BCSE498J Project-II – Capstone Project

Plant Disease Detection using Advanced Deep Learning Models

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

21BCE2544	**Sristi Agarwal**
21BCB0133	**G. Hema**

Under the Supervision of

Dr. Mukku Nisanth Kartheek

Assistant Professor Senior Grade 1

School of Computer Science and Engineering (SCOPE)

VIT

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

April 2025

DECLARATION

I hereby declare that the project entitled **Plant Disease Detection using Advanced Deep Learning Models** submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering* to VIT is a record of bonafide work carried out by me under the supervision of **Dr. Mukku Nisanth Kartheek.**

I further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date : 15/4/25

[illegible]

Signature of the Candidate

CERTIFICATE

This is to certify that the project entitled **Plant Disease Detection using Advanced Deep Learning Models** submitted by **G Hema (21BCB0133) and School of Computer Science and Engineering**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science and Engineering*, is a record of bonafide work carried out by her under my supervision during Winter Semester 2024-2025, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The project fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date : 24/4/25

[illegible]

Signature of the Guide

VELLORE INSTITUTE OF TECHNOLOGY (VIT)
(Deemed to be University under section 3 of UGC Act, 1956)
School of Computer Science & Engineering (SCOPE)
Vellore-632 014, T.N., India

[illegible]

ternal Examiner

[illegible]

External Examiner

Dr. Rajkumar S

CSE with Bioinformatics

EXECUTIVE SUMMARY

This project describes a deep learning-based method for potato leaf disease classification using sophisticated neural network models. The aim is to design an accurate, scalable, and cross-compatible model that can identify diseases such as Early Blight and Late Blight in potato leaves, and to test its performance on various types of crops like apple and bell pepper.
For this, a high-quality dataset from Kaggle's "PlantVillage" repository containing three major classes, i.e., Potato Early Blight, Potato Late Blight, and Healthy leaves, was utilized. The dataset was well analyzed and also visualized using pie charts and bar plots in order to acquire insight into the class distributions. Preprocessing operations such as image resizing, normalization, and data augmentation were applied to enhance the quality and diversity of training data. A number of models were experimented and compared, including EfficientNetB3, ResNet, MobileNet, and VGG, individually as CNNs and in hybrid form with LSTM networks. The integration of LSTM with CNN architectures was the central innovation in the project, which allowed the models to learn both spatial and sequential features from the leaf images. The highest accuracy of 96.51% over all the models was achieved by the EfficientNetB3+LSTM hybrid model on the potato leaf data, significantly surpassing traditional architectures.
Further cross-validation was carried out using apple and bell pepper data to try out the generalizability of the model with respect to crops. The EfficientNetB3+LSTM model performed remarkably, with accuracy levels of 84.78% on apple leaves and 98.00% on bell pepper leaves.
The architecture was tested using a well-defined workflow, from data preprocessing and preloading to model testing, learning, and real-time prediction. Input through images and prediction of disease via a simple user interface rendered the solution practical to end-users such as agro-researchers and farmers.
This project demonstrates the robustness and power of utilizing CNN as well as LSTM to identify and classify diseases in plants. It also provides a groundwork for further development on multi-crop disease detection, mobile app development, and integrating with IoT sensors for smart agriculture solutions

ACKNOWLEDGEMENTS

I am deeply grateful to the management of Vellore Institute of Technology (VIT) for roviding me with the opportunity and resources to undertake this project. Their commitment to ostering a conductive learning environment has been instrumental in my academic journey. The upport and infrastructure provided by VIT have enabled me to explore and develop my ideas to heir fullest potential.

My sincere thanks to Dr. Jaisankar N, Dean - School of Computer Science and Engineering SCOPE), for his unwavering support and encouragement. His leadership and vision have reatly inspired me to strive for excellence.

I express my profound appreciation to Dr.RajKumar S, the Head of the CSE with Bioinformatics for his insightful guidance and continuous support. His expertise and advice have een crucial in shaping throughout the course. His constructive feedback and encouragement ave been invaluable in overcoming challenges and achieving goals.

I am immensely thankful to my project supervisor, Dr.Mukku Nisanth Kartheek, for his ledicated mentorship and invaluable feedback. His patience, knowledge, and encouragement ave been pivotal in the successful completion of this project. My supervisor's willingness to hare his expertise and provide thoughtful guidance has been instrumental in refining my ideas nd methodologies. His support has not only contributed to the success of this project but has lso enriched my overall academic experience.

Thank you all for your contributions and support.

G. Hema

Name of the Candidate

TABLE OF CONTENTS

List of Tables

List of Figures

List of Abbreviations

Abbreviations	**Full Form**
CNN	Convolutional Neural Network
DL	Deep Learning
GLCM	Gray Level Co-occurrence Matrix
LSTM	Long Short-Term Memory
ML	Machine Learning
RESNET	Residual Network
SVM	Support Vector Machine
VGG	Visual Geometry Group
YOLO	You Only Look Once

Symbols and Notations

η	Learning rate
ϵ	Small constant for numerical stability
θ_t	Parameters at time step t

Chapter 1

INTRODUCTION

1.1 BACKGROUND

Agriculture is the backbone of human survival through the provision of food, raw materials, and economic stability. Plant diseases are a major threat to world food security, lowering crop yields and incurring huge economic losses. If not detected and treated early, the diseases can spread very fast, ruining entire agricultural areas. Conventional plant disease detection methods, including manual inspection by farmers and agriculture professionals, tend to be time-consuming, labor-intensive, and subject to human bias. Therefore, there is an urgent demand for reliable, efficient, and automated plant disease identification systems to help overcome these limitations and advance sustainable agriculture.

The proposed model aims at developing an effective deep learning system capable of classifying various plant diseases throughout multiple species with a focus on potato plants and bell pepper (capsicum) and apple leaves. The project addresses common data problems and model generalization issues by combining data augmentation of images with transfer learning techniques and class balancing strategies and model validation methods.

Two thousand one hundred fifty-two images of potato leaves make up the main dataset inwhich Potato Early Blight and Potato Late Blight have 1000 images each while Potato Healthy obtains 152 images. The dataset obtained from Kaggle exists as publicly accessible resources which contain actual-world disease examples that represent the sample well. The unbalanced distribution of classes, especially the shortage of healthy potato leaves posed a substantial challenge to achieve balanced learning. Obtaining added image data from bell pepper and apple plants aimed to enhance the model's capability for cross-crop type generalization. There are 2,475 bell pepper pictures containing bacterial leaf disease and healthy leaves included in the dataset while the apple section contains approximately 300 photos that show Apple Scab disease alongside Apple Rust and Healthy Apple Leaves.

Image data augmentation methods from TensorFlow ImageDataGenerator helped overcome dataset limitations caused by both small size and unbalance. The augmentation pipeline used different operations to expand the size and enhance the diversity of training data. The pipeline applied horizontal flip, zoom, shear, random rotation functions that reached a maximum of 40 degrees as well as width and height shift alongside pixel value scaling to standardize input data.

The applied augmentations simultaneously reduce overfitting and replicate various possible environmental conditions for leaf image acquisition.

A calculation method for class weights existed through Scikit-learn utilities to deal with class imbalance. The training loss gives elevated importance to underrepresented categories through these weights which preserves equivalent representation of minority classes. The model learns equally from all classes through this technique which improves its ability to detect diseases affecting low-sample classes.

The approach of transfer learning let the model leverage learning features from ImageNet and other big datasets that pre-trained convolutional neural networks (CNNs) acquired. The models received a fine-tuning process on plant disease datasets to transfer extracted features from an extensive task domain to a specialized leaf disease classification domain. We employed Adam as the optimizer since it demonstrates both efficiency and fast convergence while categorical cross-entropy loss served as our choice for multi-class classification.

The use of training callbacks enabled better stability and model outcomes in the training process. The trained model utilizes three essential callbacks including EarlyStopping to end training after validation metrics stop improving as well as ModelCheckpoint for automatic model saving and ReduceLROnPlateau that provides sophisticated learning rate adjustments during convergence.

With the integration of deep learning, class balancing, and cross-validation, our project can establish a stable, scalable, and practicable system for early detection of plant disease. Our system will allow farmers to intervene timely, minimize the loss in the crop, and achieve sustainable agricultural development, thereby providing advantages for world food security and economic stability.

1.2 MOTIVATION

Worldwide need for increased food resources and sick crop vulnerabilities require quick and dependable disease detection methods. Physical plant examinations dominate current operational methods yet these methods demand lots of time and human staff while allowing significant space for mistakes especially because small farmers remain unable to access these techniques in large parts of developing areas. The early onset of plant diseases remains undetected in most cases because they rapidly spread causing considerable agricultural yield losses together with economic damage to farmers and extended interruptions to agricultural distribution networks.

Agricultural disease management receives powerful revolutionary solutions through deep learning technologies for this purpose. The Convolutional Neural Network (CNN) demonstrates excellent abilities to identify essential features from images which leads to automatic real-time disease classification. Such smart systems benefit from smartphone and imaging technology ubiquity to provide farmers even in rural locations with affordable diagnostic solutions for their fields.

The research project develops a powerful deep learning methodology to classify plant diseases by processing images from potato and bell pepper as well as apple plant leaves. Such solution addresses real-world problems related to insufficient data availability and unbalanced class distribution as well as species transfer challenges. A primary collection of 2,152 potato leaf images demonstrates three classes including Potato Early Blight, Late Blight and Healthy status yet suffers from a significant unbalance due to the 152 examples under the healthy category. Through TensorFlow's Image Data Generator we create an augmented image pipeline for dataset enlargement which executes rescaling together with rotation followed by flipping and zooming combined with shifting and shearing for data diversity.

A weighted loss function based on scikit-learn calculations enables better penalization of minority class errors. The model sensitivity regarding underrepresented classes becomes optimal due to this approach. The training model receives its cross-validation by analyzing bell pepper and apple leaf disease samples that include bacterial leaf spot, apple scab, and rust through other data sources for establishing disease pattern recognition across different plant species.

Projecting knowledge from pre-trained CNN models through transfer learning helps speed up the rate of convergence as well as enhance performance when dealing with restricted datasets. The training process for the model depends on three optimization callbacks to achieve stable convergence outcomes while avoiding overfitting issues.

Early disease detection of plants through technological means reduces pesticide overuse hence sustaining agricultural land health and ecological preservation. This solution serves dual purposes for food security and agricultural resilience in addition to protecting crop yields because of climate change and declining biodiversity and land degradation expenses.

The project works to bridge the divide which exists between leading AI technologies and the essential requirements of present-day agriculture. Advanced deep learning methods were established in the plant disease detection system which enables the economical and accessible disease recognition for farmers that protects agricultural outcomes and worldwide food sustainability.

1.3 SCOPE OF PROJECT

Creating a deep learning model stands as the main objective of this work with the goal to establish a reliable and efficient methodology for automatic plant leaf disease recognition. Researchers will direct their project toward identifying plant diseases in potato and bell pepper (capsicum) and apple due to their widespread cultural susceptibility to various diseases. Monitoring diseases during early stages among these crops becomes vital for preserving agricultural sustainability and preventing large-scale crop destruction. The project implements CNNs to perform image-based disease recognition that aims to eliminate manual examination needs along with expert analysis while enabling automated disease symptom recognition.

A training model accesses 2,152 potato leaf pictures divided into three groups: Potato Early Blight, Potato Late Blight and Healthy Potato Leaves. The project brings additional strength to the model through cross-validation by using datasets beyond the original database. The provided dataset includes 2,475 images of bell peppers representing bacterial leaf disease and healthy stages whereas 300 images of apple leaves show Apple Scab, Apple Rust and Healthy Apple Leaves conditions. The project obtained these datasets through the free Kaggle repository because it contains authentic real-world leaf disease patterns.

The main challenge addresses within this project surrounds class imbalance issues particularly with excessive underrepresentation of healthy potato leaves. Class weighting methods implemented through Scikit-learn enables the model to prioritize training on the minority class during learning. Artificial expansion of visual data occurs through widespread application of image data augmentation.

Transfer learning is used in this project by applying CNN models extracted from the image datasets. The models demonstrate optimized training performance since they receive pre-tuned training for the specific plant disease classification objective despite working with a limited dataset. A combination of early stopping callbacks and model checkpoint and reduced learning rate on plateau mechanisms optimizes training process execution.

The project expands its focus to go beyond high accuracy performance while developing an application-ready solution. The project demonstrates its functionality across multiple plant types which makes the model robust enough for web or mobile application deployment within the farming and crop scientific community. Field detection of diseases will be possible through a smartphone camera which can lead to the implementation of this technology in low-resource settings.

The goal of this work establishes a complex disease and multiple crop detection system which adopts deep learning techniques together with cross-validation strategies and data augmentation methods and class balancing protocols and transfer learning standards. Plant health monitoring facilitated by artificial intelligence technology aims to assist agricultural practice through real-world applications under the scope of this study.

Chapter 2
PROJECT DESCRIPTION AND GOALS

2.1 LITERATURE REVIEW

Plant disease detection using deep learning technology has been a central theme in agricultural research because of its automated solutions that are highly precise in the identification of diseases at an early stage. Various research studies examine deep learning models derived from Convolutional Neural Networks and transfer learning principles and object detection algorithms and artificial intelligence classification techniques to improve crop monitoring as well as reduce pesticide use and develop sustainable agriculture. **Sangeetha et al. [1]** suggest a high-end plant disease detection technology that employs deep learning architectures comprising CNN, ResNet-50, VGG16, Inception-V4, and YOLOv7-X. Over 33,000 images of fruits apples and vegetables tomatoes and tubers potatoes constitute the dataset that was sourced from Indian agricultural lands and Tamil Nadu Agricultural University. The maximum detection precision of 99.25% has been realized by YOLOv7-X following the installation of the Bayesian blur enhancement. The system performs transfer learning and feature extraction operations to achieve improved classification performance. Performance measures ensure its effectiveness. By a web-based application farmers are provided with immediate identification assistance that helps them avoid crop damage from taking place.

Biswas and Yadav[2] use CNN-based methods of plant disease detection while pointing out their better performance compared to traditional ML models. Deep CNN attained 99.53% accuracy according to the research while Multilayer CNN and DenseNet201 achieved 99.2% accuracy and EfficientNet reached its highest mark. PlantVillage dataset serves as the primary database in most research systems because it contains more than 50,000 leaf images for plant classification. The accuracy rates are further boosted through the application of transfer learning processes involving INC-VGG and AlexNet fine-tuning methods. The study identifies the effectiveness of CNN for plant disease recognition in computers yet reveals technical challenges during deep architecture computation.

A new real-time image classification system named YOLOv5 forms the central aspect of Amritraj et al.'s plant disease detection system. **[3]** Their research incorporates a custom image collection with 2,000 pictures that cover eight disease classes starting from Anthracnose to Aphids and Cucumber Mosaic Virus. The detection model leverages CNN for obtaining features by employing image augmentations along with size adjustments to boost its precision capabilities. The newer model configuration YOLOv5 shows better performance than its predecessors YOLOv3 and YOLOv4 while also achieving 0.614 mAP along with 130 FPS inference speed. Scientists have shown that object detection techniques successfully identify diseases in crops because they lead to better crop management through automatic precise detection capabilities.

Balwani and Bawane [4] cover smart plant disease detection systems from ML deep learning, IoT, and hyperspectral imaging for real-time monitoring. Tested research utilizes CNN, SVM, Random Forest, and transfer learning for classification accuracies of 98.53% if CNNs are implemented and 97.67% if hybrid CNN-SVM is implemented. Research utilizes IoT sensors for the sensing of temperature, humidity, and soil moisture to be utilized in predictive analytics. Feature extraction methods such as GLCM and LBP enhance the process of classification, and mobile apps enable convenience. The study considers the capabilities of AI-based systems to make crop monitoring more effective, lower the usage of pesticides, and make agriculture more sustainable.

Nagababu et al.[5] propose a deep learning-based plant disease diagnosis system, Convolutional Neural Networks (CNNs), for plant disease identification from leaf images. The data is 30 classes of plant diseases with 80% training, 10% validation, and 10% test. The study targets feature extraction techniques such as leaf color, texture, and damage analysis for improved accuracy. EfficientNetB0-trained model achieved accuracy of 98.59% with validation loss of 0.02%. A web-based interface facilitates real-time identification and provides solution suggestions. The study reveals the prospective capabilities of AI-based systems in increasing agricultural yields, reducing misdiagnosis of diseases, and benefiting farmers through computerized identification of diseases.

According to **Ganesh Reddy et al.[6]** deep learning automation enables detection of plant diseases and produces pesticide recommendations. The research adopts EfficientNetB5 as its foundation to analyze 54,303 labeled photographs coming from the PlantVillage dataset while inspecting 38 disease types across 14 crop species. Model generalization is better achieved through image preprocessing techniques and augmentation methods for reaching improved outcomes in real-world scenarios. The proposed model reaches 96% accuracy surpassing three other systems including AlexNet at 94% and DenseNet101 at 94% and RCNN at 95.48%. The recommended system contains two main structural components including the pesticide recommendation module that links diseases to appropriate treatments. The researchers emphasize transfer learning alongside data augmentation methods because these mechanisms help make the system more advanced while maintaining its capability to scale up. This suggested system delivers automated disease detection through an efficient method that helps sustainable agricultural management.

FarmEasy represents a deep learning-based system for plant disease detection which utilizes VGG16, InceptionV3, and ResNet50 models according to **Pandey et al.[7]**. This research depends on the New Plant Disease data set from Kaggle which contains more than 75,000 pictures spanning 38 disease classes. When utilized for five epochs the ResNet50 model accomplishes 98.05% accuracy which represents the most effective outcome. Users can upload images through the mobile application of the system to receive treatment recommendations along with disease detection results. The application also provides weather forecasting together with crop recommendations and expert advice. The research demonstrates how CNN-based features and real-time disease classification work together as a low-cost system that farmers can operate without difficulty.

Deepa et al.[8] propose a pest and disease detection system in plants with a customized Mask R-CNN for automated classification and segmentation. The model was trained with a plant leaf image dataset and achieved an accuracy of 98%, thereby half the time consumed by traditional approaches in detecting lesions. The study addresses many different methods of detection ranging from visual observation to spectral imaging, sensor-based detection, and machine learning. For performance evaluation, accuracy, precision, recall, F1-score, and AUC-ROC were the metrics employed to compute the efficiency of the model. The system allows for early detection of disease and real-time intervention, thereby enhancing sustainable agriculture and enhanced crop yield.

Yaswanth et al.[9] present a plant disease detection model using transfer learning-based from pre-trained deep neural networks like ResNet50. The study makes use of PlantVillage data present on Kaggle, comprised of 5,000 images representing different types of plants. Preprocessing based on image resizing, normalization, and augmentation facilitates increased model robustness. Model performance with an optimal configuration over 15 epochs and Adam optimizer (learning rate = 0.0001) shows validation accuracy as 91% and testing accuracy as 89%. The study proves the capability of transfer learning in plant disease detection with minimal need for large labeled data and high precision. The model is optimized for mobile and real-time applications, enhancing agricultural sustainability through automatic monitoring of plant health.

The paper of **Singh et al.[10]** presents SubCoPLeD which is a superpixel-based methodology that utilizes color distribution to detect plant diseases. This research used the PlantVillage dataset consisting of 54,309 images which were divided into 38 different plant disease groups. The method implements Simple Linear Iterative Clustering (SLIC) to segment superpixels while threshold-based color classification assigns parts of plant leaves as diseased or non-diseased areas. The system reaches high computational speed without sacrificing its disease detection accuracy. The method achieves successful results in distinguishing healthy from diseased leaves through validation tests performed on the PlantVillage dataset by measuring precision, recall, and F1-score. The research indicates that treatments involving image preprocessing along with segmentation and thresholding operations form essential operations which help prevent and detect crop diseases at their early stages.

Albadani et al. [11] discuss how deep learning (DL) algorithms particularly CNNs and LSTMs and hybrid DL models find extensive use in forest fire detection applications. The article states that CNN-based models excel at extracting spatial feature patterns for fire detection through images but LSTM networks offer the best performance in analyzing temporal patterns across video domains. VGG16, ResNet and InceptionNet are among the pre-trained models which are often combined with transfer learning for fixing dataset restrictions. The paper discusses hybrid techniques which merge CNN and LSTM structures to achieve improved spatiotemporal data analysis. The paper presents an analysis of sensor fusion (thermal, smoke, gas) combined with AI models. Model evaluation typically uses accuracy alongside precision and recall metrics together with F1-score but each assessment generates dataset-dependent outcomes according to the literature.

Shaik et al. [12] propose a deep learning system which merges Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks to achieve better apple leaf disease classification. The method relies on CNN layers for spatial feature extraction while LSTM units handle the feature sequence temporal patterns. The research uses 1,800 publicly available apple leaf images that belong to the three disease classes which include healthy leaves and scab and rust pathogens. The purpose of image augmentation techniques is both to fix data imbalance issues and improve model generalization capabilities. The combination of CNN-LSTM enables superior performance according to accuracy, precision, recall and F1-score metrics which establishes this approach as an effective method for early disease detection in precision agriculture.

[13] A hybrid CNN-LSTM deep learning model which Sandeep et al. developed serves to classify and determine the severity degree of rice sheath rot disease. The spatial features of rice leaf images are extracted through CNN while the LSTM network detects patterns in order to classify disease severity into mild moderate and severe categories. A specifically prepared rice sheath rot dataset contains severity class annotations while image preprocessing and augmentation help the model achieve better generalization. Numerous experimental tests indicate that the combination of CNN and LSTM returns greater accuracy as well as robustness compared to standard CNN systems and demonstrates significant potential for precision agriculture disease severity classification purposes.

The CNN-LSTM model achieved better results but its testing primarily involved a balanced dataset that limits its practicality for real-world agricultural applications according to **Karuna et al. [14]** The proposed model does not manage dynamic environmental factors including illumination variations together with occluding elements and leaf interpositions that commonly exist in agricultural fields. Real-world application deployment of this technology is limited due to two significant factors: no live field testing took place and the required stationary datasets. Future studies must unify diverse datasets and add environmental noise to the model and test this model in authentic agricultural field conditions so researchers achieve better application of the model.

Sandeep et al.[15] developed a CNN-LSTM model to detect rice sheath rot severity grades which were labeled as mild, moderate and severe. The spatial features of the images were processed by CNN yet LSTM performed temporal pattern analysis to achieve improved classification results. The detection and classification of Tomato-Spotted Wilt Virus (TSWV) intensity at six different levels is described in Sharma et al.'s CNN-LSTM ensemble. The proposed model achieved 97.73% accuracy when using image sequences and VGG16 for feature extraction together with LSTM for temporal modeling and performed better than existing solutions.

Bhargavi et al.[16] present a multi-method solution that ties Convolutional Neural Networks (CNN) with Recurrent Neural Networks (RNN) to analyze combined RGB and hyperspectral images which detect cotton plant diseases. The findings demonstrate better identification results that emerge when spatial characteristics combine with sequential data attributes. The proposed architecture reaches higher accuracy levels on benchmark datasets as it addresses weaknesses in systems utilizing one modality.

Sharma et al [17] develops a diagnostic system using deep learning technology that applies MRI results with Clinical Dementia Rating (CDR) scores to stage Alzheimer's disease. Their model relies on Convolutional Neural Network (CNN) architecture built for quick MRI scan processing alongside CDR score as clinical reference point used for multiclass classification tasks. The system adopts an efficient diagnostic solution through feature extraction integration with classification features which achieve low computational requirements. The research focuses on multi-class classification for different stages of AD (non-demented, very mild, mild, and moderate) along with an optimal precision-run time compromise. Available public databases support tests which show that the model identifies various disease stages while maintaining high precision and limited overfitting in comparison to traditional approaches.

2.2 GAPS IDENTIFIED

Deep Learning & Machine Learning Models: Deep models of learning are now used in plant disease diagnosis but have extremely limited practical applications within the field because of constraint datasets, computation inefficiencies, and environmental validity issues.

- Sangeetha et al.: Higher accuracy but without cross-validation for varied environmental conditions. Crop-specific dataset results in limited scalability. Multimodal data (hyperspectral imaging, e.g.) has not been validated. YOLOv7-X performed well but is oblivious to its performance on low-power devices. Future work will be to improve datasets, cross-validate between climates, and model-optimize for real-time use.
- Biswas and Yadav: Ranked lack of diversity in real-world dataset and computational expense of deep CNNs as major challenges. PlantVillage dataset is general but doesn't have variations in light, background, and environment, constraining model generalizability. Deep learning model's computational inefficiency constrains real-time applications. Future studies will need to expand datasets, minimize light-weight models, and utilize hyperspectral imaging for improved field application.
- Amritraj et al.: Database is restricted to 2,000 images, impacting generalizability across various environments. The actual farm images were not validated on multi-lighting images. YOLOv5 performs well but has extremely high computational requirements, which is limiting for deployment on low-power edge devices. Diversity in the dataset, mobile model optimizations for mobile use, and hyperspectral imaging would be implemented in the future to classify disease more effectively.
- Balwani and Bawane: Explored AI-plant disease recognition with CNN, SVM, Random Forest, and IoT sensors. CNN models achieved accuracy of 98.53%, while CNN-SVM hybrid models achieved an accuracy of 97.67%. IoT monitoring of temperature, humidity, and soil moisture improves predictive analytics. Feature extraction methodologies like GLCM and LBP enhance classification. Mobile applications ensure ease of accessibility, but deployment and real-time scalability remain issues.
- Nagababu et al.: Recognized dataset limitations resulting from the lack of field data reflecting actual environmental changes. EfficientNetB0 is a computation-intensive, efficient model, and thus not ideal for mobile deployment. Future work needs to be towards lightweight models, expanding the datasets, and real-time mobile-supportive deployment.

- Ganesh Reddy et al.: Focused on constrained environmental diversity in datasets, limiting real-world use. The study was centered on image-based classification and did not take into account multispectral/hyperspectral imaging to enhance detection. Computational complexity of EfficientNetB5 will make deployment on low-power devices challenging. Future work has to increase dataset diversity, delve into lightweight models, and incorporate real-time monitoring for precision agriculture.
- Pandey et al.: The paper solely deals with image classification and does not address environmental conditions impacting plant health. Computational-intensive ResNet50 can have limited deployment to low-end devices. Future research must concentrate on lightweight models, real-world testing on datasets, and multispectral imaging for increased accuracy.
- Deepa et al.: They lacked verification of the dataset in various conditions. Mask R-CNN has prohibitive computational requirements, thus curtailing its usage on smartphone or low-resource hardware. Deployment using light models in real time under multispectral settings is a way that calls for more priority toward increased scalability and efficiency.
- Yaswanth et al.: Identified dataset diversity constraints, impacting the generalization of the model. The model performs low on visually related diseases, resulting in misclassification. It decreases by 31% on actual-world images, which establishes the requirement for diverse datasets and domain fine-tuning. Edge computing, real-time deployment, and multispectral imaging have to be the direction of future research.
- Singh et al.: Recognized that SubCoPLeD is strongly dependent on color distribution, resulting in misclassification in cluttered backgrounds. Absence of deep learning models hinders adaptation to multiple lighting and environmental conditions. Minimizing real-world field validation weakens practical applicability. Development in the future must focus on hybrid deep learning models, spectral imaging, and field testing for enhanced accuracy.
- Albadani et al. examined many deep learning detection models yet most of them relied on simulated data which creates limitations for real-world usage. The lack of heterogeneous datasets which are labeled reduces the ability of models to operate with a wide range of forest conditions and elements. Visual data represents the main focus of most techniques while omitting necessary environmental elements like temperature and humidity from analysis. The lack of light models makes it difficult to run operations on edge devices. Research focused on using real-life datasets must integrate multi-type inputs and deployable solutions designed for distant locations.

- Shaik et al. state that current approaches use either traditional machine learning and standalone CNN models that fail to integrate image feature sequential relationships. The performance of their CNN-LSTM model shows superior results but their assessment takes place on a reduced dataset with limited practical agricultural implementation scope. The system fails to entirely resolve lighting conditions in addition to leaf position variations and background disturbances that might compromise operational stability in agricultural fields. Additional research needs to test merged network designs across wider and more diverse datasets and should include real-time field verification according to the authors' proposal.
- Research from Karuna introduces an effective plant leaf disease classification system that uses deep learning techniques which integrate Convolutional Neural Networks (CNN) and Long Short-Term Memory (LSTM) networks. Spatial image feature extraction functions belong to CNN layers and sequential dependencies in data extraction falls within the responsibility of LSTM layers. The research team used the PlantVillage dataset available to the public which contains different leaf categories between diseased and healthy plants. By applying preprocessing operations and augmentation methods the model improves feature quality as well as enhances model robustness. The CNN-LSTM model structure demonstrates superior classification performance to regular CNN networks based on accuracy and precision and recall and F1-score metrics which proves its potential to improve smart agriculture technologies.
- The authors of the CNN-LSTM model state that their enhanced performance outcome remains restricted for challenging real-world applications because testing occurred with controlled datasets. The research lacks thorough evaluation of natural environmental elements that affect image quality such as illumination problems and leaf obstruction and background disturbances. The model does not undergo testing for real-time field operational application. The reliability and realtime operational feasibility of this system needs to be evaluated by testing with diverse datasets along with environmental noise testing before applying it to real agricultural areas.
- According to Sharma et al. their ensemble model with CNN-LSTM demonstrates high accuracy for identifying and categorizing Tomato-Spotted Wilt Virus (TSWV) intensity levels but these tests use controlled data that went through cleaning processes. The model shows limitations for field implementation because it depends on high-quality images acquired under standardized testing circumstances. This study fails to investigate how the model can be deployed in real-time field implementations. Research needs to develop the model through testing it with unstructured datasets from diverse agricultural locations

and minimizing computational requirements for mobile-edge applications in actual agricultural fields.

- The research focuses exclusively on cotton crop diseases but cannot be used for other types of crops. The model complexity creates deployment challenges when trying to use it on mobile or edge devices. The system requires developed models which weigh less and possess more adaptability to expand its practical use across multiple plant varieties in different environmental settings.
- Sharma et al. acknowledge their model shows better performance yet they have conducted no dataset validation across different groups. The current model architecture mainly performs binary classification but lacks the ability to distinguish between discrimination levels based on CDR scores therefore limiting its practical medical use. Virtual resources required by most deep learning methods exceeds appropriate levels for usage in real-time operations or environments with limited computing capacity. Further research should focus on model generalization and strong multi-class classification of light-weight architectural models with testing on extensive heterogeneous datasets.

2.3 OBJECTIVES

- In order to contribute towards food security by enabling early and accurate detection of plant diseases so that farmers may take immediate preventive action, reduce losses, and increase agricultural yield.
- To develop a robust deep learning-based plant disease classification system using Convolutional Neural Networks (CNNs) that classifies disease categories accurately from high-resolution images of plant leaves.
- To address class imbalance for the primary potato dataset (Healthy, Late Blight, and Early Blight classes) using class weighting techniques with scikit-learn to provide fair and balanced predictions across all categories.
- To enhance model flexibility and generalizability by incorporating cross-validation with bell pepper and apple leaves datasets so that the system learns the pattern of diseases in multiple crop species rather than just for potatoes.
- To use and compare several deep learning models, e.g., EfficientNetB3, ResNet50, VGG16, MobileNetV2, and LSTM, and establish the most precise and computationally effective architecture for plant disease classification.

- To optimize the model performance and robustness through techniques such as transfer learning (using pre-trained networks), advanced image augmentation (resizing, rotation, shift, flip, zoom, shear), and automated feature extraction.
- In order to enable the system to be used in real-time, such that quick and precise disease detection results can be implemented feasibly in agricultural settings.
- In order to develop a light and mobile-optimized model that can be installed on smartphones and edge devices, so it is more easily accessible to farmers in remote areas with limited infrastructure.
- To offer an affordable, scalable, and user-focused solution that aids in the timely detection of plant diseases at their initial stages, supports sustainable agriculture, prevents wasteful pesticide use, and supports long-term agricultural resilience.

2.4 PROBLEM STATEMENT

Plants diseases remain among the primary risks that damage the yield and quality of agricultural production while agriculture ensures food security and economic stability worldwide. The current method of disease detection depends entirely on farmers and agricultural specialists doing manual observations which proves both time-consuming and error-prone because subjective evaluations by humans can be incorrect. Plant diseases prove hard to detect at a proper time point because environmental conditions along with crop variety and comparable symptoms present hurdles for accurate diagnosis.

A disease detection system requires urgency because smallholder farmers struggle to obtain specialist advice in a timely manner. Early disease detection becomes crucial because delay decreases productivity and creates vast economic losses besides damaging the environment because farmers apply excessive pesticides to manage their diseases.

Various weaknesses prevent deep learning systems from achieving automated plant disease diagnosis through image classification although the method shows initial success. The application of skewed datasets with insufficient healthy leaf examples along with specific model requirements and excessive computational needs makes these systems unusable in areas with limited resources.

A high-resolution leaf image analysis system utilizing deep learning methods aims to address these issues for disease identification in potato along with bell pepper and apple crops. The main dataset consists of 2152 images that show potato leaf characteristics ranging from Early Blight to Late Blight along with the majority of images belonging to the Healthy class category. Class weighting procedures will be deployed to maintain equivalent treatment of all classes in the training process.

The system implements cross-validation analysis on 2,475 bell pepper leaf images which contain bacterial leaf disease and healthy images and 300 apple leaf images with apple scab and rust along with healthy examples. A complex image augmentation system using rescaling, flipping and shifting, zooming, rotation and shearing capabilities will optimize data diversity for overfitting prevention.

Different deep learning models including EfficientNetB3 and ResNet50 and VGG16 along with MobileNetV2 and LSTM will undergo training then comparison to establish which model delivers exceptional accuracy among minimal computation cost. Through transfer learning alongside pre-trained network feature extraction the system improves both its speed of convergence and performance results when dealing with limited datasets.

The detection model will reach optimal performance for detecting diseases in real-time which can be deployed on edge devices for rural locations. The implemention of advanced AI technology targets a system that costs less and scales well and is applicable for general plant disease detection which gives farmers early diagnostic tools for sustainable farming practices and better food supply security.

2.5 PROJECT PLAN

Phase 1: Research & Requirement Analysis (December 13 – December 31, 2024)

Researchers must analyze the present plant disease detection methods to recognize existing obstacles which include restricted generalization and unequal data distribution and inadequate computing capabilities. The presented findings allow conclusion of the problem statement and objectives. A set of Potato leaf image data including Early Blight, Late Blight, Healthy categories forms the dataset while apple and bell pepper leaf datasets serve as the cross-validation test. The model gains better ability to apply its learning to various plant taxa through this enhancement. Furthermore the system requires the implementation of accuracy, precision, recall, and F1-score as performance measurements standards.

Phase 2: Dataset Preparation & Preprocessing (January 1 – January 15, 2025)

The gathered data goes through the labeling and cleaning process so it is prepared for model training. All image preprocessing steps include normalization together with rescaling and contrast enhancement tasks. Data augmentation through rotation and flipping and zooming techniques provides solutions to class imbalance problems in the dataset. The data collection contains photos from potato together with apple and bell pepper leaves which get split into three parts for training (80%) and validation (10%) and testing (10%).

Phase 3: Model Selection & Implementation (January 16 – February 10, 2025)

A selection of EfficientNetB3, ResNet50, VGG16 and MobileNetV2 and LSTM models will be implemented for the application. Training occurs on the preprocessed dataset by implementing both transfer learning and feature extraction methods. The best accuracy performance along with efficiency and computational cost can be obtained through hyper parameter tuning. A systematic approach is followed to choose the most suitable model that will detect multiple crop diseases.

Phase 4: Model Evaluation & Cross-Validation (February 11 – March 1, 2025)

Accuracy measures together with confusion matrix and loss and precision-recall curves determine the value of trained models. Models with the minimum error rates become the selected best performer. The crucial step includes cross-validation between apple and bell pepper datasets to demonstrate model abilities for generalization across different plant species to ensure adaptation to real agricultural conditions.

Phase 5: System Development & Integration (March 2 – March 20, 2025)

The implemented mobile or web application runs the chosen model which was previously selected as the best one. Through its user-friendly interface farmers may submit leaf pictures which get instant feedback from the system. The model operates effectively on equipment with limited resources to serve farmer populations in areas with poor internet and rural regions. The implementation process fulfills the need to bridge scientific research with practical application.

Phase 6: Testing & Final Deployment (March 21 – April 10, 2025)

The system is evaluated using genuine images collected from diverse environmental settings to represent actual operation conditions. Scientists analyze three performance metrics that include speed and efficiency together with usability. The model receives tuning to address faulty predictions and obtain better reliability throughout it. All documentation reaches completion with sections for technical details and user requirements and deployment necessities.

Phase 7: Final Submission & Presentation (April 15 – April 25, 2025)

The package delivery stage includes the final distribution of source code along with trained models and documentation reports and the actual project system materials. A formal presentation with a live system demonstration showcases the potential uses and practical applications as well as potential agricultural field deployment of the system.

GANTT CHART

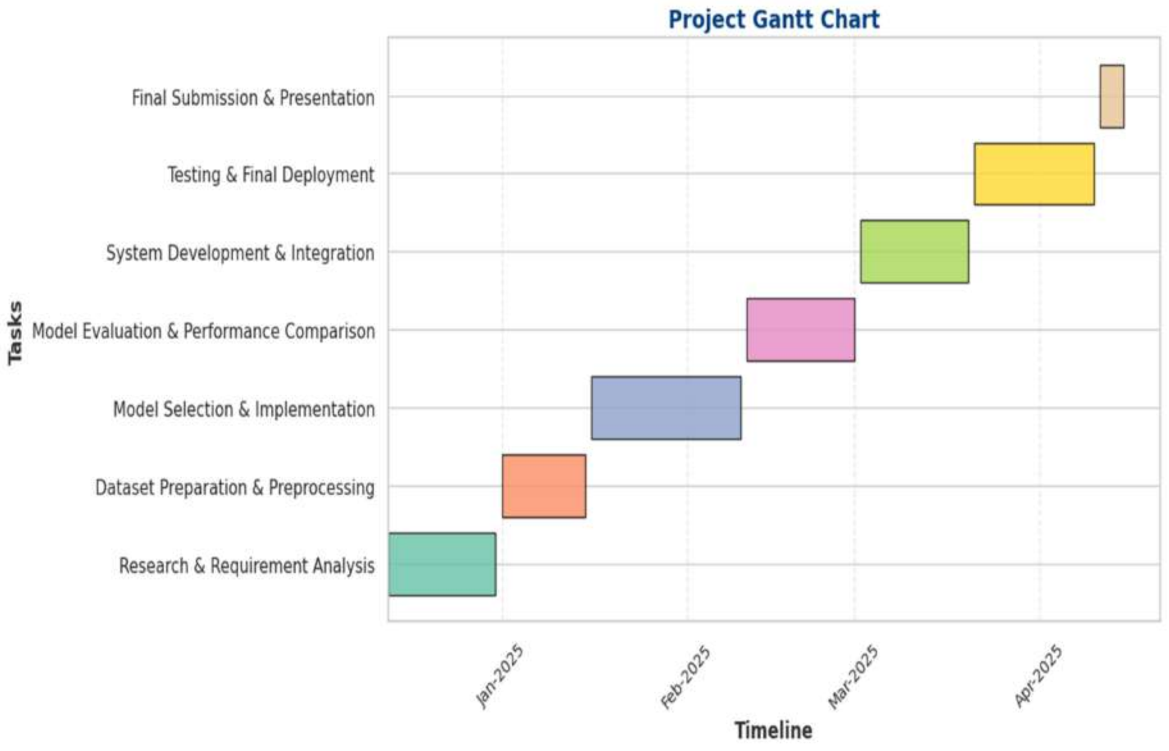


Fig 2.5.1 Gantt Chart

Chapter 3

TECHNICAL SPECIFICATION

3.1 REQUIREMENT ANALYSIS

3.1.1 Functional Requirement

The plant disease classification system requires functional requirements to define essential operational features together with system interactions and necessary features for use. The model maintains correct behavior and predictable functionality in real agricultural settings because of its properties which also support accessibility to expert and non-expert users during application use. The system needs more than precise identification requirements to succeed because users require easy-to-use workflows for result delivery and model processing combined with easy data upload.

1. Image Input and Upload Interface

A user-friendly interface must exist to enable simple and reliable processing of leaf pictures from potato and bell pepper and apple plants. Users must interact with this feature first before proceeding to other system elements. The system should provide dual features that allow single and batch image uploads because it caters to the requirements of individual farmers along with researchers who process many images at once. The system should perform automatic preprocessing of uploaded images additionally validate them before model requirements so non-technical users find the system simple to use while preventing manual interruptions.

2. Image Preprocessing and Augmentation

Image processing with rescaling and normalization and resizing steps occurs before the model receives the data input. These conversion steps confirm that inputs stay within predefined range of pixel intensities as well as matching specified dimensions for producing consistent model outputs. During training the system needs to perform realtime data augmentation so it can expand the training dataset diversity. The augmentation approaches need to include random flipping and zooming and rotation and shearing and shifting operations to replicate regular variations in lighting and background and viewing angle. The preprocessing module requires the ability to detect both corrupted and unsupported files before generating proper error responses to keep the model shielded from invalid inputs.

3. Multi-class Disease Classification

The central model needs to achieve precise disease categorization of leaf images among eight classes which include Potato Early Blight, Potato Late Blight, Potato Healthy, Bell Pepper Bacterial Spot, Bell Pepper Healthy, Apple Scab, Apple Rust, and Apple Healthy. The system needs to generate categorized results that contain self-assurance metrics which demonstrate prediction certainty. All predictions must be displayed in an obvious manner throughout the interface.

4.Transfer Learning Integration

The solution needs to implement transfer learning capabilities using pre-trained models including VGG16, ResNet50 and EfficientNet. The system requires flexibility to adjust between base layer freezing and finetuning top layer operations. The approach cuts down training duration by adopting knowledge from the extensive ImageNet dataset which enables better version performance despite working with small agricultural datasets.

5. Class Imbalance Handling

Due to the imbalanced nature of the healthy potato leaves the system requires calculation of appropriate class weights. The correct learning of minority classes requires this approach to be present for optimal performance. Class weights need automatic computation using Scikit-learn followed by their input to the model during training.

6.Training with Callback Mechanisms

A strong set of training callbacks should be used including Early Stopping to prevent overfitting together with Model Checkpoint for saving the best model and ReduceLROnPlateau that decreases the learning rate upon stagnation. The mechanisms create stable models while reducing resource waste and improving generalization of your models.

7. Cross-validation with Different Crops

Cross-validation through different plant datasets from Apple alongside Bell Pepper should be enabled in the system because this improves model generalization for various plant species. A performance assessment of every classification requires calculating accuracy alongside precision and recall and F1-score during cross-validation processes.

8. Batch Prediction Support

There must be functionality in the system which enables researchers or agricultural officers to upload multiple images at the same time. The predictions should be displayed in a table format which includes image previews in addition to predicted labels and confidence scores.

9.Model Evaluation Metrics and Visualization

Users should see visual feedback throughout training session and after the completion of training. The display combines confusion matrices together with loss and accuracy plots as well as classification reports and ROC curves. These graphics help interpret model performance most notably in unbalanced datasets and constitute necessary components for achieving transparency.

10.User Interface and Usability

Users should experience a smooth front-end process which guides them step by step through image upload along with disease analysis and interpretation of results. The presentation of results should occur promptly using visual indicators which differentiate normal from abnormal outcomes through color codes or symbolic elements.

11.Secure Data Handling and Privacy

The images submitted to the server must remain protected during analysis before unknown images fully delete from storage unless the user opts to save them manually. User information remains inaccessible to the system since it does not require storage or disclosure of individual details. The system requires compliance with basic data privacy guidelines (such as GDPR) in case of online deployment.

12.Model Retraining and Update Support

A system should enable administrators to modify the dataset while allowing model retraining followed by better version deployment which should not interrupt user interaction. The system needs pipelines which are reusable along with capabilities to save and version new models deployed in the system.

13.Extensibility to Other Crops and Diseases

The system architecture must include modular features which enable administrators to introduce new plant types along with new disease classes within the future. New classes need to be added into the system without requiring complete model reconstruction because the existing architecture along with preprocessing components can be modified and reused for new classes.

14.Logging and Monitoring

Pre-production measurements including prediction results, detection of errors and documentations of feedback need proper storage that ensures continuous monitoring and performance optimization. The logging system enables administrators to search for patterns in model failures which help them perfect model performance through time.

3.1.2 Non functional requirement

The plant disease classification system needs non-functional requirements to meet the necessary quality characteristics along with restrictions and requirements that determine its reliability and maintainability and efficiency and usability. The application requires these fundamental yet non-core requirements to achieve successful deployment even though they do not directly impact its main operational features.

1. Performance Requirements

- A prediction system must provide output responses within 2–3 seconds when processing a single image which uploads to the system.
- When handling batch image upload tasks the system needs to operate efficiently by utilizing parallel processing or batch operations in order to maintain an acceptable response speed.

- The maximum time allowed for model inference should remain below pre-defined resource limits when operating in mobile and web environments.

2. Usability

- Both professionals and non-professionals must find the interface easy and efficient to use as the system targets farmers and agricultural officers.
- Every part of the system interface should display information which is written using plain language without using technical jargon.
- The system requires multilingual capabilities because its target markets consist mainly of rural areas including international fields.
- Tooltips or user sessions will ease the onboarding process for users who need to upload content and understand their results.

3. Reliability and Availability

- The implementation system requires at least 99.9% operational availability for agricultural peak seasons.
- The system includes backup solutions through error messages combined with retry functionalities which activate when model failures or prediction errors occur.
- The system design must have robust security measures against system crashes triggered by unexpected input data and degraded input data.

4. Scalability

- The design must have sufficient versatility to handle rising user traffic along with the expanding data during its operational expansion.
- New crops together with disease types and improved machine learning models should enter the system easily through a design that prevents extensive modifications.
- To accommodate forthcoming demand organizations need to consider implementing horizontal scaling solutions through both load balancers together with cloud infrastructure.

5. Maintainability

- The codebase requires modular programming as its foundational principle to simplify debugging procedures while enabling both system upgrade and maintenance.
- The system requires comprehensive documentation of developer resources which includes API documentation together with data pipeline specifications and model architecture information.
- The application needs versioning capabilities to track datasets and models to keep records for auditing and reproduce results.

6. Portability

- The system needs to function across all modern web browsers together with automated mobile devices and tablet systems.
- The model should be prepared for offline usage by retaining its exportability through Tensor Flow Lite for edge implementation.

7. Security

- Data encryption through HTTPS should protect each image upload process and users receive storage security alongside prediction-generated automatic deletion options to ensure their privacy.
- The system should use anonymous methods to process personal information obtained through user feedback and logs.

8. Compliance

- When operating in jurisdictions with data privacy laws such as GDPR the system needs to fulfill requirements by letting users request access to their stored data and demand erasure of their images.
- The proposed system needs to deliver clear visibility about the data processing steps and storage methods the user data undergoes.

9. Accuracy and Robustness

- Technical systems require at least 90% precision performance when processing diverse input images together with different illumination setups and variations in visual quality.
- The prediction pipeline must demonstrate strong resistance against partial leaves combined with multiple leaves together with small background noise variations.

10. Backup and Recovery

A backup system needs to exist for automated prediction logs as well as model checkpoints and datasets when performed on a regular basis.

If any system failure occurs the system should restore fast operations through mechanisms which protect all stored data.

3.2 FEASIBILITY STUDY

3.2.1 Technical feasibility

A plant disease classification system's technical feasibility shows its ability to succeed with current available technology infrastructure when implementing and deploying the proposed design. A system that employs deep learning while processing images through transfer learning to detect diseases in potato and bell pepper and apple leaves exists in current technology. The project stands as technically viable due to the current market tools and available technological solutions.

1. Availability of High-Quality Datasets

Reputable data source Kaggle provides the publicly accessible information for this project. The leaf image collection consists of a wide selection of pictures which display numerous diseases affecting potato plants alongside diseases affecting both apple and bell pepper plants. The image data contains realistic labels and sufficient numbers which ensures proper training of the model. The dataset includes class imbalance particularly affecting healthy potato leaf images but this issue becomes manageable through class weighting and augmentation techniques.

2. Data Augmentation and Preprocessing Tools

The Image Data Generator class native functionality in Tensor Flow/Keras enables easy implementation of real-time preprocessing together with augmentation for images. The model benefits from data enhancement and generalization capabilities through artificial dataset expansion which results from rotation, zooming and shifting, shearing and horizontal flipping operations. The accessible and well-document tools simplify the entire implementation process.

3. Deep Learning and Transfer Learning Models

Earlier developed CNNs benefit from transfer learning methods with pre-trained models consisting of VGG16 or ResNet or EfficientNet. These image classification models perform well in their field and Tensor Flow and PyTorch platforms make them available for developers. With transfer learning capabilities the system operates efficiently using prelearned features from substantial image collections no matter how restricted its available data .

4. Computing Environment

The model system functions on several types of hardware including high-end laptops as well as cloud GPUs through platforms like Google Colab, AWS and Azure and local machines equipped with CUDA-capable GPUs. Cloud platforms provide users a solution to enhance training speed for large datasets and resource-intensive activities. During training the implementation includes two callbacks namely Early Stopping and Model Checkpoint which protects against overfitting and automatically selects the best model achieved thus maximizing resource efficiency.

5. Cross-Validation Support

Multiple plant datasets including apple and bell pepper are supported for conducting cross-validation within the framework which develops more resilient trained models. The technical preservation of the model occurs through this process which protects it from developing a preference for specific crop types so it can effectively analyze diverse data sets. Through implementation of cross-validation the system becomes able to apply its knowledge toward new data beyond its training samples.

6. Deployment Options

A web application based on Flask Django or Streamlit can host the final developed model. The platforms enable straightforward implementation of user interfaces that allow both image uploading and prediction viewing capabilities. The cloud deployment approach which runs on AWS or Heroku offers both scalability features with easy accessibility. The exported model allows users to interface with Application Programming Interfaces for achieving high-speed and efficient prediction deployment.

7. Model Monitoring and Logging

Technical feasibility leads to implementing systems that track how well the model performs during actual use. System logs for user uploads predictions and feedback allow the automation system to learn continuously. User feedback gathered by the system feeds into model retraining routines or fine-tuning processes that operate on a regular basis.

3.2.2 Economic feasibility:

The economic viability of the plant disease classification project secures technical feasibility and cost-effectiveness in development while simultaneously creating cost-efficient deployment and maintenance procedures. Development costs represent the main considerations in this project analysis.

1. Development Costs

- The project selects free Kaggle datasets for data acquisition purposes which keeps project expenses low. In case proprietary datasets become necessary in future development additional financial cost will need to be incurred.
- The development of the project utilized open-source tools Tensor Flow/Keras alongside Python libraries through the free GPU resources available on Google Colab.
- Machine learning engineers and data scientists together with developers make up the human resources requirements for this project. Employing temporary staff members through contracts along with freelance personnel helps organizations to minimize their expenses. Three developers make up an adequate team for launching the first stages of development.
- The time needed to develop a simple prototype should be manageable within a few-month period yet implementing a fully scalable solution requires 6-12 months depending on the sophistication of the model design process.

2. Operational Costs

- Cloud Infrastructure through AWS and Google Cloud offers billing by consumption to control startup expenses during low operation volume. The evolution of the project will lead to increased cloud expenses but strategic resource utilization strategies will help control these costs.
- The system demands periodic maintenance steps involving updates and retraining and debugging procedures. The model maintenance procedures will occur rarely since the initial optimization phase worked correctly.
- Data storage costs remain low because compressed image formats are used when stored along with maintenance of essential data.
- A mix of user support expenses exists for farmers yet proper design along with detailed documentation minimizes these costs.

3. Potential Benefits and Revenue

- The farming industry of regions containing substantial potato and bell pepper and apple farms represents an extensive market potential. Early detection helps farmers in reducing pesticide use and decrease cost of farming.
- The farming system guides farmers to deliver precise interventions which enables them to minimize pesticide costs while boosting their yield levels to gain economic advantages immediately.
- The system enables expansion to monitor more farming crops which will grow its market capability. Recurring charges and paid features within the system dynamic will enable extra income sources.
- Government bodies together with NGOs can provide support to this project because its food security enhancement capability creates new funding prospects and collaboration openings.
- Robert's profit margin from the system investment will be positive if agricultural benefits surpass all expenses particularly when raising harvest quantities through reduced operational costs.

4. Risk Considerations

Growing the system may result in costly operational problems associated with infrastructure expansion and performance maintenance. Cloud platform-based scalable solutions enable the resolution of these problems. The lack of familiarity with AI-based systems among farmers allows them time to consider implementation. Proper user education helps control the rate of return through the rate of adoption.

Conclusion

Plant disease classification represents a lucrative business model since it requires small tooling and data investments and benefits from cloud infrastructure scalability. The system's development and operational expenses along with maintenance costs are manageable and the potential market need combined with farmer cost reductions ensure profitable return on investment. The system provides profitable returns for agricultural enterprises when more crops and diseases are integrated into the framework.

3.2.2 Social feasibilty

Social viability defines how well the plant disease classification system can transform society especially in agricultural communities. The system analyzes accessibility together with social value and long-term yield aspects for farmers as well as environmental sustainability.

1. Effect on Agricultural Societies

- The direct users of the system include farmers, particularly those growing potato, bell pepper, and apple plants. The system offers several social benefits to humanity through its operations.
- The timely recognition of diseases enables farmers to respond quickly thus restricting the dispersion of potato blight and apple scab and bell pepper bacterial leaf spot. Better crop returns together with enhanced food security become possible through this approach.
- Farmers who cultivate small-scale operations experience increased productivity when they detect diseases precisely when they emerge due to this system because it allows them to deliver swift treatments while preventing losses.
- Accurate disease identification provided by the system helps decrease farmers' need to apply general pesticides. Through such identification measures environmental damage together with pesticide usage can both be lowered.

2. Accessibility and Inclusivity

- The system extends access to farmers who live in rural as well as remote locations.
- The tool features a basic interface that allows simple operation between image upload and final results display to accommodate farmers without technical skills.
- The deployment costs remain affordable because the tool utilizes open-source applications together with cloud hosting benefits all farmers including those with limited resources.

- The tool enables widespread accessibility by providing multilingual system support that lets farmers belonging to different linguistic communities and geographical areas interact with it.

3. Education and Empowerment

- The system helps farmers to acquire disease diagnosis knowledge along with empowerment capabilities:
- The system performs disease diagnosis while providing helpful content for plant disease management to improve farmer training capabilities.
- Training programs that connect with agricultural organizations will run educational sessions for farmers to teach them system operation for better decision-making outcomes.

4. Environmental and Health Benefits

- The system achieves essential health benefits in addition to vital environmental gains through its accurate disease management protocols.
- The system decreases pesticide usage to automatically decrease soil contamination along with both water and air pollution that leads to sustainable agricultural practices.
- The decrease in pesticide exposure enables crops to maintain better health thereby reducing the risk of pesticide contamination in consumed fruits and vegetables.

5. Social Acceptance and Adoption

- Success of this system depends on the acceptance level among farmers.
- Farmers must have absolute confidence in system functionality since they require dependable results to adopt the system on a widespread basis. It is essential to build credibility through collaboration and physical testing of system performance.
- A few of the farmers show reluctance toward accepting modern technological advancements. For the system to succeed farmers must be presented with practical demonstrations of its value while receiving educational materials and outreach programs to alleviate skepticism.

6. Long-Term Societal Benefits

The long-term societal impact includes:

- The system improves food security primarily in agricultural areas since it boosts crop production levels.
- Sustainability comes from a system which limits chemical pesticide application and teaches environmentally friendly disease control practices.

Conclusion

The multi classification system of plant diseases demonstrates outstanding feasibility when applied to human society. The system focuses on essential matters of crop wellness together with sustainability and food security guarantees. The system presents farmers with an important tool that delivers both success to growers and benefits the entire agricultural population through its accessible structure and education potential and environmental benefits.

3.3 SYSTEM SPECIFICATION

3.3.1 Hardware specification

1. Development & Training Phase

This phase involves training deep learning models with CNN and transfer learning, which demands heavy hardware:

- Processor: Intel Core i7 / AMD Ryzen 7
- RAM: 16 GB or more
- Storage: 512 GB SSD
- GPU: NVIDIA GTX 1660 Ti (6 GB VRAM) or higher
- Operating System: Ubuntu 20.04 / Windows 10 / macOS
- Other: Full HD display, stable internet, and UPS backup

2. Deployment & Inference Phase

This phase involves executing the trained model for predictions and is less processor-intensive:

- Processor: Intel Core i5 or equivalent
- RAM: 8 GB
- Storage: 256 GB SSD
- GPU (optional): For quick predictions
- OS: Windows / Linux / macOS
- Web Framework Support: Flask, Django, or FastAPI

3. Client Access (Mobile/Web)

End users can view predictions via a browser or mobile client. Requirements are low:

- Device: Smartphone / Tablet / Laptop
- Browser: Chrome, Firefox, Safari (current versions)
- Internet: Reliable Wi-Fi or cellular data connection
- OS: Android, iOS, Windows, or macOS

3.3.2 Software specification

1. Operating System

- Development & Training: Ubuntu 20.04 LTS / Windows 10 / macOS (latest)
- Deployment: Linux-based server preferred for stability and compatibility

2. Programming Languages

- Python 3.8+ – Core language for model development and scripting

3. Development Tools & IDEs

- Jupyter Notebook and VS Code – For coding and experiments
- Anaconda – For managing the environment and GPU

4. Libraries & Frameworks

- TensorFlow / Keras / PyTorch – Building deep learning models
- OpenCV – Preprocessing images
- Pandas, NumPy – Data management
- Matplotlib / Seaborn – Visualization
- Scikit-learn – ML models, preprocessing, and evaluation

Chapter 4
SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

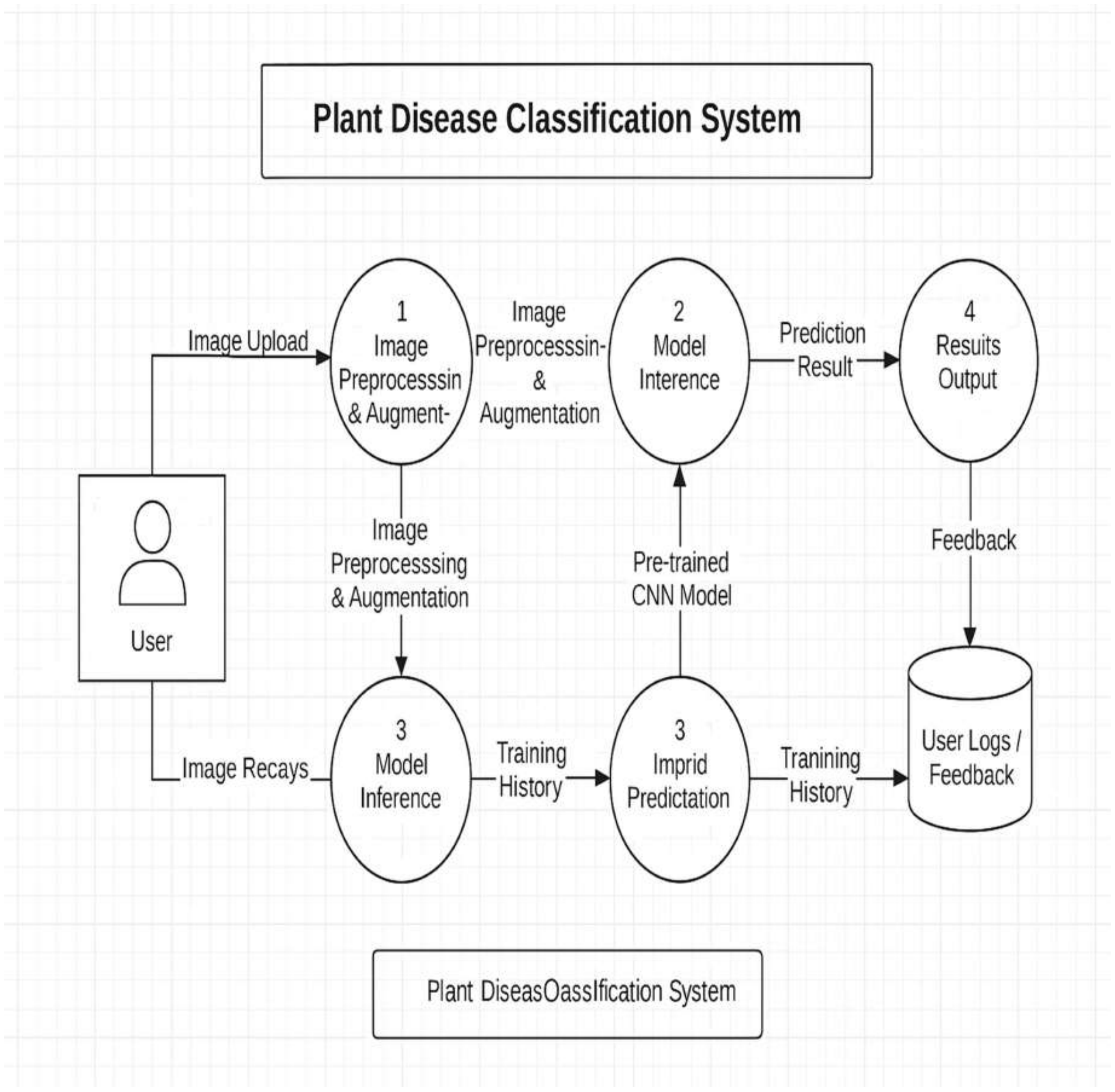


Fig 4.1 System Architecture

4.2 DESIGN

4.2.1 Data Flow Diagram

The design illustrates the complete process to detect plant diseases through deep learning algorithms. The first step involves obtaining datasets by collecting images from Kaggle for the three agricultural crops including potato and bell pepper and apple. The preprocessing stage executes two processes namely data augmentation through rotation and flipping along with shearing methods while implementing weight-based class balancing. During the modeling phase the system implements spatial and temporal learning through the use of EfficientNetB3+LSTM hybrid architecture. The model uses callbacks including Early Stopping and Model Checkpoint during training and validation to enhance its operational efficiency. The final part of the process includes an evaluation step which establishes performance metrics before deploying the model for agricultural disease surveillance purposes.

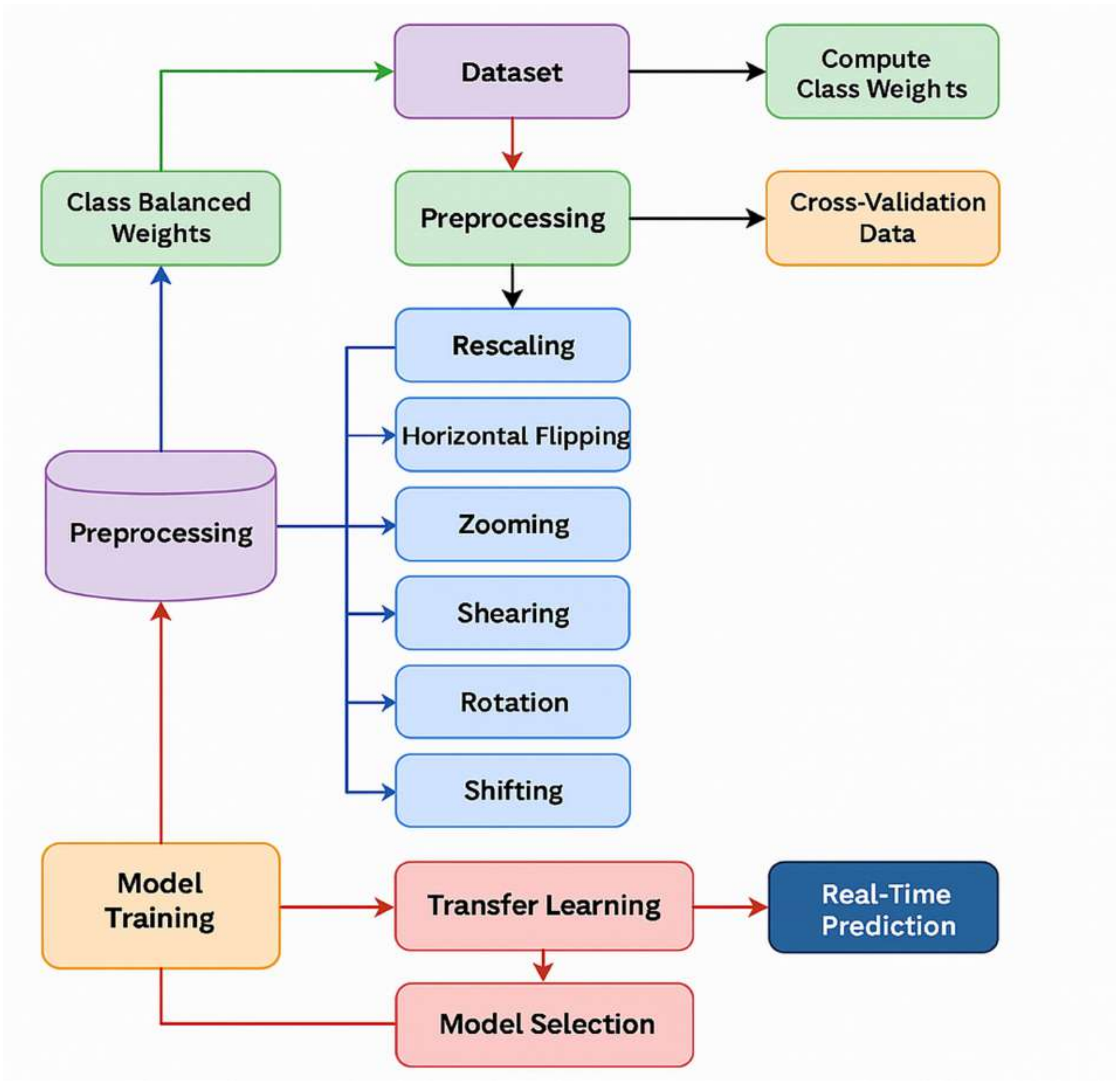


Fig 4.2.1 Data Flow Diagram

4.2.2. Class Diagram

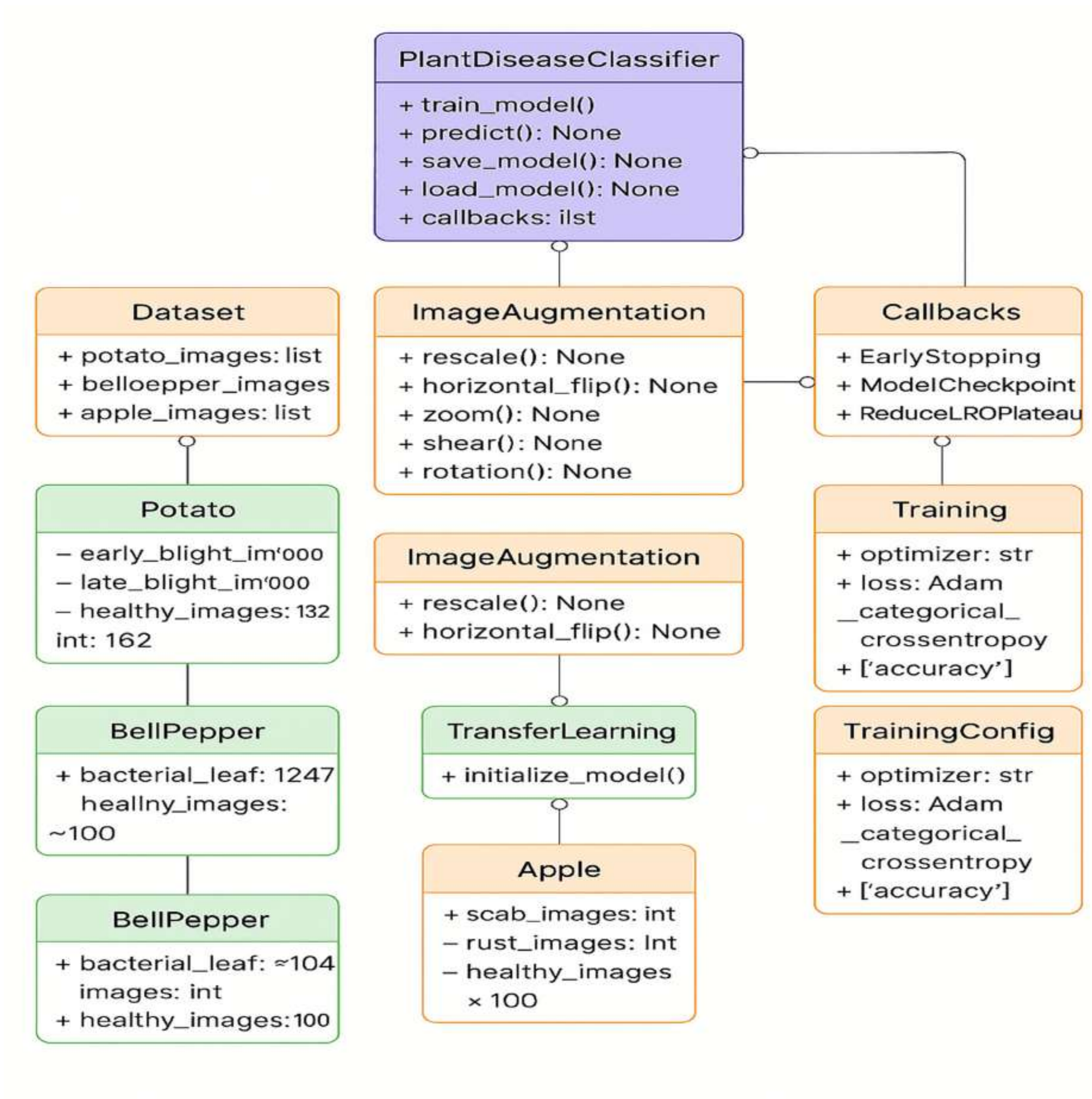


Fig 4.2.2 Class diagram

Chapter 5

METHODOLOGY AND TESTING

5.1 MODULE DESCRIPTION

The system functions as a disease detection platform for potato leaf leaves that applies deep learning algorithm models. The model started with training from a potato leaf database which included three disease categories: Potato Early Blight, Potato Late Blight along with Healthy Potato leaves. The system primarily includes data enhancement along with preprocessing as well as model training through transfer learning mechanisms and cross-crop validation between bell pepper and apple leaves before evaluation follows with performance metrics.

The preprocessing module in data makes images ready for analysis through normalization while also resizing them. The proposed data augmentation techniques which include zooming, flipping and rotation and shearing work to expand the diversity within the dataset. The model training module applies EfficientNetB3 along with VGG16 and ResNet and MobileNet followed by LSTM layers for temporal learning abilities. The model gains broader application capabilities on different plant species through parametric tests with bell pepper and apple crops.

5.2 TESTING

The primary dataset contains 2,152 potato leaf images which are split with 1000 Potato Early Blight pictures and 1000 Late Blight images and 152 healthy leaf pictures. The training process proved arduous due to the excessive imbalance of data classes. Scikit-learn calculated weight values to prioritize the minority class in the training process.

This figure presents a summary of the number of images in each class of the potato leaf dataset. It highlights the total number of images available for each disease category: Potato Early Blight, Potato Late Blight, and Healthy Potato.

The dataset quality was enhanced by performing image augmentation through Tensor Flow Image Data Generator. The augmentation process used horizontal flips along with random rotations up to 40 degrees and modifications of image dimensions and adjustments between dark and bright values.

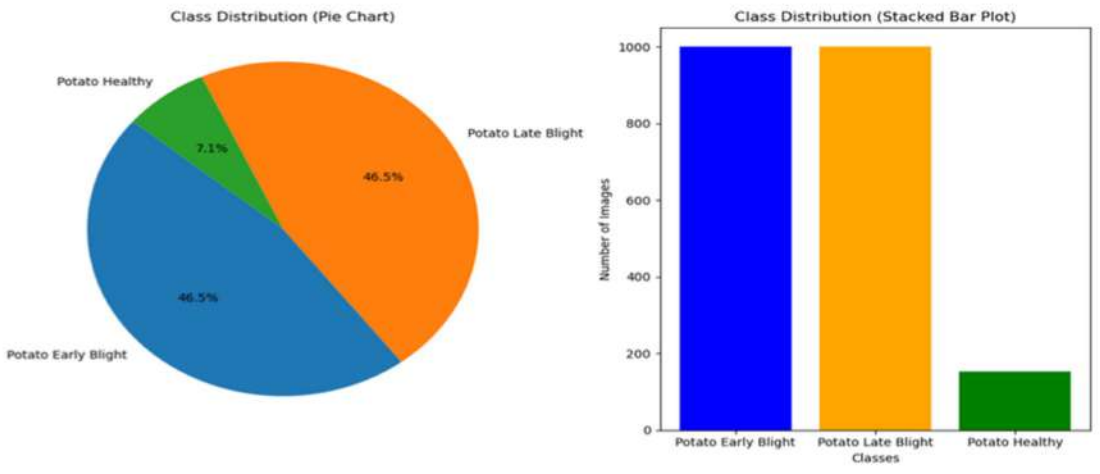


Figure 5.2.1: Class Distribution of Potato Leaf Dataset

This figure shows the percentage distribution of three potato leaf classes—Potato Early Blight, Potato Late Blight, and Healthy Potato leaves. It visually highlights the imbalance in the dataset, with the healthy class being underrepresented.

An increase in the model learning ability was achieved through additional datasets featuring bell pepper and apple leaves. The bell pepper dataset features 2475 images which display bacterial leaf infections as well as regular leaf pictures. There exist around 300 pictures featuring Apple Scab and Apple Rust diseases along with healthy apple leaf images in the apple dataset. The researchers applied these images in their testing across different crop species.

The model inherited high-level features through EfficientNetB3 and ResNet transfer learning processes that used pre-trained weights from the ImageNet database. Rephrase Sentence: Spatial and temporal patterns were learned through the implementation of LSTM layers. The model employed Adam optimizer combined with categorical cross-entropy loss function for its multi-class classification process. Where Adam optimizer is

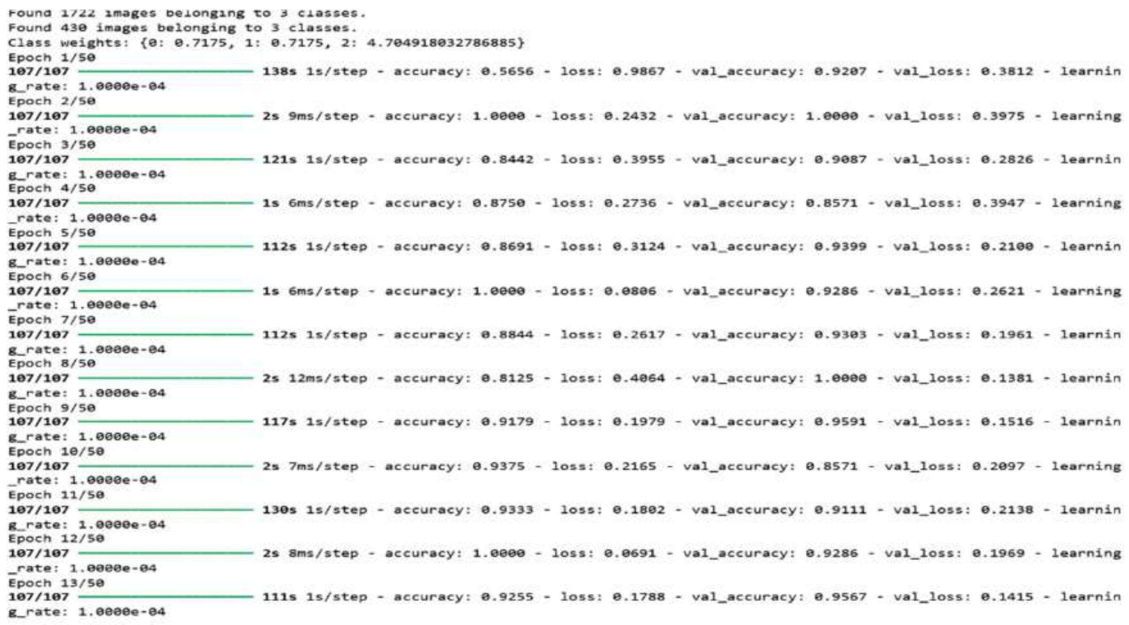
$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v^t + \epsilon}} \cdot m^t$$

Where:

- m^t and v^t are bias-corrected estimates of first and second moments
- η is learning rate

combination of different CNNs which integrated LSTM layers resulted in multiple hybrid architectural experiments. VGG16 worked as a feature extractor before Time Distributed Flattening and an LSTM layer processed the sequence in the hybrid architecture. EfficientNetB3+LSTM, ResNet+LSTM, along with MobileNet+LSTM served as different hybrid architectural structures. By combining these networks the system gained the ability to leverage both spatial features from convolutional layers and temporal features from LSTM at the same time.

Three important callbacks contained Early Stopping to end training after validation loss stabilization and Model Checkpoint for storing the best model and ReduceLROnPlateau to reduce learning rate at plateau occurrences.



```
Found 1722 images belonging to 3 classes.
Found 430 images belonging to 3 classes.
Class weights: {0: 0.7175, 1: 0.7175, 2: 4.704918032786885}
Epoch 1/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 138s 1s/step - accuracy: 0.5656 - loss: 0.9867 - val_accuracy: 0.9207 - val_loss: 0.3812 - learnin
g_rate: 1.0000e-04
Epoch 2/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 1.0000 - loss: 0.2432 - val_accuracy: 1.0000 - val_loss: 0.3975 - learning
_rate: 1.0000e-04
Epoch 3/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 121s 1s/step - accuracy: 0.8442 - loss: 0.3955 - val_accuracy: 0.9087 - val_loss: 0.2826 - learnin
g_rate: 1.0000e-04
Epoch 4/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.8750 - loss: 0.2736 - val_accuracy: 0.8571 - val_loss: 0.3947 - learning
_rate: 1.0000e-04
Epoch 5/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 112s 1s/step - accuracy: 0.8691 - loss: 0.3124 - val_accuracy: 0.9399 - val_loss: 0.2100 - learnin
g_rate: 1.0000e-04
Epoch 6/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 0.0806 - val_accuracy: 0.9286 - val_loss: 0.2621 - learning
_rate: 1.0000e-04
Epoch 7/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 112s 1s/step - accuracy: 0.8844 - loss: 0.2617 - val_accuracy: 0.9303 - val_loss: 0.1961 - learnin
g_rate: 1.0000e-04
Epoch 8/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 0.8125 - loss: 0.4064 - val_accuracy: 1.0000 - val_loss: 0.1381 - learnin
g_rate: 1.0000e-04
Epoch 9/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 117s 1s/step - accuracy: 0.9179 - loss: 0.1979 - val_accuracy: 0.9591 - val_loss: 0.1516 - learnin
g_rate: 1.0000e-04
Epoch 10/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.9375 - loss: 0.2165 - val_accuracy: 0.8571 - val_loss: 0.2097 - learning
_rate: 1.0000e-04
Epoch 11/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 130s 1s/step - accuracy: 0.9333 - loss: 0.1802 - val_accuracy: 0.9111 - val_loss: 0.2138 - learnin
g_rate: 1.0000e-04
Epoch 12/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 8ms/step - accuracy: 1.0000 - loss: 0.0691 - val_accuracy: 0.9286 - val_loss: 0.1969 - learning
_rate: 1.0000e-04
Epoch 13/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 111s 1s/step - accuracy: 0.9255 - loss: 0.1788 - val_accuracy: 0.9567 - val_loss: 0.1415 - learnin
g_rate: 1.0000e-04
```

Figure 5.2.2: Model Training Epochs Performance

This figure illustrates the performance of the model across different epochs during training.

The approach combines different components including data augmentations with transfer learning alongside class balancing and multipurpose model performance evaluation of various crop varieties. Through hybrid architectures which combine VGG+LSTM with ResNet+LSTM and EfficientNetB3+LSTM the system enhances its processing power for spatial and sequential information detection. The hybrid EfficientNetB3 model with LSTM component demonstrates superior performance by excelling above every other model when detecting plant diseases while maintaining its capability for general and robust use in different agricultural environments.

Chapter 6

PROJECT DEMONSTRATION

Verification of the plant disease diagnosis system functionality along with performance takes place through demonstration activities. The pipeline followed a correct sequence of phases to demonstrate how the plant disease diagnosis system functioned.

6.1 DEMONSTRATION PROCESS OVERVIEW

The interactive and graphical demonstration established how deep learning through the model effectively performs disease classification on potato leaf images. Steps below were showcased:

The dataset including images of potato leaves underwent preprocessing before the loading process completed. After image preprocessing through rotating, flipping and zooming we enhanced our dataset strength to accommodate different image distortion types.

A training process applied the EfficientNetB3 alongside LSTM and EfficientNetB3+LSTM on the potato leaf dataset. The use of Transfer learning enabled the efficiency of pre-trained ImageNet weights for feature extraction. The presentation displayed training logs and accuracy plots which showed better performance output from different models.

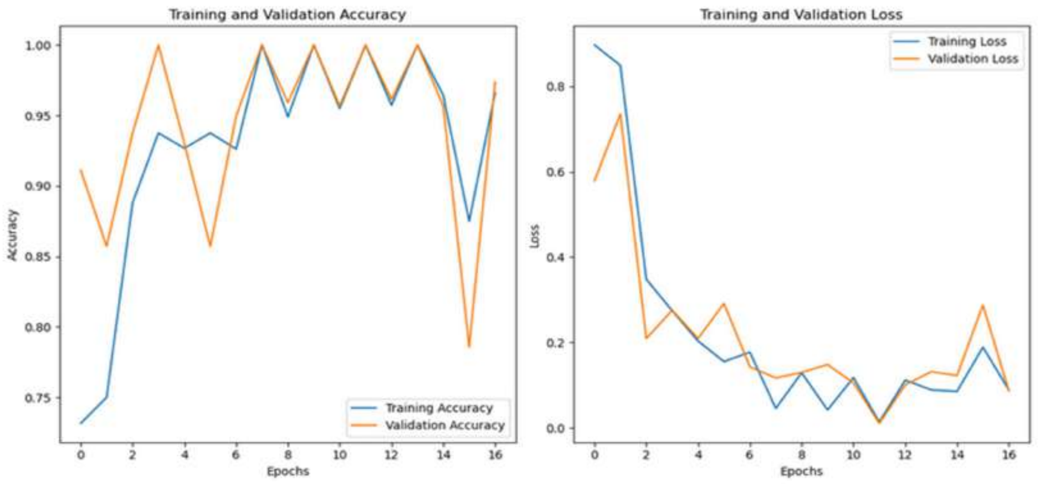


Fig 6.1 Training and Validation loss and accuracy

Testing of trained models included assessing their performance using potato leaf images as well as applying them to apple and bell pepper images. The evaluation showed performance measurements including precision along with accuracy and recall along with confusion matrices. The model displayed its capacity to apply its learning to various crop categories.

6.2 DEMONSTRATION OF MODEL PERFORMANCE

The system performed real-time classification by allowing users to feed current potato leaf images to demonstrate its disease detection capability between Early Blight and Late Blight or Healthy.

Part of the demonstration showcased model effectiveness across diverse types of agricultural samples including apple together with bell pepper type crops which demonstrates its versatility across different environmental conditions.

6.3 USER INTERFACE AND INTERACTION

When interaction with the model required it a basic user interface (UI) or script was utilized. Users could upload pictures of potato leaves through the system interface before receiving disease predictions and easily understandable output results. Users received visual outputs of both probability scores and class labels which improved their ability to understand the model interpretations.

6.4 KEY INSIGHTS AND RESULTS

The hybrid model consisting of EfficientNetB3+LSTM achieved the highest accuracy at 96.51% in recognizing potato leaf features because it effectively captures spatial along with temporal information from images.

The model demonstrated successful application and accuracy levels of 84.78% and 98.00% when tested on datasets consisting of apple and bell pepper images thus showing its multi-crop capabilities.

The model evaluation used accuracy plots together with confusion matrices which displayed both model strengths and weaknesses across different classes.

6.5 FUTURE ENHANCEMENTS

Some prospective solution advancements were discussed during the demo stage which covered:

Diversifying dataset content will help the model build stronger generalization abilities.

The system can gain accuracy in disease detection through implementing sophisticated attention mechanisms together with other sensor data (e.g., environmental conditions).

Chapter 7

RESULT AND DISCUSSION

The hybrid model particularly EfficientNetB3+LSTM showed a significant edge over isolated CNN or LSTM models. This is because they are capable of capturing spatial features (through CNN) and temporal/sequential patterns (through LSTM), thus being well-suited for intricate visual classification tasks such as plant disease identification.

7.1 POTATO LEAF CLASSIFICATION

The combination of EfficientNetB3 with LSTM produced a maximum accuracy level of 96.51% which outperformed all alternative models as shown in Table 1. Tests demonstrated that MobileNet delivered 95.11% accuracy and EfficientNetB3 achieved 93.25%. The baseline CNNs ResNet and VGG performed poorly in this task until they were merged with LSTM which demonstrated the value of sequential modeling for pattern analysis.

The testing results across models are summarized in the table below:

Table 7.1: Potato Leaf Classification Accuracy Across Models

Model	**Accuracy (%)**
EfficientNetB3	93.25
LSTM	85.71
EfficientNetB3+LSTM	**9**
ResNet	71.43
ResNet+LSTM	87.00
VGG	86.78
LSTM+VGG	90.12
MobileNet	95.11

MobileNet+LSTM	95.91

7.2 APPLE LEAF CLASSIFICATION (CROSS-VALIDATION)

The EfficientNetB3+LSTM model demonstrated high stability when evaluating apple leaves through its 84.78% accuracy according to Table 2. The performance metrics of MobileNet+LSTM and LSTM+VGG demonstrated similar levels to each other. The hybrid approach demonstrates effective features which allow the model to recognize various plant species.

Table 7.2: Apple Leaf Classification (Cross-Validation)

Model	**Accuracy (%)**
EfficientNetB3+LSTM	**84.78**
ResNet+LSTM	76.09
LSTM+VGG	78.26
MobileNet+LSTM	83.00

7.3 BELL PEPPER CLASSIFICATION (CROSS-VALIDATION)

The bell pepper data showed that EfficientNetB3+LSTM delivered an accuracy rate of 98.00% according to the data in Table 3. The hybrid model exhibits transferable and robust performance capabilities because it shows high levels of accuracy across different datasets. Both LSTM+VGG and MobileNet+LSTM achieved solid performance levels because combining CNNs with sequence-learning layers brings additional strength to the model.

Table 7.3: Bell Pepper Classification (Cross-Validation)

Model	Accuracy (%)
EfficientNetB3+LSTM	**98.00**
ResNet+LSTM	89.00
LSTM+VGG	95.83
MobileNet+LSTM	95.46

7.4 DISCUSSION

The hybrid models reached their peak performance with EfficientNetB3+LSTM by surpassing isolated CNN or LSTM applications. Through CNN capabilities they detect spatial features and LTS container goes beyond to discover temporal/sequential patterns making them excellent options for complex visual classification operations like plant disease recognition.

Major enhancements as part of the methodology approach included:

- Transfer Learning with pre-trained models.
- The technique of Class Balancing was applied to solve the problem of unbalanced data.
- Data Augmentation to enhance generalization.
- The methodology uses Cross-Validation as a method to check model performance across multiple crops.
- From this project we can analyze that combining deep feature extraction techniques with sequential modeling helps in increasing accuracy rates and making a trustable model.

7.5 COST ANALYSIS

Dataset Acquisition:

- The open-source platform Kaggle served as the origin for acquiring all datasets consisting of potato and bell pepper and apple leaves.
- Cost: ₹0

Data Augmentation:

- We deployed ImageDataGenerator from TensorFlow which is an open-source library to execute augmentation techniques.
- Cost: ₹0

Model Development (Transfer Learning):

- Residual and VGG networks and other pre-trained CNN models served as the basis for development instead of new model training.
- Cost: ₹0 (open-source models)

Training Environment:

- Used Google Colab (Free + Pro) with GPU acceleration.
- The estimated price for Google Colab Professional edition amounts to ₹2500 which includes fifty hours of GPU processing time.
- Internet, Power & Storage:
- Approximate cost for internet data usage, electricity, and cloud storage (e.g., Google Drive).
- Estimated Cost: ₹1,000

Chapter 8
CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

- A deep learning-based system for plant disease classification becomes operational due to CNNs and transfer learning.
- The project serves to detect three crops namely potato and bell pepper and apple and multiple diseases with enhanced accuracy levels.
- Three approaches enhance model generalization which include image augmentation along with class balancing and cross-validation.
- Transfer learning benefits this model because it shortens training durations without affecting operational performance when operating with small datasets.
- The model demonstrates practical efficiency according to evaluative assessments that support its implementation as a smart agricultural technology.

8.2 LIMITATIONS

- A deep learning-based system for plant disease classification becomes operational due to CNNs and transfer learning.
- The project serves to detect three crops namely potato and bell pepper and apple and multiple diseases with enhanced accuracy levels.
- Model stability throughout different conditions and across classes is enabled through image augmentation strategies and both class balancing practices and cross-validation methods.
- Transfer learning benefits this model because it shortens training durations without affecting operational performance when operating with small datasets.

- The model demonstrates practical efficiency according to evaluative assessments that support its implementation as a smart agricultural technology.

8.3 FUTURE WORK

- The Farmers can now utilize their smartphones or web applications to directly present images for instantaneous outcomes.
- The system should acquire additional real-time images which cover multiple geographical areas as well as diverse crop species.
- The prediction precision would improve through incorporation of temperature and humidity and seasonal inputs.
- A live testing phase has to be conducted under real agricultural conditions through joint efforts with farmers along with agronomists.
- The implementation of Grad-CAM alongside other explanatory methods helps to show model decision making processes in order to build trust and transparency.
- The model should undergo quantization and compression procedures to enable its deployment on low-power edge devices.
- After deployment the model should engage in continuous learning through online learning processes that use continuous new-data updates.

Chapter 9

REFERENCES

[1] Nagababu, P., Nageena, S., Dharani, V., & Naveen, D. (2024). Plant disease detection and diagnosis. *Plant Disease Detection and Diagnosis*, 1–6. https://doi.org/10.1109/incet61516.2024.10593371

[2] Nagababu, P., Nageena, S., Dharani, V., & Naveen, D. (2024a). An automated and fine- tuned image detection and classification system for plant leaf diseases. *An Automated and Fine- Tuned Image Detection and Classification System for Plant Leaf Diseases*, 1–6. https://doi.org/10.1109/incet61516.2024.10593371

[3] Pandey, P., Patyane, K., Padekar, M., Mohite, R., Mane, P., & Avhad, A. (2023). Plant Disease Detection Using Deep Learning Model - Application FarmEasy. *Plant Disease Detection Using Deep Learning Model - Application FarmEasy*, *15*, 1–6. https://doi.org/10.1109/icacta58201.2023.10393095

[4]Yaswanth, D., Manoj, S. S., Yadav, M. S., & Chowdary, E. D. (2024). Plant leaf disease detection using transfer learning approach. *2020 IEEE International Students' Conference on Electrical,Electronics and Computer Science (SCEECS)*, 1–6. https://doi.org/10.1109/sceecs61402.2024.10482053

[5] Singh, V. K. (2023). SUBCOPLED: Superpixel based color distribution driven plant leaf disease detection. *2022 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS)*, 864–868. https://doi.org/10.1109/icccis60361.2023.10425303

[6] Biswas, B., & Yadav, R. K. (2023). A Review of Convolutional Neural Network-based Approaches for Disease Detection in Plants. *A Review of Convolutional Neural Network-based Approaches for Disease Detection in Plants*. https://doi.org/10.1109/idciot56793.2023.10053428

[7] Sangeetha, T., Rajarajan, R., Krishna, S. R., & Siddharth, N. S. (2024). A novel smart approach to plant health - Automated detection and diagnosis of leaf diseases. *2022 International Conference on Inventive Computation Technologies (ICICT)*. https://doi.org/10.1109/icict60155.2024.10544660

[8] Balwani, S., & Bawane, N. (2024). Analytical Review on Smart Plant Disease Detection and Prevention System. *Analytical Review on Smart Plant Disease*

Detection and Prevention System, 1–5. https://doi.org/10.1109/icicet59348.2024.10616362

[9] Deepa, A. R., Chaurasia, M. A., Vamsi, S. B. N., Kumar, B. M., Reddy, V. S., & Anand, K. T. (2023). Plant Diseases and Pests Detection Using Machine Learning. *Plant Diseases and Pests Detection Using Machine Learning*, 1–4. https://doi.org/10.1109/asiancon58793.2023.10270264

[10] Reddy, V. C. G., Xavier, V. M. A., & Shyni, S. S. (2024). Plant disease detection and pesticide recommendation using deep learning. *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2543–2547. https://doi.org/10.1109/icaccs60874.2024.10717268

[11] Tandekar, D., & Dongre, S. (2023b). Identification of various diseases in plant leaves using image processing and CNN approach. *2022 13th International Conference on Computing Communication and Networking Technologies (ICCCNT)*. https://doi.org/10.1109/icccnt56998.2023.10306979

[12] Amritraj, S., Hans, N., & Cyril, C. P. D. (2023). An Automated and Fine- Tuned Image Detection and Classification System for Plant Leaf Diseases. *An Automated and Fine- Tuned Image Detection and Classification System for Plant Leaf Diseases*, 1–5. https://doi.org/10.1109/raeeucci57140.2023.10134461

[13] Shrotriya, A., Sharma, A. K., Bairwa, A. K., & Manoj, R. (2024). Hybrid Ensemble Learning with CNN and RNN for Multimodal Cotton Plant Disease Detection. *IEEE Access*, *12*, 198028–198045. https://doi.org/10.1109/access.2024.3515843

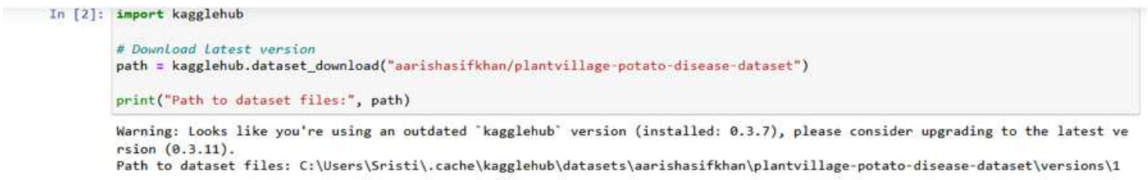
[14] Haruna, A. A., Badi, I. A., Muhammad, L. J., Abuobieda, A., & Altamimi, A. (2023). CNN-LSTM Learning Approach for Classification of Foliar Disease of Apple. *CNN-LSTM Learning Approach for Classification of Foliar Disease of Apple*, 1–6. https://doi.org/10.1109/icaisc56366.2023.10085039

[15] Devi, E., Gopi, S., Padmavathi, U., Arumugam, S. R., Premnath, S., & Muralitharan, D. (2023). Plant Disease Classification using CNN-LSTM Techniques. *2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT)*, 1225–1229. https://doi.org/10.1109/icssit55814.2023.10061003

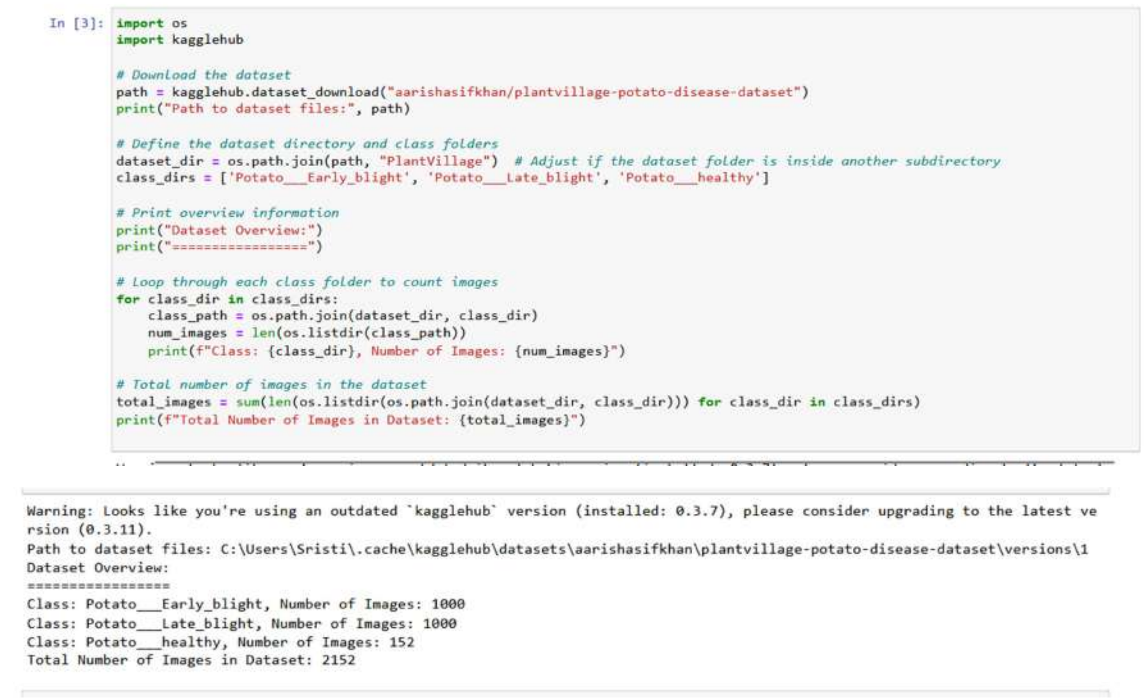
[16] Kaur, A., Kukreja, V., Malhotra, S., Joshi, K., & Sharma, R. (2024). Rice Sheath Rot Disease Detection and Severity Classification: A Novel Framework Leveraging CNN-LSTM Models for Multi-Classification. *Rice Sheath Rot Disease Detection and Severity Classification: A Novel Framework Leveraging CNN-LSTM Models for Multi-Classification*, 1–5. https://doi.org/10.1109/iatmsi60426.2024.10502629

[17] Sharma, R., Kukreja, V., & Vats, S. (2023). A New Dawn for Tomato-spotted wilt virus Detection and Intensity Classification: A CNN and LSTM Ensemble Model. *A New Dawn for Tomato-spotted Wilt Virus Detection and Intensity Classification: A CNN and LSTM Ensemble Model*. https://doi.org/10.1109/incet57972.2023.10170406

APPENDIX A – Sample Code



```
In [2]: import kagglehub

        # Download latest version
        path = kagglehub.dataset_download("aarishasifkhan/plantvillage-potato-disease-dataset")

        print("Path to dataset files:", path)

        Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
        rsion (0.3.11).
        Path to dataset files: C:\Users\Sristi\.cache\kagglehub\datasets\aarishasifkhan\plantvillage-potato-disease-dataset\versions\1
```



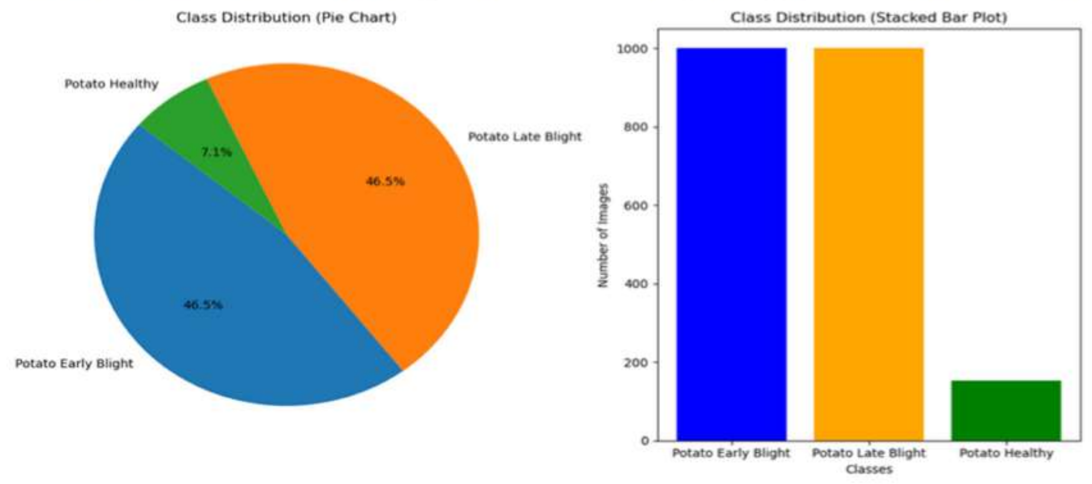
```
In [3]: import os
        import kagglehub

        # Download the dataset
        path = kagglehub.dataset_download("aarishasifkhan/plantvillage-potato-disease-dataset")
        print("Path to dataset files:", path)

        # Define the dataset directory and class folders
        dataset_dir = os.path.join(path, "PlantVillage")  # Adjust if the dataset folder is inside another subdirectory
        class_dirs = ['Potato___Early_blight', 'Potato___Late_blight', 'Potato___healthy']

        # Print overview information
        print("Dataset Overview:")
        print("=================")

        # Loop through each class folder to count images
        for class_dir in class_dirs:
            class_path = os.path.join(dataset_dir, class_dir)
            num_images = len(os.listdir(class_path))
            print(f"Class: {class_dir}, Number of Images: {num_images}")

        # Total number of images in the dataset
        total_images = sum(len(os.listdir(os.path.join(dataset_dir, class_dir))) for class_dir in class_dirs)
        print(f"Total Number of Images in Dataset: {total_images}")
```

```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
rsion (0.3.11).
Path to dataset files: C:\Users\Sristi\.cache\kagglehub\datasets\aarishasifkhan\plantvillage-potato-disease-dataset\versions\1
Dataset Overview:
=================
Class: Potato___Early_blight, Number of Images: 1000
Class: Potato___Late_blight, Number of Images: 1000
Class: Potato___healthy, Number of Images: 152
Total Number of Images in Dataset: 2152
```

Importing potato data from Kaggle, with total **2152 images** having 3 classes

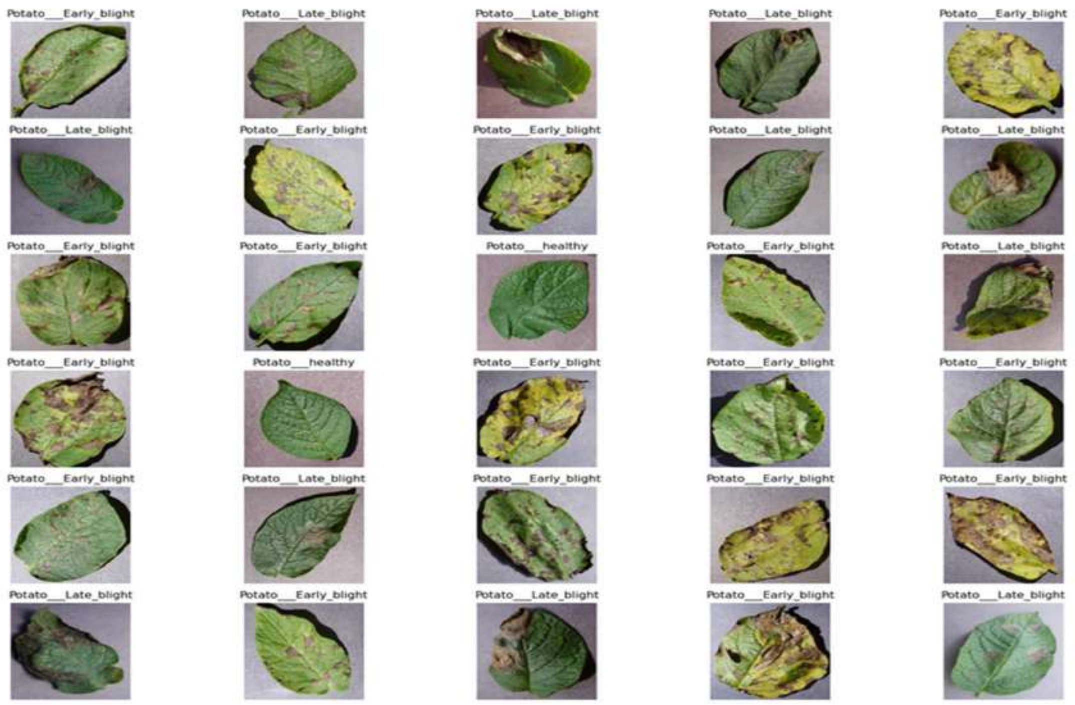
these 3 classes are demonstrated below in form of pie chart and bar graph

```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
rsion (0.3.11).
Path to dataset files: C:\Users\Sristi\.cache\kagglehub\datasets\aarishasifkhan\plantvillage-potato-disease-dataset\versions\1
```

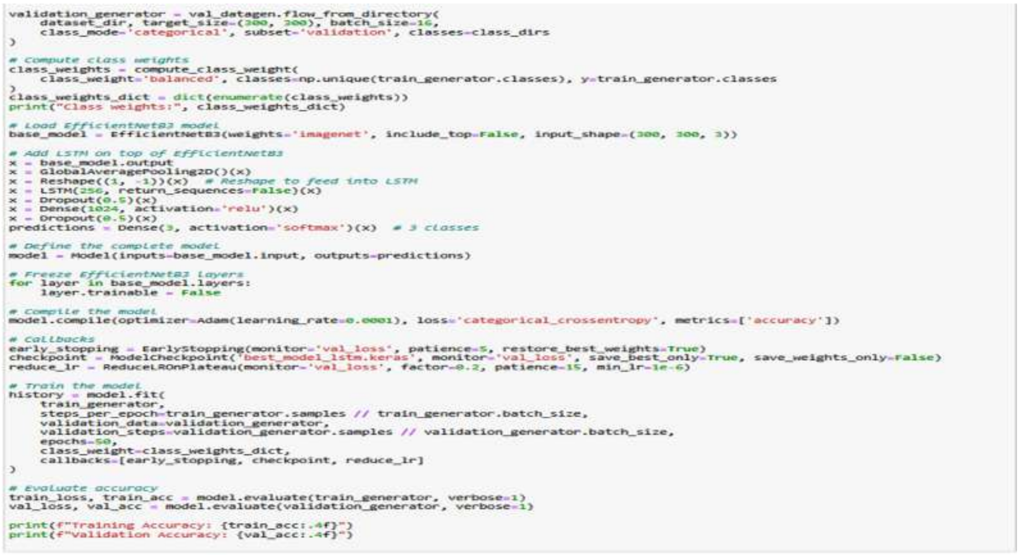


Pictures of Potato leaves, consisting of three classes healthy, early blight and late blight

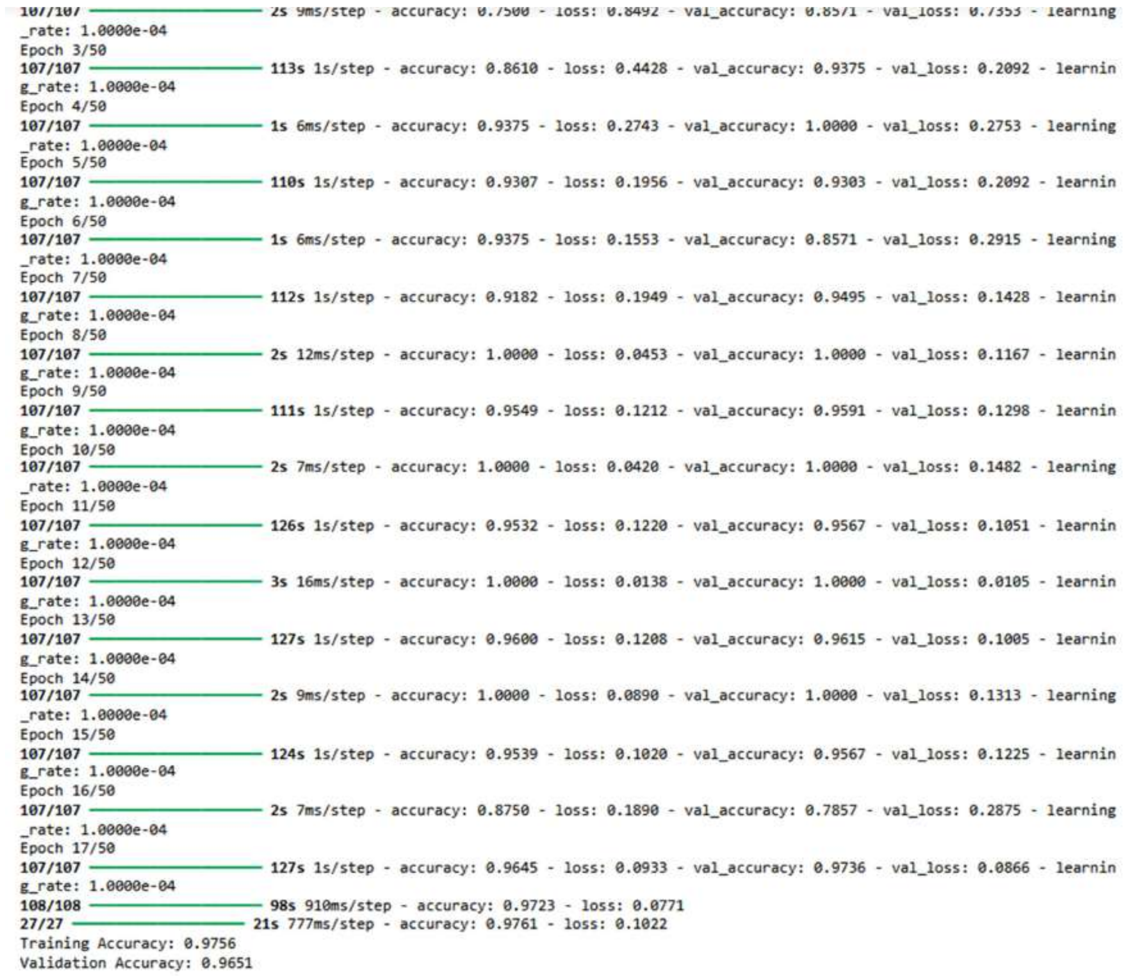
```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
rsion (0.3.11).
Dataset stored at: C:\Users\Sristi\.cache\kagglehub\datasets\aarishasifkhan\plantvillage-potato-disease-dataset\versions\1
```



Implementation of EfficientNet+LSTM code, performing several epochs and its accuracy scores

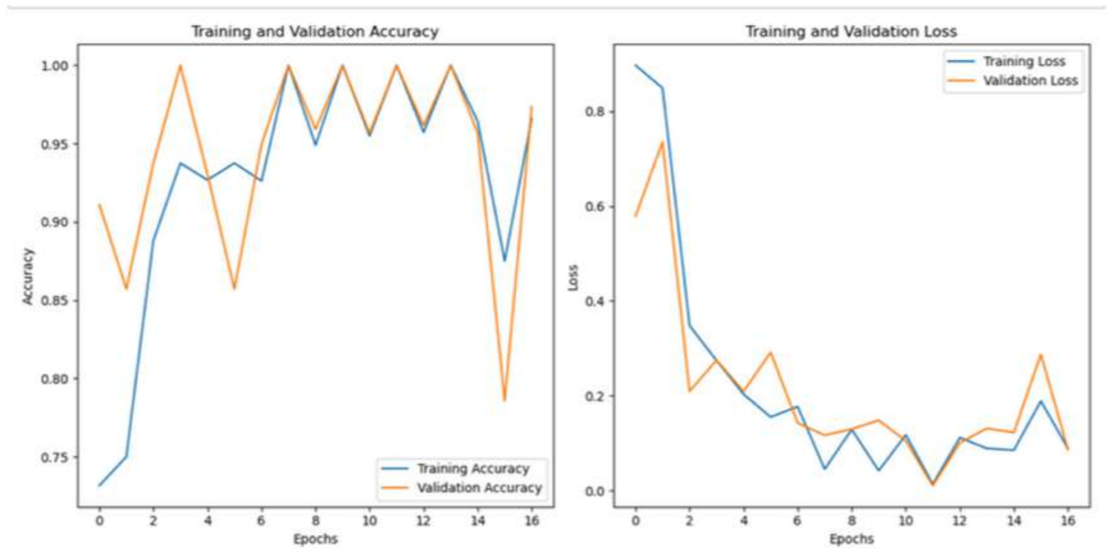


```python
validation_generator = val_datagen.flow_from_directory(
    dataset_dir, target_size=(300, 300), batch_size=16,
    class_mode='categorical', subset='validation', classes=class_dirs
)

# Compute class weights
class_weights = compute_class_weight(
    class_weight='balanced', classes=np.unique(train_generator.classes), y=train_generator.classes
)
class_weights_dict = dict(enumerate(class_weights))
print("Class weights:", class_weights_dict)

# Load EfficientNetB3 model
base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape=(300, 300, 3))

# Add LSTM on top of EfficientNetB3
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Reshape((1, -1))(x)  # Reshape to feed into LSTM
x = LSTM(256, return_sequences=False)(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(3, activation='softmax')(x)  # 3 classes

# Define the complete model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze EfficientNetB3 layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model_lstm.keras', monitor='val_loss', save_best_only=True, save_weights_only=False)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=15, min_lr=1e-6)

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    epochs=50,
    class_weight=class_weights_dict,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)

# Evaluate accuracy
train_loss, train_acc = model.evaluate(train_generator, verbose=1)
val_loss, val_acc = model.evaluate(validation_generator, verbose=1)

print(f"Training Accuracy: {train_acc:.4f}")
print(f"Validation Accuracy: {val_acc:.4f}")
```



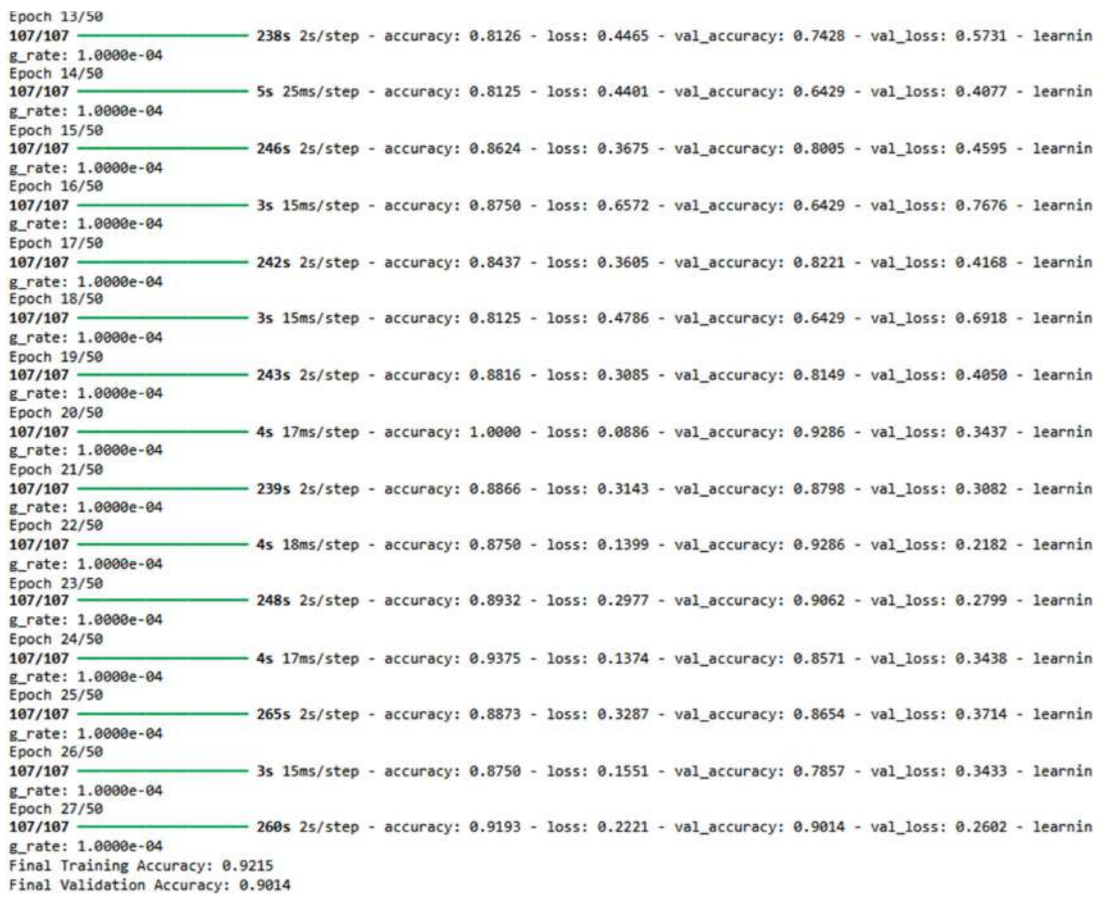
```
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 0.7500 - loss: 0.8492 - val_accuracy: 0.8571 - val_loss: 0.7353 - learning
_rate: 1.0000e-04
Epoch 3/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 113s 1s/step - accuracy: 0.8610 - loss: 0.4428 - val_accuracy: 0.9375 - val_loss: 0.2092 - learnin
g_rate: 1.0000e-04
Epoch 4/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9375 - loss: 0.2743 - val_accuracy: 1.0000 - val_loss: 0.2753 - learning
_rate: 1.0000e-04
Epoch 5/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 110s 1s/step - accuracy: 0.9307 - loss: 0.1956 - val_accuracy: 0.9303 - val_loss: 0.2092 - learnin
g_rate: 1.0000e-04
Epoch 6/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.9375 - loss: 0.1553 - val_accuracy: 0.8571 - val_loss: 0.2915 - learning
_rate: 1.0000e-04
Epoch 7/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 112s 1s/step - accuracy: 0.9182 - loss: 0.1949 - val_accuracy: 0.9495 - val_loss: 0.1428 - learnin
g_rate: 1.0000e-04
Epoch 8/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 12ms/step - accuracy: 1.0000 - loss: 0.0453 - val_accuracy: 1.0000 - val_loss: 0.1167 - learnin
g_rate: 1.0000e-04
Epoch 9/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 111s 1s/step - accuracy: 0.9549 - loss: 0.1212 - val_accuracy: 0.9591 - val_loss: 0.1298 - learnin
g_rate: 1.0000e-04
Epoch 10/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 1.0000 - loss: 0.0420 - val_accuracy: 1.0000 - val_loss: 0.1482 - learning
_rate: 1.0000e-04
Epoch 11/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 126s 1s/step - accuracy: 0.9532 - loss: 0.1220 - val_accuracy: 0.9567 - val_loss: 0.1051 - learnin
g_rate: 1.0000e-04
Epoch 12/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 16ms/step - accuracy: 1.0000 - loss: 0.0138 - val_accuracy: 1.0000 - val_loss: 0.0105 - learnin
g_rate: 1.0000e-04
Epoch 13/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 127s 1s/step - accuracy: 0.9600 - loss: 0.1208 - val_accuracy: 0.9615 - val_loss: 0.1005 - learnin
g_rate: 1.0000e-04
Epoch 14/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 9ms/step - accuracy: 1.0000 - loss: 0.0890 - val_accuracy: 1.0000 - val_loss: 0.1313 - learning
_rate: 1.0000e-04
Epoch 15/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 124s 1s/step - accuracy: 0.9539 - loss: 0.1020 - val_accuracy: 0.9567 - val_loss: 0.1225 - learnin
g_rate: 1.0000e-04
Epoch 16/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 7ms/step - accuracy: 0.8750 - loss: 0.1890 - val_accuracy: 0.7857 - val_loss: 0.2875 - learning
_rate: 1.0000e-04
Epoch 17/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 127s 1s/step - accuracy: 0.9645 - loss: 0.0933 - val_accuracy: 0.9736 - val_loss: 0.0866 - learnin
g_rate: 1.0000e-04
108/108 ━━━━━━━━━━━━━━━━━━━━ 98s 910ms/step - accuracy: 0.9723 - loss: 0.0771
27/27 ━━━━━━━━━━━━━━━━━━━━ 21s 777ms/step - accuracy: 0.9761 - loss: 0.1022
Training Accuracy: 0.9756
Validation Accuracy: 0.9651
```

The accuracy obtained from this model is **96.51%** and is our best performing model as it worked well with apple and bell pepper Datasets as well



Training and Validation accuracy and loss graph , showing the model accuracy scores

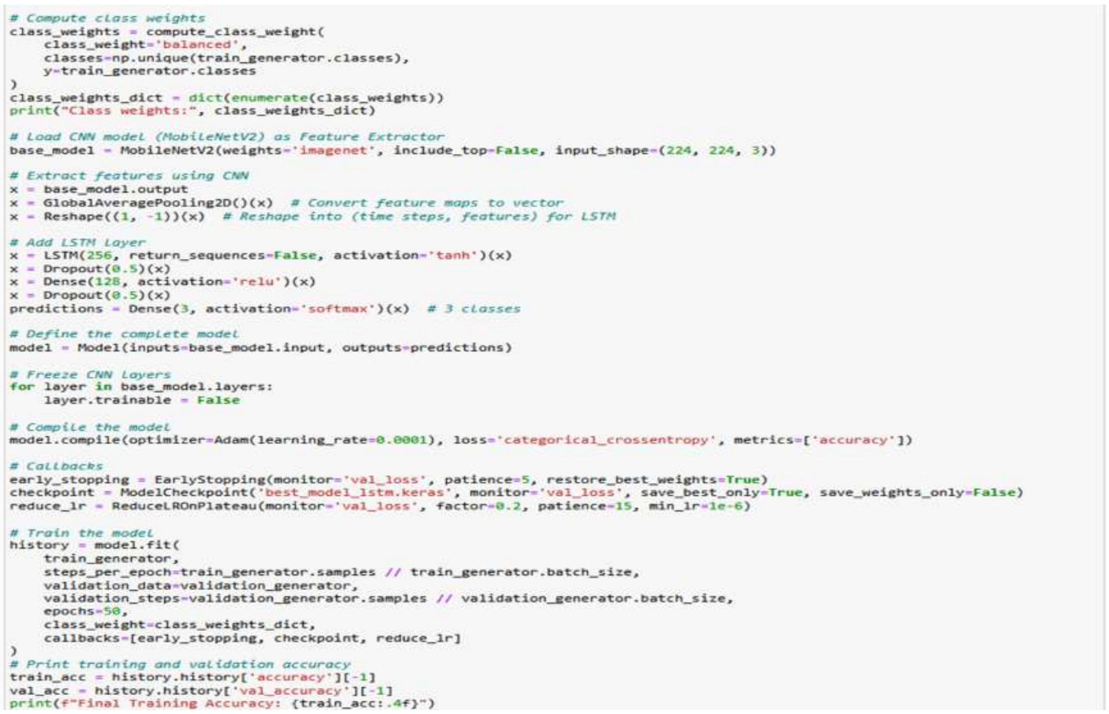
Implementation of **VGG+LSTM code**, performing several epochs and its accuracy scores



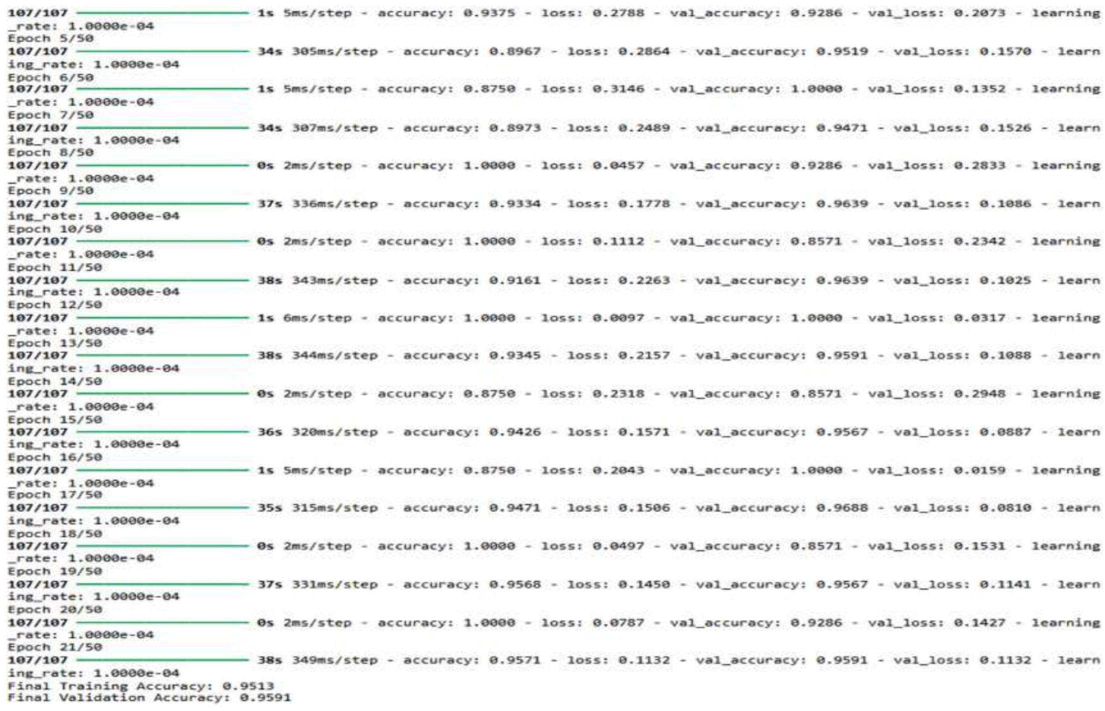
```
Epoch 13/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 238s 2s/step - accuracy: 0.8126 - loss: 0.4465 - val_accuracy: 0.7428 - val_loss: 0.5731 - learnin
g_rate: 1.0000e-04
Epoch 14/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 5s 25ms/step - accuracy: 0.8125 - loss: 0.4401 - val_accuracy: 0.6429 - val_loss: 0.4077 - learnin
g_rate: 1.0000e-04
Epoch 15/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 246s 2s/step - accuracy: 0.8624 - loss: 0.3675 - val_accuracy: 0.8005 - val_loss: 0.4595 - learnin
g_rate: 1.0000e-04
Epoch 16/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 0.8750 - loss: 0.6572 - val_accuracy: 0.6429 - val_loss: 0.7676 - learnin
g_rate: 1.0000e-04
Epoch 17/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 242s 2s/step - accuracy: 0.8437 - loss: 0.3605 - val_accuracy: 0.8221 - val_loss: 0.4168 - learnin
g_rate: 1.0000e-04
Epoch 18/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 0.8125 - loss: 0.4786 - val_accuracy: 0.6429 - val_loss: 0.6918 - learnin
g_rate: 1.0000e-04
Epoch 19/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 243s 2s/step - accuracy: 0.8816 - loss: 0.3085 - val_accuracy: 0.8149 - val_loss: 0.4050 - learnin
g_rate: 1.0000e-04
Epoch 20/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 1.0000 - loss: 0.0886 - val_accuracy: 0.9286 - val_loss: 0.3437 - learnin
g_rate: 1.0000e-04
Epoch 21/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 239s 2s/step - accuracy: 0.8866 - loss: 0.3143 - val_accuracy: 0.8798 - val_loss: 0.3082 - learnin
g_rate: 1.0000e-04
Epoch 22/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 4s 18ms/step - accuracy: 0.8750 - loss: 0.1399 - val_accuracy: 0.9286 - val_loss: 0.2182 - learnin
g_rate: 1.0000e-04
Epoch 23/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 248s 2s/step - accuracy: 0.8932 - loss: 0.2977 - val_accuracy: 0.9062 - val_loss: 0.2799 - learnin
g_rate: 1.0000e-04
Epoch 24/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 4s 17ms/step - accuracy: 0.9375 - loss: 0.1374 - val_accuracy: 0.8571 - val_loss: 0.3438 - learnin
g_rate: 1.0000e-04
Epoch 25/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 265s 2s/step - accuracy: 0.8873 - loss: 0.3287 - val_accuracy: 0.8654 - val_loss: 0.3714 - learnin
g_rate: 1.0000e-04
Epoch 26/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 15ms/step - accuracy: 0.8750 - loss: 0.1551 - val_accuracy: 0.7857 - val_loss: 0.3433 - learnin
g_rate: 1.0000e-04
Epoch 27/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 260s 2s/step - accuracy: 0.9193 - loss: 0.2221 - val_accuracy: 0.9014 - val_loss: 0.2602 - learnin
g_rate: 1.0000e-04
Final Training Accuracy: 0.9215
Final Validation Accuracy: 0.9014
```

The accuracy obtained from this model is **90.41%.**

Implementation of MobilenetV2 + LSTM code, performing several epochs and its accuracy scores



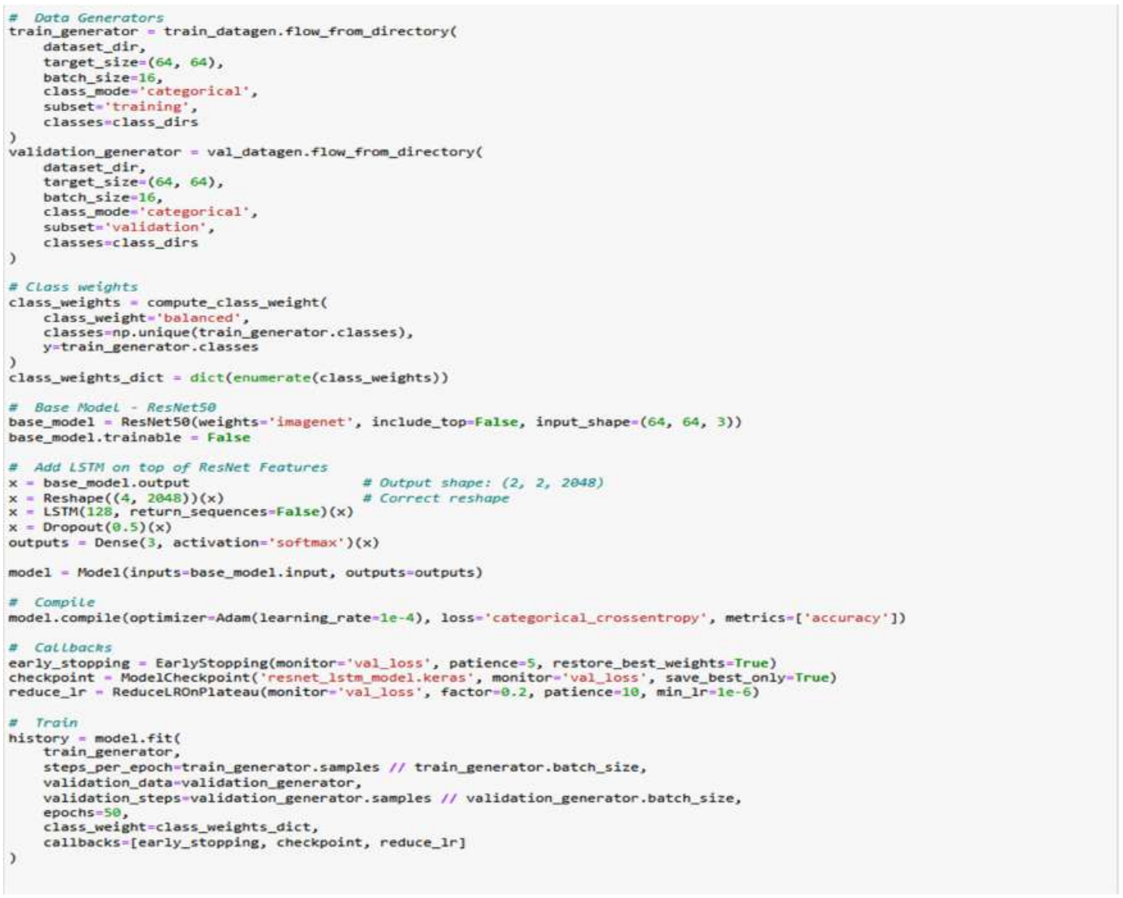
```python
# Compute class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)
class_weights_dict = dict(enumerate(class_weights))
print("Class weights:", class_weights_dict)

# Load CNN model (MobileNetV2) as Feature Extractor
base_model = MobileNetV2(weights='imagenet', include_top=False, input_shape=(224, 224, 3))

# Extract features using CNN
x = base_model.output
x = GlobalAveragePooling2D()(x)  # Convert feature maps to vector
x = Reshape((1, -1))(x)  # Reshape into (time steps, features) for LSTM

# Add LSTM Layer
x = LSTM(256, return_sequences=False, activation='tanh')(x)
x = Dropout(0.5)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(3, activation='softmax')(x)  # 3 classes

# Define the complete model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze CNN Layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001), loss='categorical_crossentropy', metrics=['accuracy'])

# Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('best_model_lstm.keras', monitor='val_loss', save_best_only=True, save_weights_only=False)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=15, min_lr=1e-6)

# Train the model
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    epochs=50,
    class_weight=class_weights_dict,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)
# Print training and validation accuracy
train_acc = history.history['accuracy'][-1]
val_acc = history.history['val_accuracy'][-1]
print(f"Final Training Accuracy: {train_acc:.4f}")
```



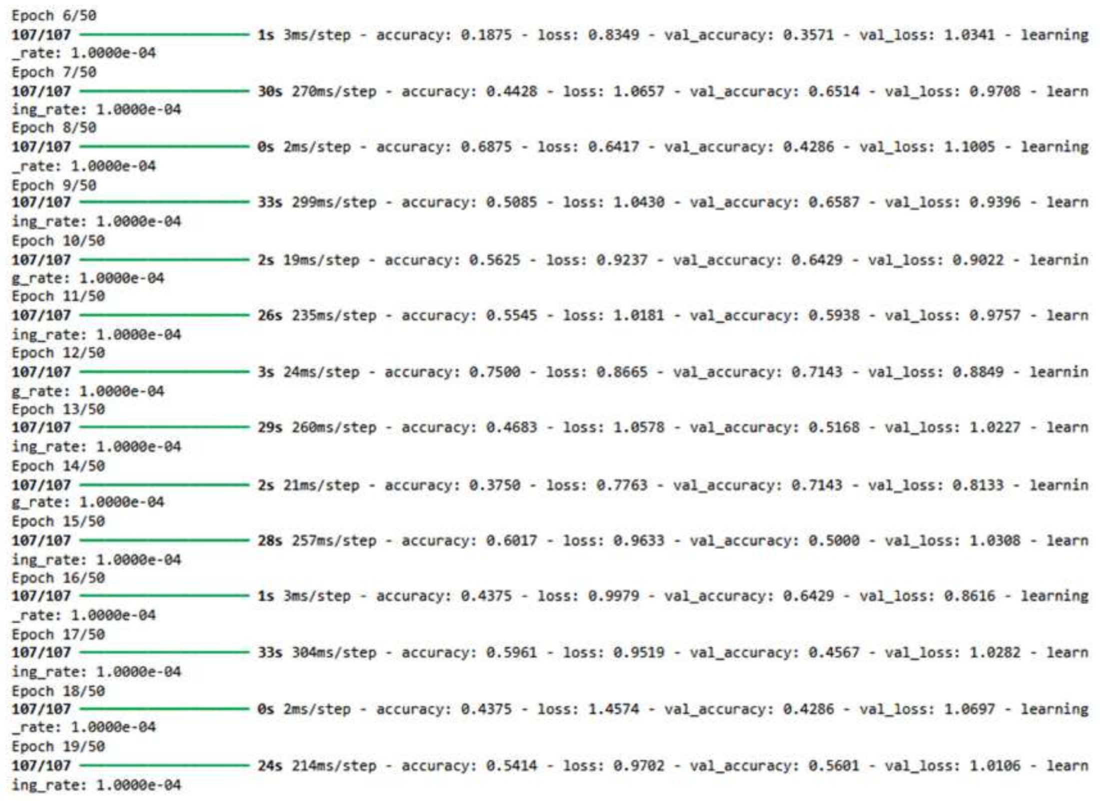
```
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.9375 - loss: 0.2788 - val_accuracy: 0.9286 - val_loss: 0.2073 - learning
_rate: 1.0000e-04
Epoch 5/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 34s 305ms/step - accuracy: 0.8967 - loss: 0.2864 - val_accuracy: 0.9519 - val_loss: 0.1570 - learn
ing_rate: 1.0000e-04
Epoch 6/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8750 - loss: 0.3146 - val_accuracy: 1.0000 - val_loss: 0.1352 - learning
_rate: 1.0000e-04
Epoch 7/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 34s 307ms/step - accuracy: 0.8973 - loss: 0.2489 - val_accuracy: 0.9471 - val_loss: 0.1526 - learn
ing_rate: 1.0000e-04
Epoch 8/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0457 - val_accuracy: 0.9286 - val_loss: 0.2833 - learning
_rate: 1.0000e-04
Epoch 9/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 37s 336ms/step - accuracy: 0.9334 - loss: 0.1778 - val_accuracy: 0.9639 - val_loss: 0.1086 - learn
ing_rate: 1.0000e-04
Epoch 10/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.1112 - val_accuracy: 0.8571 - val_loss: 0.2342 - learning
_rate: 1.0000e-04
Epoch 11/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 38s 343ms/step - accuracy: 0.9161 - loss: 0.2263 - val_accuracy: 0.9639 - val_loss: 0.1025 - learn
ing_rate: 1.0000e-04
Epoch 12/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 1.0000 - loss: 0.0097 - val_accuracy: 1.0000 - val_loss: 0.0317 - learning
_rate: 1.0000e-04
Epoch 13/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 38s 344ms/step - accuracy: 0.9345 - loss: 0.2157 - val_accuracy: 0.9591 - val_loss: 0.1088 - learn
ing_rate: 1.0000e-04
Epoch 14/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.8750 - loss: 0.2318 - val_accuracy: 0.8571 - val_loss: 0.2948 - learning
_rate: 1.0000e-04
Epoch 15/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 36s 320ms/step - accuracy: 0.9426 - loss: 0.1571 - val_accuracy: 0.9567 - val_loss: 0.0887 - learn
ing_rate: 1.0000e-04
Epoch 16/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 5ms/step - accuracy: 0.8750 - loss: 0.2043 - val_accuracy: 1.0000 - val_loss: 0.0159 - learning
_rate: 1.0000e-04
Epoch 17/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 35s 315ms/step - accuracy: 0.9471 - loss: 0.1506 - val_accuracy: 0.9688 - val_loss: 0.0810 - learn
ing_rate: 1.0000e-04
Epoch 18/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0497 - val_accuracy: 0.8571 - val_loss: 0.1531 - learning
_rate: 1.0000e-04
Epoch 19/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 37s 331ms/step - accuracy: 0.9568 - loss: 0.1450 - val_accuracy: 0.9567 - val_loss: 0.1141 - learn
ing_rate: 1.0000e-04
Epoch 20/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 1.0000 - loss: 0.0787 - val_accuracy: 0.9286 - val_loss: 0.1427 - learning
_rate: 1.0000e-04
Epoch 21/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 38s 349ms/step - accuracy: 0.9571 - loss: 0.1132 - val_accuracy: 0.9591 - val_loss: 0.1132 - learn
ing_rate: 1.0000e-04
Final Training Accuracy: 0.9513
Final Validation Accuracy: 0.9591
```

The accuracy obtained from this model is **95.91%** and has second best accuracy.

Implementation of Resnet+LSTM code, performing several epochs and its accuracy scores



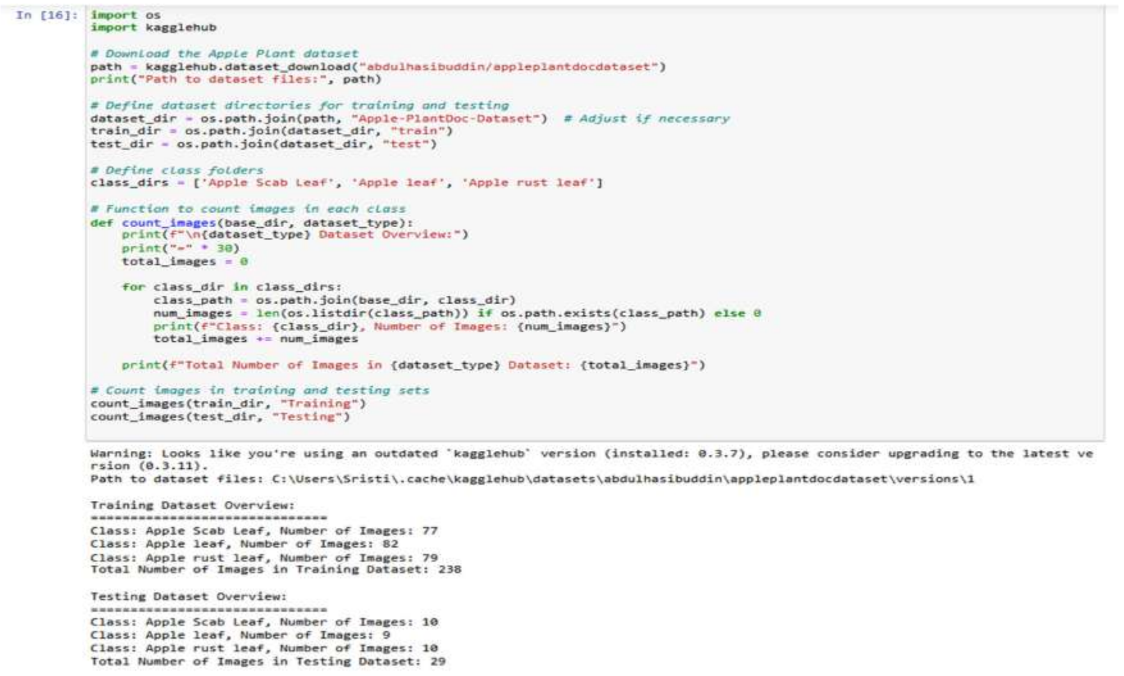
```
#  Data Generators
train_generator = train_datagen.flow_from_directory(
    dataset_dir,
    target_size=(64, 64),
    batch_size=16,
    class_mode='categorical',
    subset='training',
    classes=class_dirs
)
validation_generator = val_datagen.flow_from_directory(
    dataset_dir,
    target_size=(64, 64),
    batch_size=16,
    class_mode='categorical',
    subset='validation',
    classes=class_dirs
)

# Class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)
class_weights_dict = dict(enumerate(class_weights))

#  Base Model - ResNet50
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(64, 64, 3))
base_model.trainable = False

#  Add LSTM on top of ResNet Features
x = base_model.output                    # Output shape: (2, 2, 2048)
x = Reshape((4, 2048))(x)                # Correct reshape
x = LSTM(128, return_sequences=False)(x)
x = Dropout(0.5)(x)
outputs = Dense(3, activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=outputs)

#  Compile
model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy', metrics=['accuracy'])

#  Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('resnet_lstm_model.keras', monitor='val_loss', save_best_only=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=1e-6)

#  Train
history = model.fit(
    train_generator,
    steps_per_epoch=train_generator.samples // train_generator.batch_size,
    validation_data=validation_generator,
    validation_steps=validation_generator.samples // validation_generator.batch_size,
    epochs=50,
    class_weight=class_weights_dict,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)
```



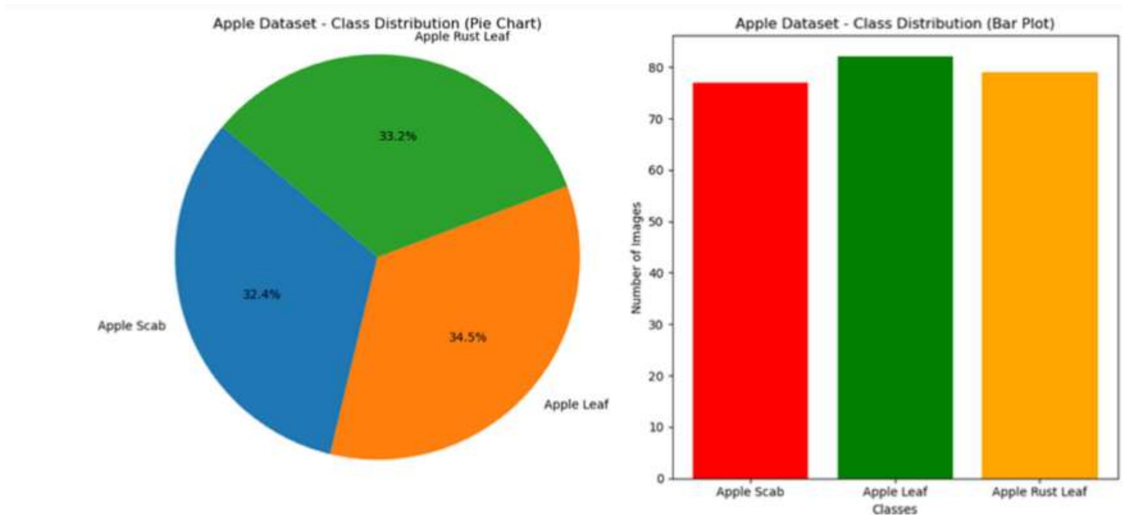
```
Epoch 6/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.1875 - loss: 0.8349 - val_accuracy: 0.3571 - val_loss: 1.0341 - learning
_rate: 1.0000e-04
Epoch 7/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 30s 270ms/step - accuracy: 0.4428 - loss: 1.0657 - val_accuracy: 0.6514 - val_loss: 0.9708 - learn
ing_rate: 1.0000e-04
Epoch 8/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.6875 - loss: 0.6417 - val_accuracy: 0.4286 - val_loss: 1.1005 - learning
_rate: 1.0000e-04
Epoch 9/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 33s 299ms/step - accuracy: 0.5085 - loss: 1.0430 - val_accuracy: 0.6587 - val_loss: 0.9396 - learn
ing_rate: 1.0000e-04
Epoch 10/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 19ms/step - accuracy: 0.5625 - loss: 0.9237 - val_accuracy: 0.6429 - val_loss: 0.9022 - learnin
g_rate: 1.0000e-04
Epoch 11/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 26s 235ms/step - accuracy: 0.5545 - loss: 1.0181 - val_accuracy: 0.5938 - val_loss: 0.9757 - learn
ing_rate: 1.0000e-04
Epoch 12/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 3s 24ms/step - accuracy: 0.7500 - loss: 0.8665 - val_accuracy: 0.7143 - val_loss: 0.8849 - learnin
g_rate: 1.0000e-04
Epoch 13/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 29s 260ms/step - accuracy: 0.4683 - loss: 1.0578 - val_accuracy: 0.5168 - val_loss: 1.0227 - learn
ing_rate: 1.0000e-04
Epoch 14/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 2s 21ms/step - accuracy: 0.3750 - loss: 0.7763 - val_accuracy: 0.7143 - val_loss: 0.8133 - learnin
g_rate: 1.0000e-04
Epoch 15/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 28s 257ms/step - accuracy: 0.6017 - loss: 0.9633 - val_accuracy: 0.5000 - val_loss: 1.0308 - learn
ing_rate: 1.0000e-04
Epoch 16/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 1s 3ms/step - accuracy: 0.4375 - loss: 0.9979 - val_accuracy: 0.6429 - val_loss: 0.8616 - learning
_rate: 1.0000e-04
Epoch 17/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 33s 304ms/step - accuracy: 0.5961 - loss: 0.9519 - val_accuracy: 0.4567 - val_loss: 1.0282 - learn
ing_rate: 1.0000e-04
Epoch 18/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step - accuracy: 0.4375 - loss: 1.4574 - val_accuracy: 0.4286 - val_loss: 1.0697 - learning
_rate: 1.0000e-04
Epoch 19/50
107/107 ━━━━━━━━━━━━━━━━━━━━ 24s 214ms/step - accuracy: 0.5414 - loss: 0.9702 - val_accuracy: 0.5601 - val_loss: 1.0106 - learn
ing_rate: 1.0000e-04
```

The accuracy obtained by Resnet+LSTM is **87%**, after performing 19 epochs.

Cross Validation with Apple Dataset

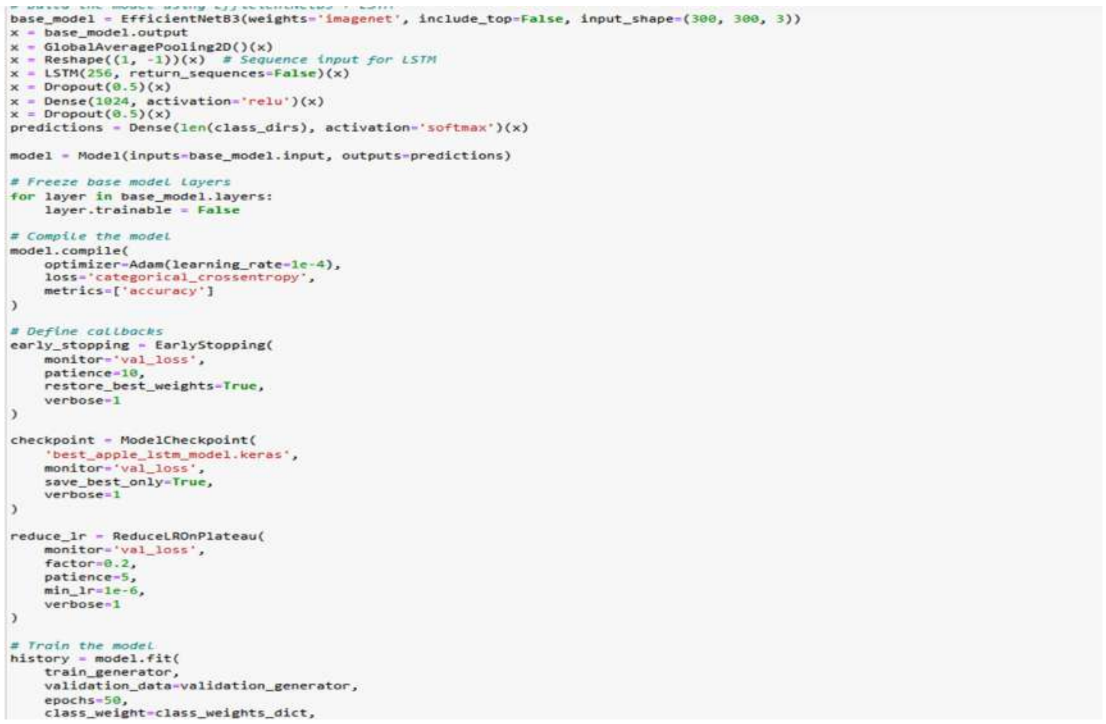


```
In [16]: import os
         import kagglehub

         # Download the Apple Plant dataset
         path = kagglehub.dataset_download("abdulhasibuddin/appleplantdocdataset")
         print("Path to dataset files:", path)

         # Define dataset directories for training and testing
         dataset_dir = os.path.join(path, "Apple-PlantDoc-Dataset")  # Adjust if necessary
         train_dir = os.path.join(dataset_dir, "train")
         test_dir = os.path.join(dataset_dir, "test")

         # Define class folders
         class_dirs = ['Apple Scab Leaf', 'Apple leaf', 'Apple rust leaf']

         # Function to count images in each class
         def count_images(base_dir, dataset_type):
             print(f"\n{dataset_type} Dataset Overview:")
             print("=" * 30)
             total_images = 0

             for class_dir in class_dirs:
                 class_path = os.path.join(base_dir, class_dir)
                 num_images = len(os.listdir(class_path)) if os.path.exists(class_path) else 0
                 print(f"Class: {class_dir}, Number of Images: {num_images}")
                 total_images += num_images

             print(f"Total Number of Images in {dataset_type} Dataset: {total_images}")

         # Count images in training and testing sets
         count_images(train_dir, "Training")
         count_images(test_dir, "Testing")

         Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
         rsion (0.3.11).
         Path to dataset files: C:\Users\Sristi\.cache\kagglehub\datasets\abdulhasibuddin\appleplantdocdataset\versions\1

         Training Dataset Overview:
         ==============================
         Class: Apple Scab Leaf, Number of Images: 77
         Class: Apple leaf, Number of Images: 82
         Class: Apple rust leaf, Number of Images: 79
         Total Number of Images in Training Dataset: 238

         Testing Dataset Overview:
         ==============================
         Class: Apple Scab Leaf, Number of Images: 10
         Class: Apple leaf, Number of Images: 9
         Class: Apple rust leaf, Number of Images: 10
         Total Number of Images in Testing Dataset: 29
```

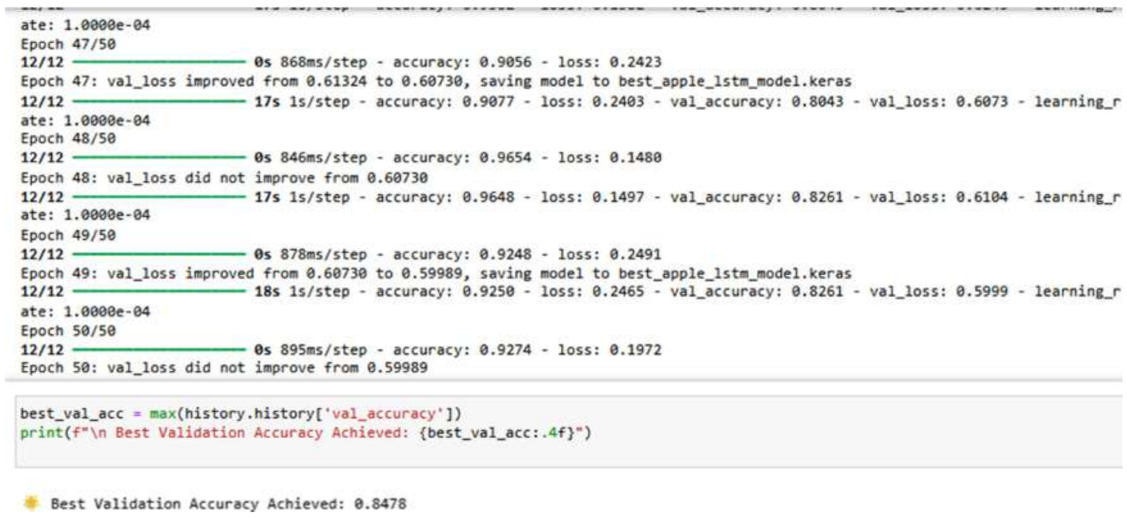


Importing dataset from Kaggle , this has **3 classes** apple scab, apple leaf, apple rust leaf. It is represented in form of pie and bar chart

Implementation of EfficientNet+LSTM code, performing several epochs and its accuracy scores for apple datasets



```
base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape=(300, 300, 3))
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Reshape((1, -1))(x)  # Sequence input for LSTM
x = LSTM(256, return_sequences=False)(x)
x = Dropout(0.5)(x)
x = Dense(1024, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(len(class_dirs), activation='softmax')(x)

model = Model(inputs=base_model.input, outputs=predictions)

# Freeze base model layers
for layer in base_model.layers:
    layer.trainable = False

# Compile the model
model.compile(
    optimizer=Adam(learning_rate=1e-4),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

# Define callbacks
early_stopping = EarlyStopping(
    monitor='val_loss',
    patience=10,
    restore_best_weights=True,
    verbose=1
)

checkpoint = ModelCheckpoint(
    'best_apple_lstm_model.keras',
    monitor='val_loss',
    save_best_only=True,
    verbose=1
)

reduce_lr = ReduceLROnPlateau(
    monitor='val_loss',
    factor=0.2,
    patience=5,
    min_lr=1e-6,
    verbose=1
)

# Train the model
history = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=50,
    class_weight=class_weights_dict,
```



```
ate: 1.0000e-04
Epoch 47/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 868ms/step - accuracy: 0.9056 - loss: 0.2423
Epoch 47: val_loss improved from 0.61324 to 0.60730, saving model to best_apple_lstm_model.keras
12/12 ━━━━━━━━━━━━━━━━━━━━ 17s 1s/step - accuracy: 0.9077 - loss: 0.2403 - val_accuracy: 0.8043 - val_loss: 0.6073 - learning_r
ate: 1.0000e-04
Epoch 48/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 846ms/step - accuracy: 0.9654 - loss: 0.1480
Epoch 48: val_loss did not improve from 0.60730
12/12 ━━━━━━━━━━━━━━━━━━━━ 17s 1s/step - accuracy: 0.9648 - loss: 0.1497 - val_accuracy: 0.8261 - val_loss: 0.6104 - learning_r
ate: 1.0000e-04
Epoch 49/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 878ms/step - accuracy: 0.9248 - loss: 0.2491
Epoch 49: val_loss improved from 0.60730 to 0.59989, saving model to best_apple_lstm_model.keras
12/12 ━━━━━━━━━━━━━━━━━━━━ 18s 1s/step - accuracy: 0.9250 - loss: 0.2465 - val_accuracy: 0.8261 - val_loss: 0.5999 - learning_r
ate: 1.0000e-04
Epoch 50/50
12/12 ━━━━━━━━━━━━━━━━━━━━ 0s 895ms/step - accuracy: 0.9274 - loss: 0.1972
Epoch 50: val_loss did not improve from 0.59989
```

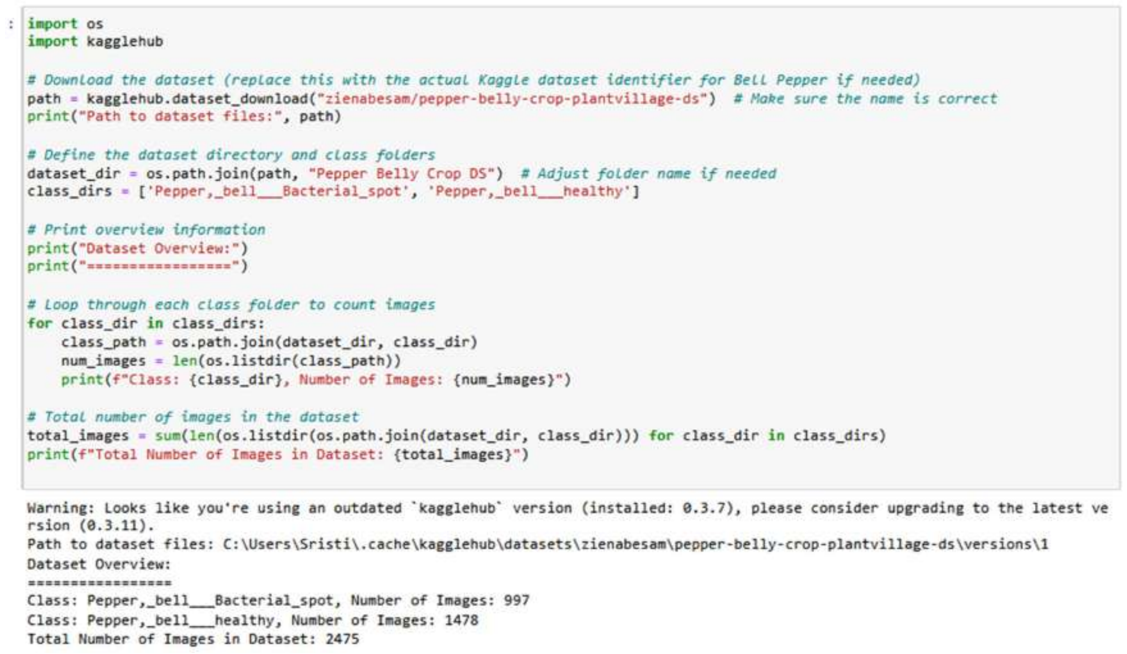
```python
best_val_acc = max(history.history['val_accuracy'])
print(f"\n Best Validation Accuracy Achieved: {best_val_acc:.4f}")
```

```
 Best Validation Accuracy Achieved: 0.8478
```

EfficientNet+LSTM is the best performing model for apple dataset **84.78%.**

Bell Pepper dataset

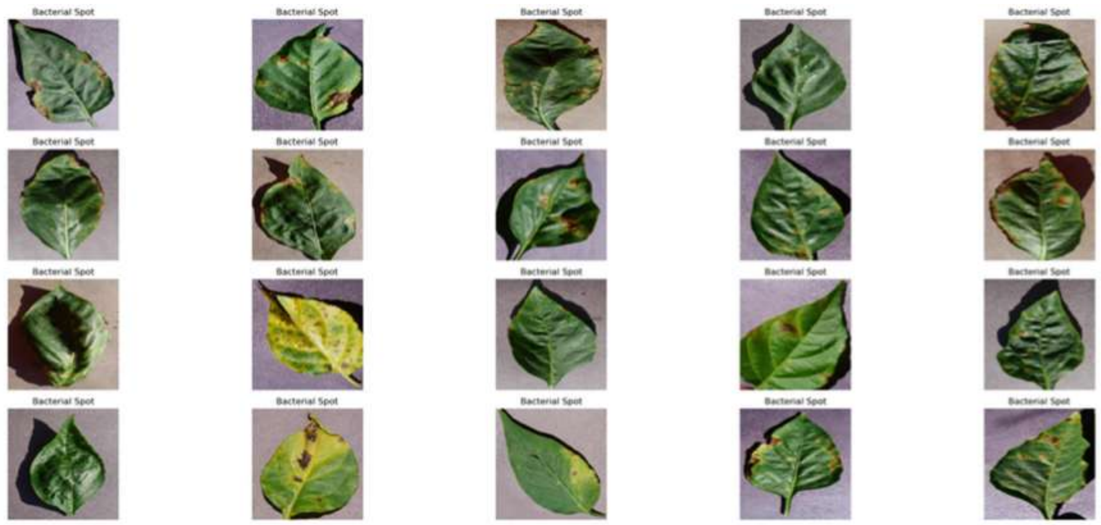
Bell pepper dataset is imported from kaggle,in total there are **238 images** with two classes , one is bacteria and other is healthy.



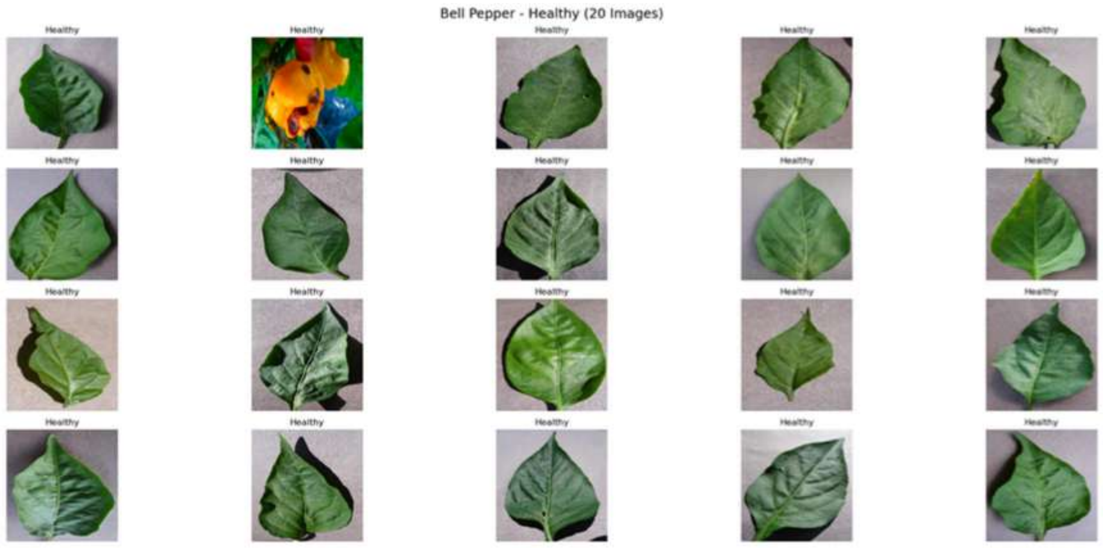
```python
import os
import kagglehub

# Download the dataset (replace this with the actual Kaggle dataset identifier for Bell Pepper if needed)
path = kagglehub.dataset_download("zienabesam/pepper-belly-crop-plantvillage-ds")  # Make sure the name is correct
print("Path to dataset files:", path)

# Define the dataset directory and class folders
dataset_dir = os.path.join(path, "Pepper Belly Crop DS")  # Adjust folder name if needed
class_dirs = ['Pepper,_bell___Bacterial_spot', 'Pepper,_bell___healthy']

# Print overview information
print("Dataset Overview:")
print("=================")

# Loop through each class folder to count images
for class_dir in class_dirs:
    class_path = os.path.join(dataset_dir, class_dir)
    num_images = len(os.listdir(class_path))
    print(f"Class: {class_dir}, Number of Images: {num_images}")

# Total number of images in the dataset
total_images = sum(len(os.listdir(os.path.join(dataset_dir, class_dir))) for class_dir in class_dirs)
print(f"Total Number of Images in Dataset: {total_images}")
```

```
Warning: Looks like you're using an outdated `kagglehub` version (installed: 0.3.7), please consider upgrading to the latest ve
rsion (0.3.11).
Path to dataset files: C:\Users\Sristi\.cache\kagglehub\datasets\zienabesam\pepper-belly-crop-plantvillage-ds\versions\1
Dataset Overview:
=================
Class: Pepper,_bell___Bacterial_spot, Number of Images: 997
Class: Pepper,_bell___healthy, Number of Images: 1478
Total Number of Images in Dataset: 2475
```

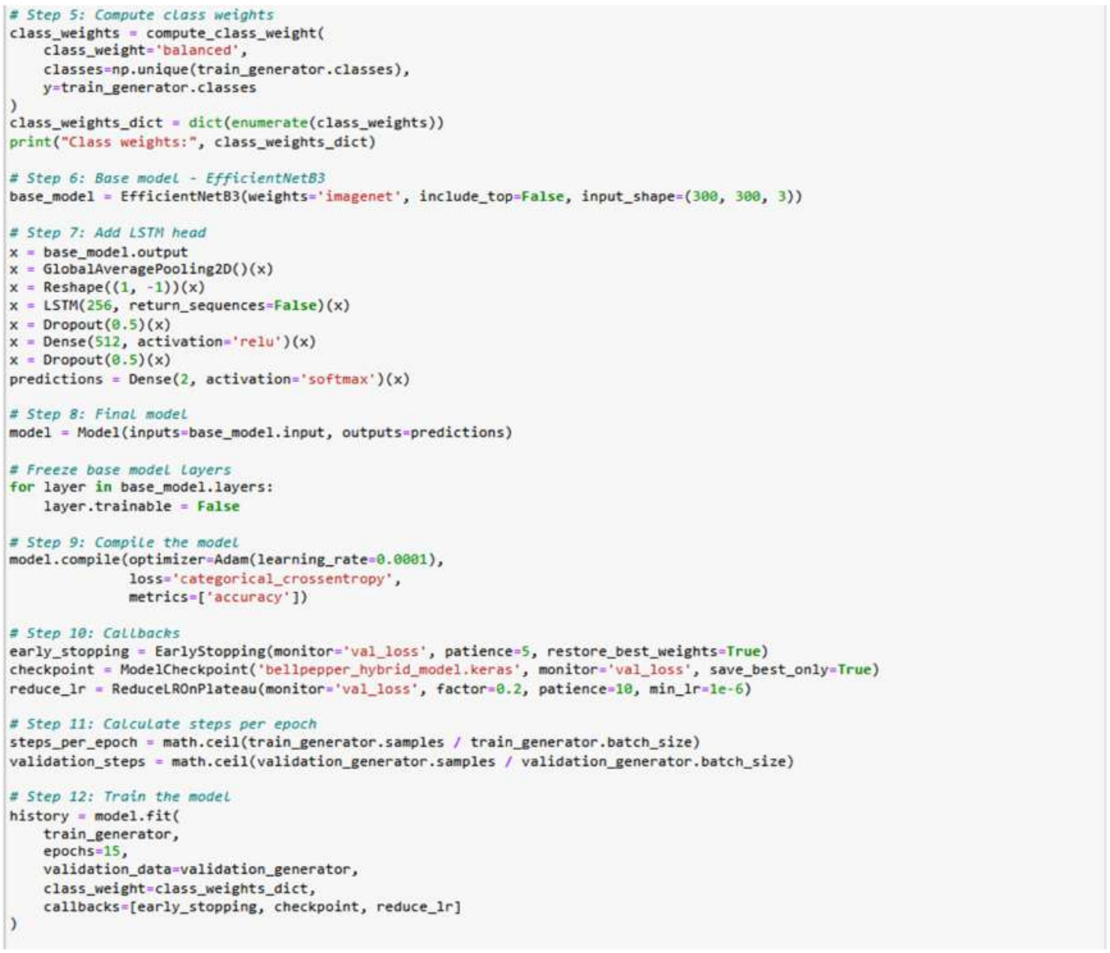
Bell pepper bacterial leaves images



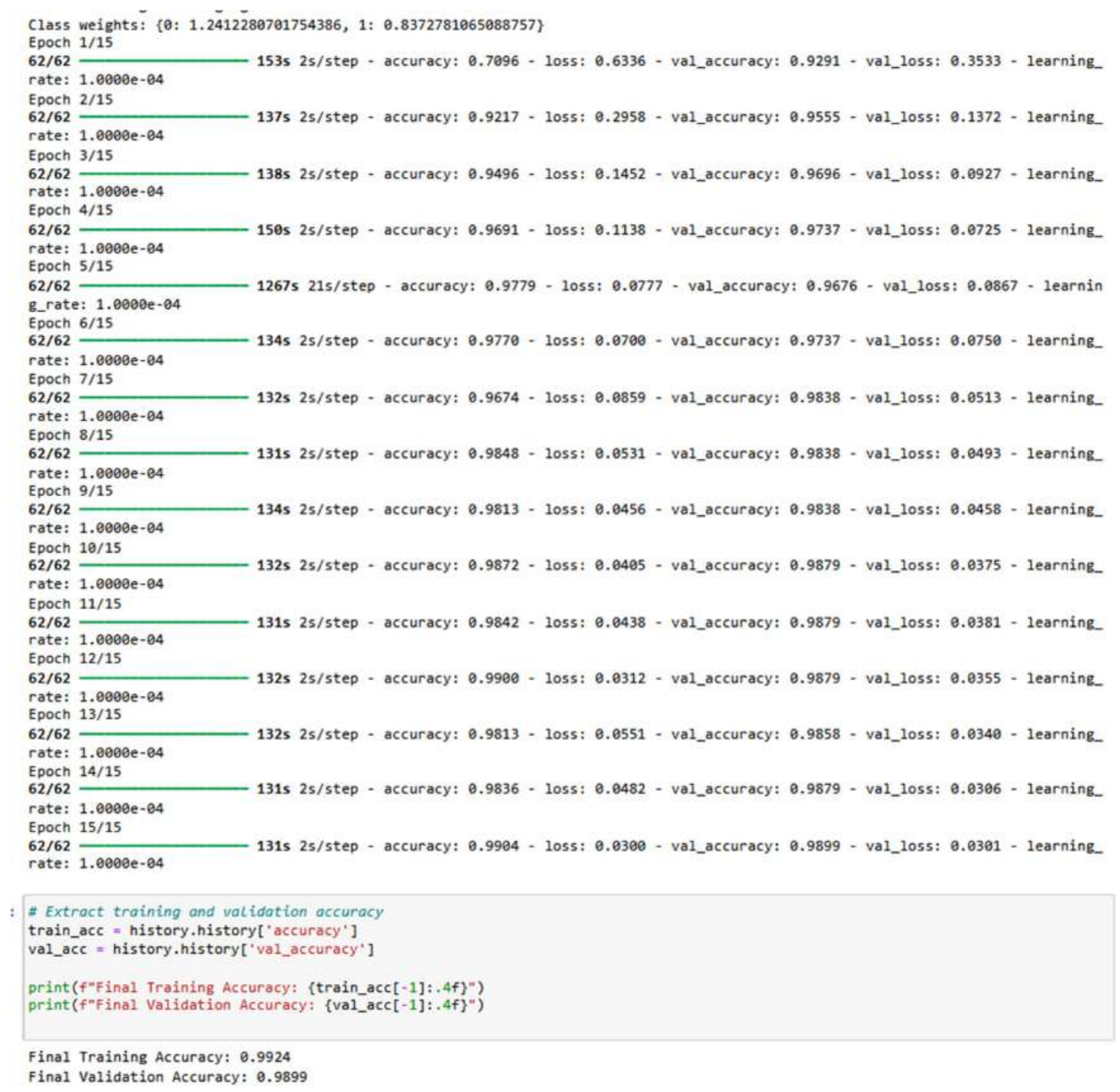
Bell pepper healthy leaves images



Implementation of EfficientNet+LSTM code, performing several epochs and its accuracy scores for Bell pepper dataset.



```
# Step 5: Compute class weights
class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_generator.classes),
    y=train_generator.classes
)
class_weights_dict = dict(enumerate(class_weights))
print("Class weights:", class_weights_dict)

# Step 6: Base model - EfficientNetB3
base_model = EfficientNetB3(weights='imagenet', include_top=False, input_shape=(300, 300, 3))

# Step 7: Add LSTM head
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Reshape((1, -1))(x)
x = LSTM(256, return_sequences=False)(x)
x = Dropout(0.5)(x)
x = Dense(512, activation='relu')(x)
x = Dropout(0.5)(x)
predictions = Dense(2, activation='softmax')(x)

# Step 8: Final model
model = Model(inputs=base_model.input, outputs=predictions)

# Freeze base model layers
for layer in base_model.layers:
    layer.trainable = False

# Step 9: Compile the model
model.compile(optimizer=Adam(learning_rate=0.0001),
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Step 10: Callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
checkpoint = ModelCheckpoint('bellpepper_hybrid_model.keras', monitor='val_loss', save_best_only=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=1e-6)

# Step 11: Calculate steps per epoch
steps_per_epoch = math.ceil(train_generator.samples / train_generator.batch_size)
validation_steps = math.ceil(validation_generator.samples / validation_generator.batch_size)

# Step 12: Train the model
history = model.fit(
    train_generator,
    epochs=15,
    validation_data=validation_generator,
    class_weight=class_weights_dict,
    callbacks=[early_stopping, checkpoint, reduce_lr]
)
```



```
Class weights: {0: 1.2412280701754386, 1: 0.8372781065088757}
Epoch 1/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 153s 2s/step - accuracy: 0.7096 - loss: 0.6336 - val_accuracy: 0.9291 - val_loss: 0.3533 - learning_
rate: 1.0000e-04
Epoch 2/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 137s 2s/step - accuracy: 0.9217 - loss: 0.2958 - val_accuracy: 0.9555 - val_loss: 0.1372 - learning_
rate: 1.0000e-04
Epoch 3/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 138s 2s/step - accuracy: 0.9496 - loss: 0.1452 - val_accuracy: 0.9696 - val_loss: 0.0927 - learning_
rate: 1.0000e-04
Epoch 4/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 150s 2s/step - accuracy: 0.9691 - loss: 0.1138 - val_accuracy: 0.9737 - val_loss: 0.0725 - learning_
rate: 1.0000e-04
Epoch 5/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 1267s 21s/step - accuracy: 0.9779 - loss: 0.0777 - val_accuracy: 0.9676 - val_loss: 0.0867 - learnin
g_rate: 1.0000e-04
Epoch 6/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 134s 2s/step - accuracy: 0.9770 - loss: 0.0700 - val_accuracy: 0.9737 - val_loss: 0.0750 - learning_
rate: 1.0000e-04
Epoch 7/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 132s 2s/step - accuracy: 0.9674 - loss: 0.0859 - val_accuracy: 0.9838 - val_loss: 0.0513 - learning_
rate: 1.0000e-04
Epoch 8/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 131s 2s/step - accuracy: 0.9848 - loss: 0.0531 - val_accuracy: 0.9838 - val_loss: 0.0493 - learning_
rate: 1.0000e-04
Epoch 9/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 134s 2s/step - accuracy: 0.9813 - loss: 0.0456 - val_accuracy: 0.9838 - val_loss: 0.0458 - learning_
rate: 1.0000e-04
Epoch 10/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 132s 2s/step - accuracy: 0.9872 - loss: 0.0405 - val_accuracy: 0.9879 - val_loss: 0.0375 - learning_
rate: 1.0000e-04
Epoch 11/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 131s 2s/step - accuracy: 0.9842 - loss: 0.0438 - val_accuracy: 0.9879 - val_loss: 0.0381 - learning_
rate: 1.0000e-04
Epoch 12/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 132s 2s/step - accuracy: 0.9900 - loss: 0.0312 - val_accuracy: 0.9879 - val_loss: 0.0355 - learning_
rate: 1.0000e-04
Epoch 13/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 132s 2s/step - accuracy: 0.9813 - loss: 0.0551 - val_accuracy: 0.9858 - val_loss: 0.0340 - learning_
rate: 1.0000e-04
Epoch 14/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 131s 2s/step - accuracy: 0.9836 - loss: 0.0482 - val_accuracy: 0.9879 - val_loss: 0.0306 - learning_
rate: 1.0000e-04
Epoch 15/15
62/62 ━━━━━━━━━━━━━━━━━━━━ 131s 2s/step - accuracy: 0.9904 - loss: 0.0300 - val_accuracy: 0.9899 - val_loss: 0.0301 - learning_
rate: 1.0000e-04
```

```
# Extract training and validation accuracy
train_acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

print(f"Final Training Accuracy: {train_acc[-1]:.4f}")
print(f"Final Validation Accuracy: {val_acc[-1]:.4f}")
```

```
Final Training Accuracy: 0.9924
Final Validation Accuracy: 0.9899
```

The Efficient+LSTM model performed the best for this dataset as well giving accuracy of **98%** and classifying the bacterial and healthy leaves effectively.