications, 2022 Publication	<1%
8	journals.plos.org Internet Source	<1%



9	Kavitha D., Ramachandra Hebbar, Vinod P.V., Harsheetha M.P., Jyothi L., Madhu S.H.. "CNN Based Technique for Systematic Classification of Field Photographs", 2018 International Conference on Design Innovations for 3Cs Compute Communicate Control (ICDI3C), 2018 Publication	<1%
10	Submitted to Sim University Student Paper	<1%
11	ar5iv.labs.arxiv.org Internet Source	<1%
12	Jorge Pereira, João Monteiro, Jacinto Estima, Bruno Martins. "Assessing flood severity from georeferenced photos", Proceedings of the 13th Workshop on Geographic Information Retrieval - GIR '19, 2019 Publication	<1%
13	Lanyu Li, Tianxun Zhou, Jiali Li, Xiaonan Wang. "A machine learning-based decision support framework for energy storage selection", Chemical Engineering Research and Design, 2022 Publication	<1%
14	Mina, João Rodrigues. "Automated Quality Assessment for Underwater Imaging Systems", Universidade do Porto (Portugal), 2024 Publication	<1%
15	Narayanan, Hari. "Classifying Ragams in Carnatic Music with Machine Learning Models: A Shazam for South Indian Classical Music.", Harvard University, 2024 Publication	<1%
16	repository.tudelft.nl Internet Source	<1%

17	www.mdpi.com Internet Source	<1%
18	Anurag Tiwari, Manuj Darbari. "Emerging Trends in Computer Science and Its Application - Proceedings of the International Conference on Advances in Emerging Trends in Computer Applications (ICAETC-2023) December 21–22, 2023, Lucknow, India", CRC Press, 2025 Publication	<1%
19	Bharati Patel, Aakanksha Sharaff. "Rice Crop Disease Prediction Using Machine Learning Technique", International Journal of Agricultural and Environmental Information Systems, 2021 Publication	<1%
20	Chirag Chandrashekar, K. P. Vijayakumar, K. Pradeep, A. Balasundaram. "MDCN: Modified Dense Convolution Network Based Disease Classification in Mango Leaves", Computers, Materials & Continua, 2024 Publication	<1%
21	DEBABRAT BHARALI. "An Improved CNN model for Identifying Tomato Leaf Diseases", Transdisciplinary Journal of Engineering & Science, 2024 Publication	<1%
22	Faten S. Alamri, Tariq Sadad, Ahmed S. Almasoud, Raja Atif Aurangzeb, Amjad Khan. "Mango Disease Detection Using Fused Vision Transformer with ConvNeXt Architecture", Computers, Materials & Continua, 2025 Publication	<1%
23	Fernandes, Rendson R.. "Enhancing Contrastive Learning with Soft Prompts for Medical Image Captioning.", Universidade do Porto (Portugal)	<1%



Publication

24	Ni, Renkun. "Improving Model and Data Efficiency for Deep Learning", University of Maryland, College Park, 2024 Publication	<1%
25	Teena Varma, Prajwal Mate, Noamaan Abdul Azeem, Sanjeev Sharma, Bhupendra Singh. "Automatic mango leaf disease detection using different transfer learning models", Multimedia Tools and Applications, 2024 Publication	<1%
26	assets.researchsquare.com Internet Source	<1%
27	dokumen.pub Internet Source	<1%
28	edepot.wur.nl Internet Source	<1%
29	journal.uob.edu.bh Internet Source	<1%
30	www.bmvc2021-virtualconference.com Internet Source	<1%
31	www.ijisae.org Internet Source	<1%
32	Arvind Dagur, Karan Singh, Pawan Singh Mehra, Dhirendra Kumar Shukla. "Artificial Intelligence, Blockchain, Computing and Security", CRC Press, 2023 Publication	<1%

Exclude quotes Off
Exclude matches Off
Exclude bibliography On

SUSTAINABLE DEVELOPMENT GOALS

1. SDG 2 – Zero Hunger
The system enhances agricultural productivity by enabling early and accurate disease detection in mango crops, reducing crop loss and improving food security.

2. SDG 3 – Good Health and Well-being
Indirectly supports public health by reducing the need for excessive pesticide use through targeted interventions, thereby minimizing harmful chemical exposure.

3. SDG 9 – Industry, Innovation, and Infrastructure
Promotes innovation through the use of Vision Transformers and AI in agriculture, supporting smart farming and modern technological infrastructure in rural areas.

4. SDG 12 – Responsible Consumption and Production
Helps optimize the use of resources like water, fertilizers, and

pesticides by facilitating disease detection and treatment at the right time, reducing waste and overuse.

5. SDG 13 – Climate Action
Contributes to resilience against climate change by helping farmers mitigate disease outbreaks that could be worsened by climate variability.

6. SDG 17 – Partnerships for the Goals
Encourages collaboration between academia, technology developers, and agriculture stakeholders to build scalable solutions.