INTERNSHIP I/ DISSERTATION I REPORT

Submitted in partial fulfillment of the requirements for the degree of

Master of Technology Computer Science & Engineering

(Specialization in Artificial Intelligence and Machine Learning)

by

NIDHI SINGH

VIT®

Vellore Institute of Technology

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

November 2023

DECLARATION

I, **NIDHI SINGH** hereby declare that the thesis entitled **“PLANT DISEASES DETECTION USING DEEP LEARNING AND MACHINE VISION”** submitted to Vellore Institute of Technology (VIT), Vellore for the award of the degree of ***Master of Technology Computer Science & Engineering (Specialization in Artificial Intelligence and Machine Learning)*** is a record of bona fide work carried out by me under the supervision of **RAJKUMAR S**, Associate Professor Sr., School of Computer science and Engineering(SCOPE), Vellore Institute of Technology, Vellore.

I further declare that the work reported in this project report 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 instituteor any other institute or university.

NidhiSingh

Place: Vellore Signature of the Candidate

Date: 25/11/2023

CERTIFICATE

This is to certify that the thesis entitled **“PLANT DISEASES DETECTION USING DEEP LEARNING AND MACHINE VISION”** submitted by **NIDHI SINGH**, School of Computer Science and Engineering(SCOPE), Vellore Institute of Technology, Vellore for the award of the degree of ***Master of Technology Computer Science & Engineering (Specialization in Artificial Intelligence and Machine Learning)***, is a record of bona fide work carried out by her under my supervision, 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 dissertation fulfills the requirements and regulations of the Institute and in my opinion meets the necessary standards for submission.

[illegible]

Place: Vellore   Name and Signature of the Guide

Date: 25/11/2023

Signature of the HoD with seal   Signature of the School Dean with seal

ABSTRACT

This research presents a comprehensive study on the application of deep learning and machine vision techniques for the detection of plant leaf diseases in agricultural settings, specifically targeting farm village datasets. The dataset utilized in this study comprises both authentic farm village data and synthetic data generated using Generative Adversarial Networks (GANs). Three state-of-the-art convolutional neural network (CNN) models, namely VGG16 with transfer learning, ResNet50 with transfer learning, and InceptionNet V3, are employed for accurate and efficient disease classification. The proposed approach leverages the power of transfer learning to enhance the models' performance by fine-tuning pre-trained networks on the given datasets. The GAN-generated dataset serves as a valuable augmentation tool, addressing potential limitations associated with insufficient real-world data. The models are systematically trained and evaluated, considering key performance metrics such as accuracy, precision, recall, and F1 score. The results demonstrate the effectiveness of the proposed methodology in achieving robust and reliable plant leaf disease detection, showcasing the models' adaptability to both authentic and synthetic datasets. The incorporation of GAN-generated data aids in mitigating the challenges of data scarcity, enhancing the models' generalization capabilities. The comparative analysis of VGG16, ResNet50, and InceptionNet V3 models reveals nuanced insights into their respective strengths and weaknesses in the context of plant disease detection. ResNet50 emerged as the standout performer, achieving the highest accuracy at 83.23% in compared to VggNet16 with 79.2% and InceptionNet V3 with 76.4%. This research contributes to the burgeoning field of precision agriculture by presenting a comprehensive and technically robust framework for plant leaf disease detection. The incorporation of advanced CNN architectures, coupled with transfer learning and GAN-augmented datasets, exemplifies a novel approach to addressing the nuanced challenges inherent in agricultural disease identification. The findings of this study hold promise for the development of efficient and accurate tools that can significantly impact crop yield optimization and contribute to sustainable farming practices.

Keywords: *Plant leaf diseases, Convolutional neural networks, Transfer learning, Generative Adversarial Networks, Synthetic data.*

ACKNOWLEDGEMENTS

With immense pleasure and a deep sense of gratitude, I wish to express my sincere thanks to my supervisor **RAJKUMAR S**, Associate Professor Sr., School of Computer science and Engineering(SCOPE), Vellore Institute of Technology (VIT), Vellore without his/her motivation and continuous encouragement, this research would not have been successfully completed.

I am grateful to the Chancellor of VIT, **Dr. G. Viswanathan**, the Vice Presidents, and the Vice Chancellor for motivating me to carry out research in the Vellore Institute of Technology.

It would be no exaggeration to say that the Dean of SCOPE, **Prof Ramesh Babu K**, was always available to clarify any queries and clear the doubts I had during the course of my project.

I would also like to acknowledge the role of the HoD, **Dr. ANANDA KUMAR S**, who was instrumental in keeping me updated with all necessary formalities and posting all the required formats and document templates through the mail, which I was glad to have had.

Finally, I would like to thank **Vellore Institute of Technology** for providing me with infrastructural facilities, a flexible choice and for supporting my research and execution related to the dissertation work.

Place: Vellore **NAME OF THE SCHOLAR**

NIDHI SINGH

Date: 25/11/2023

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

CNN -	Convolutional Neural Network
VGG -	Visual Geometry Group
RMSProp -	Root Mean Square Propagation
Adam -	Adaptive Moment Estimation
SGD -	Stochastic Gradient Descent
ReLU -	Rectified Linear Unit
Dropout -	Dropout Regularization
TRAINGEN -	Training data generator
VALIDGEN -	Validation data generator
OBJECTIVE_FUNCTION -	Loss function used for model training
LOSS_METRICS -	Metrics used for model evaluation
cb_checkpointer -	ModelCheckpoint callback for saving the best weights
cb_early_stopper -	EarlyStopping callback for preventing overfitting
InceptionV3 -	Inception Version 3
ResNet50 -	Residual Network with 50 layers
Dense -	Fully Connected Layer in a neural network
GlobalAveragePooling2D -	Global Average Pooling 2D layer in a neural network
Flatten	Flattening layer in a neural network
Softmax -	Softmax activation function
ML -	Machine Learning
DL -	Deep Learning
AI -	Artificial Intelligence
GPU -	Graphics Processing Unit
CPU -	Central Processing Unit

Chapter 1

INTRODUCTION

1.1 Overview

In modern agriculture, crop health and yield optimization are contingent upon the prompt and precise diagnosis of plant diseases. Particularly, plant leaf diseases represent a significant risk to agricultural productivity, calling for creative methods of early identification and control. The vulnerability of crops to diseases, especially those that impact plant leaves, poses a significant risk to the security of food supply worldwide. It is becoming more and more important to maximize crop productivity as the world's population grows. Conventional disease detection techniques are prone to subjectivity, limited scalability, and human error because they mostly rely on manual inspection by agronomists. In order to tackle these obstacles, this study explores the joint application of deep learning and machine vision for the automated identification of plant leaf diseases, with a focus on the complex environment of agricultural communities. This study explores the field of plant leaf disease detection by combining machine vision and deep learning methods in a synergistic way. The study takes into account the complexities of actual agricultural settings by placing itself in the context of farm settlements. In order to address the varied and dynamic nature of plant diseases in farm village environments, the application of state-of-the-art convolutional neural network (CNN) architectures, specifically VGG16 with transfer learning, ResNet50 with transfer learning, and InceptionNet V3, aims to propel the accuracy and efficiency of disease detection. Agronomists' visual inspection is a common component of traditional plant disease detection techniques, but it is also subject to subjectivity and scale constraints. A new age in plant disease identification has been brought about by the development of deep learning and machine vision, which offers the possibility of automated, quick, and precise diagnosis. Convolutional neural networks—which are optimized for image classification tasks—have shown impressive results in a number of fields, such as autonomous vehicles and medical imaging. Deep learning's application to plant disease identification offers a chance to completely transform crop health monitoring and management in the agricultural sector. Farm villages pose a distinct challenge to disease detection models due to their varied crop varieties, environmental factors, and agricultural practices. A flexible and strong strategy is necessary due to the wide variation in plant species and disease presentations. Moreover, developing accurate models is hampered by the lack of labeled data unique to farm settlements. To tackle these issues, this study makes use of transfer learning, a method that refines pre-trained models on extensive datasets for the particular purpose of identifying plant leaf diseases. The models are better able to identify subtle patterns in data peculiar to agricultural villages thanks to this knowledge transfer, which enables the models to inherit features learnt from varied datasets. The use of machine learning and deep learning in agriculture signals a paradigm change in disease detection techniques. Convolutional Neural Networks (CNNs) are a subclass of deep learning models that are specifically tailored for image classification. They have shown impressive performance across a range of applications, including autonomous vehicles and medical imaging. The use of CNNs in agriculture to identify plant diseases is a powerful way to transform crop management and monitoring. The main issue is creating models that can quickly and accurately identify diseases, which is a difficult undertaking given how dynamic agricultural surroundings are.

These environments are defined by a patchwork of different crops, variable weather patterns, and a variety of farming techniques. The predominance of many plant species and disease presentations calls for a flexible and robust solution. Moreover, creating precise models is made more difficult by the lack of labeled data unique to agricultural settlements. Here comes transfer learning, a method that refines pre-trained models on massive datasets for the particular purpose of identifying plant leaf diseases. This knowledge transfer enhances the models' capacity to identify subtle patterns in data specific to farm villages by infusing them with features from a variety of datasets.

1.2 Objectives

The aim of this study is to progress the current understanding of plant leaf disease detection in the intricate and ever-changing environment of rural communities. This study's main goal is to develop, test, and assess deep learning models for plant leaf disease detection in rural communities. The CNN architectures ResNet50, InceptionNet V3, and VGG16 that were selected are well known for their effectiveness in image classification applications. Transfer learning is used to use the knowledge from large datasets that is encoded in these models, making it easier to adapt them to the particular features of farm village datasets. Additionally, the study uses GAN-generated data to supplement the training set, which improves the models' ability to generalize well in practical agricultural contexts. Acknowledging the vital significance of early disease detection in agricultural environments, the study endeavors to use the potential of deep learning and machine vision. The research specifically focuses on three well-known convolutional neural network (CNN) designs that have been enhanced using transfer learning techniques: VGG16, ResNet50, and InceptionNet V3. The goal of the project is to improve plant leaf disease detection in agricultural communities through the use of these sophisticated models, with the ultimate goal of enabling precision agriculture. The two main goals of this research are to: (1) create and apply deep learning models for plant leaf disease detection in farm villages; and (2) assess how well three different CNN architectures—ResNet50, VGG16, and InceptionNet V3—perform in tackling the particular difficulties presented by datasets from farm villages. A key component of this strategy is transfer learning, which enables pre-trained models to be adjusted to the peculiarities of data unique to agricultural villages. Furthermore, the research incorporates the novel application of Generative Adversarial Networks (GANs) to produce artificial data, enhancing the training set and strengthening the models' ability to generalize successfully in actual agricultural situations.

Chapter 2

LITERATURE REVIEW

2.1 Background work

Previous research on plant disease detection highlights a shift from traditional to technologically sophisticated approaches in the literature review. Conventional methods were labor- and time-intensive, depending on visual inspection and laboratory techniques. The combination of computer vision and machine learning causes a paradigm shift in the modern world. Images of plants can be used to extract important information, and machine learning algorithms can improve the accuracy of disease classification. In order to increase the resilience of detection models, researchers stress the investigation of several variables, such as leaf texture and color. This change in approach represents a significant advancement toward prompt, automated, and effective interventions in agricultural operations that solve long-standing problems with disease identification.

2.2 Review of Literature

While the study "Plant Disease Detection Using Image Processing and Machine Learning" by Pranesh Kulkarni et al. demonstrates a commendable 93% accuracy, limitations should be acknowledged[1]. These include potential challenges in generalizing results to new datasets and varying environmental conditions. The dynamic nature of plant diseases and computational demands in large-scale agriculture may impact the model's effectiveness. Additionally, reliance on image-based detection may overlook other influential factors, such as weather conditions or soil variations. Recognizing and addressing these limitations is crucial for ensuring the practical applicability and robustness of the proposed system in diverse agricultural settings.

The study titled "Plant Disease Identification Based on Deep Learning Algorithm in Smart Farming" by Yan Guo et al. introduces a mathematical model utilizing deep learning for efficient plant disease detection[2]. Researchers employ the Region Proposal Network (RPN) for leaf localization and the Chan–Vese (CV) algorithm for leaf segmentation. The segmented leaves are then input into a transfer learning model, trained with a dataset of diseased leaves. The proposed model exhibits an accuracy of 83.57%, surpassing traditional methods, and is particularly effective against black rot, bacterial plaque, and rust diseases. Despite promising results, challenges include the iterative nature of the Chan–Vese algorithm, suggesting potential

improvements for faster identification in future research.

This research title as "Deep Learning-Based Leaf Disease Detection in Crops Using Images for Agricultural Applications," conducted by Andrew J. el at. emphasizes the critical role of the agricultural sector and the impact of plant diseases on food production[3]. The paper explores the application of convolutional neural network (CNN)-based pre-trained models, including DenseNet-121, ResNet-50, VGG-16, and Inception V4, for efficient plant disease identification. Using the PlantVillage dataset, the experiments reveal DenseNet-121's superior performance, achieving a remarkable classification accuracy of 99.81%. The study underscores the potential of deep learning in enhancing early diagnosis of plant diseases, contributing to improved food quality and reduced economic losses. Future work aims to address real-time data collection challenges and develop a multi-object deep learning model, with plans for implementing a mobile application for real-time leaf disease identification in agriculture.

Sharada P. Mohanty's et al. study on deep learning-based plant disease detection highlights impressive accuracy of 99.35%, utilizing convolutional neural networks on the PlantVillage dataset[4]. However, limitations include reduced accuracy under varied testing conditions and constraints to classifying single leaves on homogeneous backgrounds. The study acknowledges the need for more diverse training data and realistic image collection to improve generalization. Notably, the proposed approach aims to supplement, not replace, laboratory tests, recognizing challenges in early-stage diagnosis based solely on visual symptoms. Ongoing research aims to address these drawbacks for practical and effective real-world applications in agriculture.

The study by Riyao Chen et al. introduces CACPNET as a promising model for plant disease identification, yet it acknowledges some limitations[5]. While achieving high accuracy, the attention mechanism may not be specifically designed for plant diseases, potentially leading to identification errors. The trade-off between accuracy and the real-time demands of deployment is discussed, with attention to the challenge of maintaining accuracy while reducing model complexity through channel pruning. Additionally, the study highlights the importance of setting a reasonable local compression ratio for effective model pruning. Despite these limitations, CACPNET demonstrates notable advantages, emphasizing its potential for lightweight deployment and promoting artificial intelligence in precision agriculture.

The literature survey titled "Plant Disease Detection Using Deep Learning" by Kowshik B et al. explores the significance of agriculture and the lack of traditional technologies for disease detection, leading to decreased agricultural productivity[6]. The study introduces a method employing convolutional neural networks and deep neural networks to identify crop disease symptoms, emphasizing the potential of deep learning for improved accuracy. While highlighting the positive impact on early disease detection, the review acknowledges the need for further research to address existing gaps in disease detection transparency and proposes future expansions for additional features like pesticide price lists and market information. However, specific drawbacks or limitations are not explicitly outlined in the provided information.

This research title as "Deep Learning-Based Leaf Disease Detection in Crops Using Images for Agricultural Applications" by Andrew J. et al[7]. focuses on enhancing plant disease identification using deep learning, particularly convolutional neural network (CNN) models. Four pre-trained models—DenseNet-121, ResNet-50, VGG-16, and Inception V4—are fine-tuned to improve accuracy in identifying 38 different classes of plant diseases. The experiments utilize the PlantVillage dataset, containing 54,305 images across 38 classes. DenseNet-121 stands out with a remarkable 99.81% classification accuracy, surpassing state-of-the-art models. The study also emphasizes the potential deployment of the proposed model through a mobile application, facilitating real-time leaf disease identification in agriculture. Further improvements and extensions are suggested for comprehensive application across diverse crops and environments.

2.3 Summary

Studies on plant disease detection highlight varying methodologies and challenges. Kulkarni et al. note challenges in generalization and adapting to diverse environments. Guo et al. use Region Proposal Network and Chan–Vese algorithm with segmentation challenges. Andrew J. et al. focus on DenseNet-121's potential, emphasizing real-time data collection challenges. Mohanty et al. acknowledge limitations in varied testing conditions and the supplementary role of laboratory tests. Chen et al.'s CACPNET addresses challenges in the attention mechanism and real-time deployment. Kowshik B et al.'s literature survey underscores the significance of deep learning in plant disease detection, emphasizing improved accuracy while recognizing the need for further research in transparency and expanded features in agriculture.

Chapter 3

METHODOLOGY

3.1 Brief Introduction

There are numerous important steps in the process of developing a deep learning model for image classification utilizing transfer learning with VGG16, ResNet50, and InceptionV3. The dataset is first loaded with a description of its properties and separated into test, validation, and training sets. The training set can be made more diverse by using data augmentation techniques, and batch processing is made more effective by implementing a data loader. The next step is picture preparation, which includes normalizing pixel values and scaling photos to the desired input size. Choosing a pre-trained model, such as VGG16, ResNet50, or InceptionNet V3, and removing its classification head to leave only the convolutional base constitute the first step in the feature extraction process. On top of the previously trained convolutional foundation, a new classification head tailored to the particular task is added to create the custom model. After that, the model is compiled using the proper metrics, loss function, and optimizer. Convolutional layers may be frozen to maintain pre-trained weights during the training phase, and the model is trained using the training set while being monitored by the validation set. Training efficiency can be increased by using early stopping approaches and learning rate schedules. The model's performance is assessed on the test set after training.

With optional post-processing procedures like thresholding or filtering, the trained model is used to generate predictions on fresh data. Another approach for fine-tuning the model is to unfreeze the layers that have already been learned in order to continue training and improve task-specific performance. The trained model may also be deployed for inference in systems or applications, or it may be stored for later use. This thorough technique offers an organized strategy for creating and implementing deep learning models for image classification that are flexible enough to adjust to the unique features of each dataset and the demands of the task.

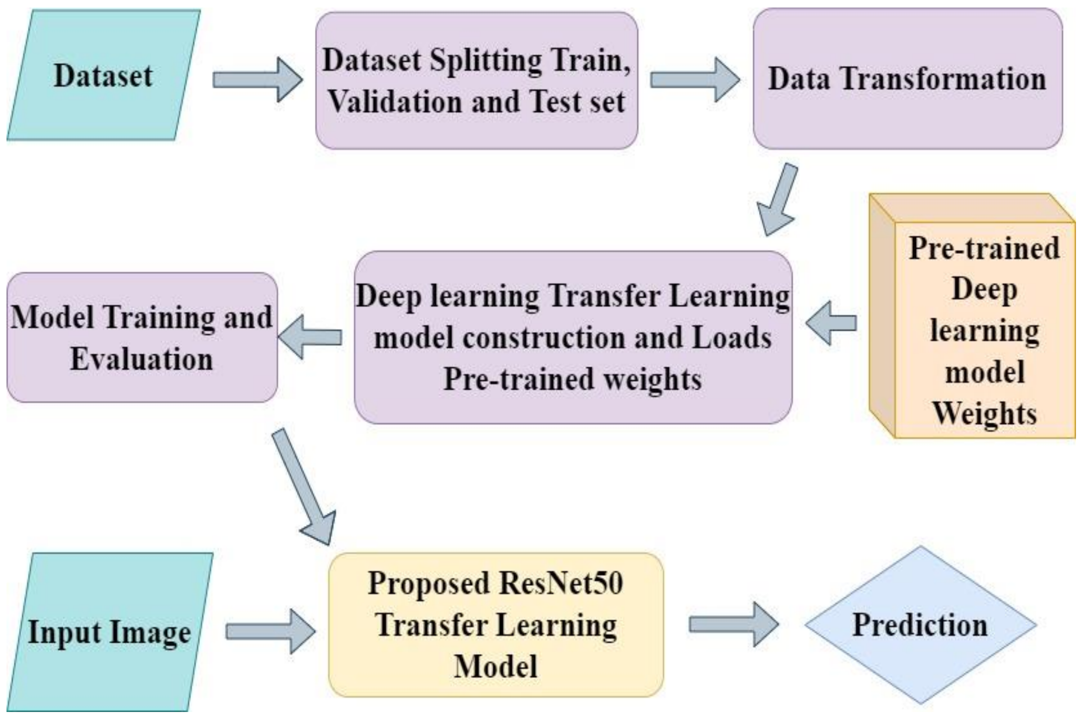

Dataset
Dataset Splitting Train, Validation and Test set
Data Transformation
Pre-trained Deep learning model Weights
Deep learning Transfer Learning model construction and Loads Pre-trained weights
Model Training and Evaluation
Input Image
Proposed ResNet50 Transfer Learning Model
Prediction

Fig. 3.1 Proposed Methodology

3.2 Architecture of the proposed system

The VGG16, ResNet50, and InceptionV3 architectures are used in the proposed plant disease detection system to exploit transfer learning, with each serving as a pre-trained backbone for feature extraction. To adapt the pre-trained models to the particular goal of recognizing plant diseases, a new classification head is added. The output layer is set up to reflect the number of illness classes in the dataset, while the input layer is set up to match the necessary image dimensions. Important hyper parameters like batch size and learning rate are selected with care, and the model is assembled using suitable optimization algorithms and loss functions. Convolutional layers may be frozen during training in order to preserve the pre-trained weights, and the model is adjusted to improve performance on particular tasks. The architecture's robustness and generalizability are enhanced by its capacity to adapt to changes in the environment and among various plant species. Clear overviews offered by visual representations of the model architecture facilitate understanding. The overall goal of the suggested approach is to combine the advantages of transfer learning with cutting-edge pre-trained models to overcome shortcomings in the current plant disease detection techniques.

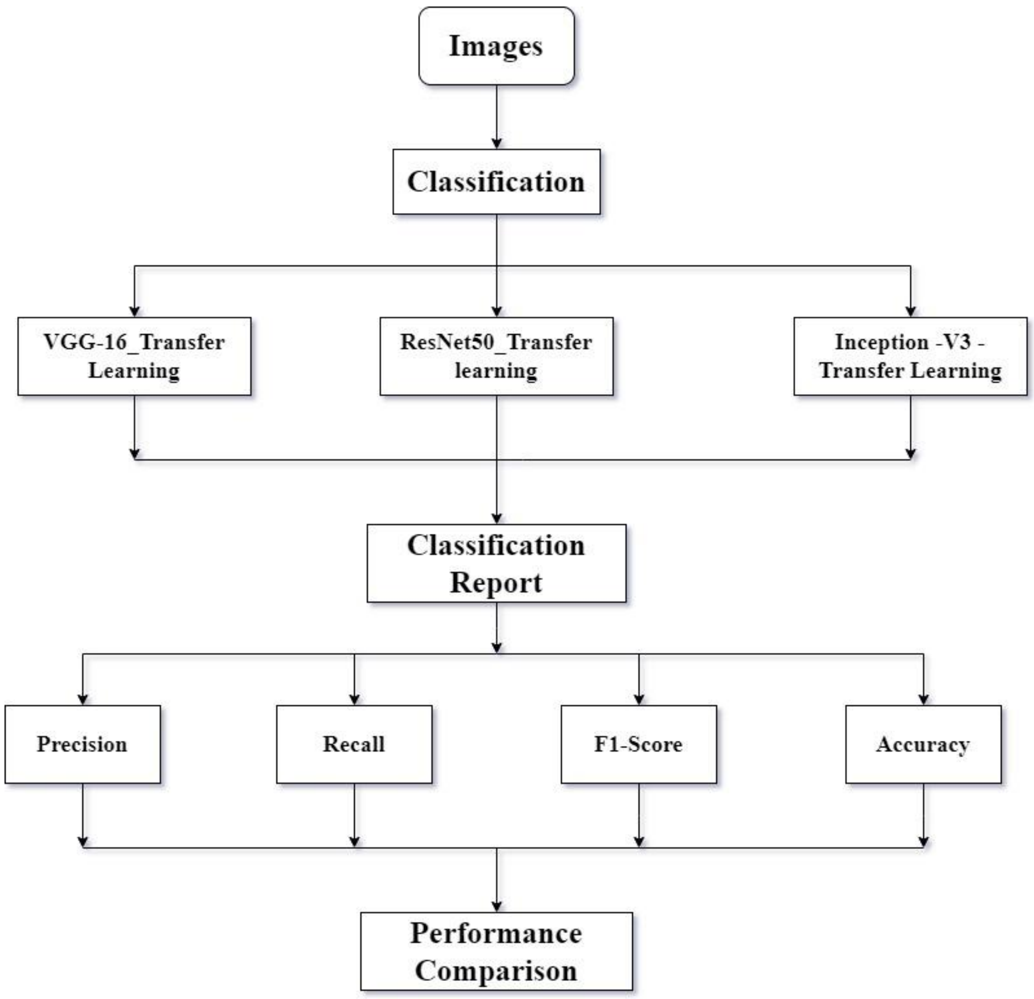


Fig. 3.2 Proposed Architecture

3.2.1 Limitations of Existing System

Numerous flaws in the current plant disease detection method make it less effective in producing results that are trustworthy and accurate. The most significant of these difficulties is the limitation of a small dataset, which frequently lacks the diversity required to fully represent the range of plant diseases. The model's generalizability is hampered by this shortcoming, especially with regard to novel or emerging illnesses or diverse plant species and variations. Robust performance in real-world agricultural settings may also be hampered by the model's sensitivity to environmental factors like fluctuating illumination and humidity levels. Misclassifications may result from interclass confusion, a condition in which it is difficult to distinguish between outwardly identical healthy and ill plants. Further contributing to its limits include the model's reliance on high-quality photographs and the computational resource requirements of the current system, which may make implementation difficult in contexts with limited resources. Consideration should also be given to real-time detection capabilities, scalability issues, and difficulties in transferring across various areas and farming techniques. Moreover, the model's adoption may be hampered by its inability to explain the decision-making process, especially in situations where stakeholders demand transparency. All of these drawbacks highlight the necessity of developing and refining the plant disease detection system in order to overcome these obstacles and increase its usefulness.

3.2.2 Proposed system

Using pre-trained convolutional neural network architectures—VGG16, ResNet50, and InceptionV3—as feature extractors, the suggested plant disease detection system uses a transfer learning methodology. Each pre-trained model has an attachment called a bespoke classification head that makes it specific to the task of classifying plant diseases. The system improves upon the limitations of the current model by integrating a larger and more varied dataset, which increases the model's capacity to generalize across different plant species and environmental circumstances. The architecture is made to reduce problems like confusion between classes and enhance real-time detection capabilities. To get the best possible model performance, important hyper parameters such as batch size and learning rate are carefully adjusted.

A method is used to selectively freeze pre-trained convolutional layers during training, maintaining learnt features and enabling fine-tuning to adjust to the unique characteristics of plant diseases. The suggested solution is made to be scalable by supporting an expanding dataset and being able to easily manage a growing number of illness types. The interpretability of the model architecture is improved by visual representations. Explainability is given a lot of weight in this system, which also includes tools to bring transparency to decision-making processes and build user and stakeholder trust.

Technically speaking, the suggested plant disease detection system combines state-of-the-art transfer learning techniques with a specially designed architecture to overcome current constraints. It seeks to offer a more accurate, flexible, and comprehensible remedy for the prompt and trustworthy identification of plant diseases in a range of farming situations.

3.2.2.1 Architecture of the ResNet50 Transfer Learning model

The ResNet50 convolutional neural network, a powerful deep learning architecture pre-trained on the ImageNet dataset, serves as the foundation for the architecture of the implemented model. To repurpose it as a feature extractor, the top layers of the ResNet50 basis are carefully excluded before the model is integrated. A Global Average Pooling (GAP) layer, which efficiently condenses the spatial dimensions of the retrieved features, comes after this convolutional basis. The final classification layer for classifying images into a predetermined number of classes is then added to the model: a Dense layer with a Softmax activation function. Interestingly, the ResNet50 base weights are designated as non-trainable, which preserves the important information learned during the pre-training phase on ImageNet. By making this calculated move, the rich hierarchical features present in the pre-trained convolutional basis are certain to help the model. Stochastic Gradient Descent (SGD) is used as the optimizer to compile the model, and parameters like learning rate, decay, momentum, and nesterov are carefully adjusted. Given that the task involves multi-class classification, categorical crossentropy is selected as the loss function, and pertinent evaluation metrics are included to assess model performance. Using training and validation data from generators (traingen and validgen), the model iteratively adjusts its weights during the training phase. Two critical callbacks oversee the training process: cb_checkpointer, which maintains the optimal model weights based on validation loss, and cb_early_stopper, which mandates an early termination in the event that the validation loss does not show improvement within the designated patience. After training, the model is preserved for possible deployment or further applications by being saved to a specified file directory.

In conclusion, this architecture creates a strong and flexible model for the particular job of plant disease detection by skillfully fusing the advanced feature extraction capabilities of the ResNet50 model with a customized classification head. The model is a reliable option for precise and effective picture categorization since it may use ImageNet's prior knowledge in conjunction with well-tuned training.

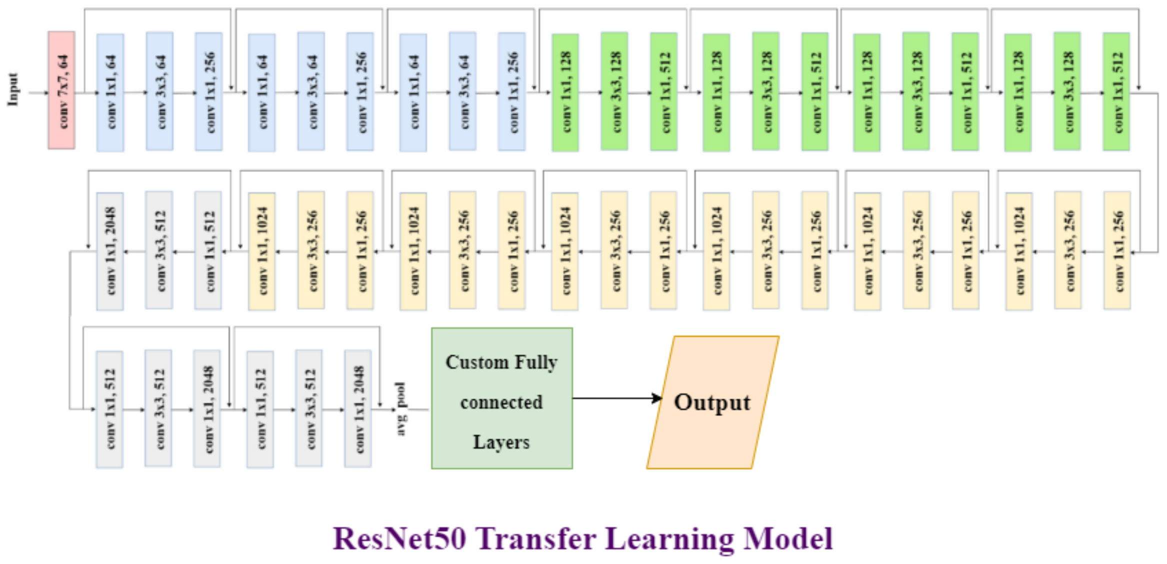


Fig. 3.2.2.1 Architecture of the ResNet50 Transfer Learning Model

3.2.2.2 Architecture of the VggNet16 Transfer Learning model

The VGG16 convolutional neural network, modified for the classification of plant diseases, serves as the foundation for the architecture of the model that is being shown. By applying weights acquired from the ImageNet dataset to the pre-trained VGG16 model, the algorithm integrates a transfer learning technique. Without its fully linked layers, the convolutional base is included, freeing up space for a specially designed classification head for that particular task. The architecture of the model allows for selective training of extra layers on top of the base model by providing fine-tuning choices depending on the fine tune parameter. A flattening layer in the custom classification head converts the convolutional base's output into a one-dimensional tensor. Two highly connected layers with activation functions of rectified linear units (ReLUs) (4096 and 1072 units, respectively) come next. A dropout layer with a 0.2 dropout rate is added to reduce overfitting. The last layer generates class probabilities for the designated number of disease classes using a softmax activation function. The model as a whole is assembled through the use of the categorical crossentropy loss function, an optimizer designated by the optimizer parameter (which is initially set to 'rmsprop'), and accuracy as the assessment measure. A thorough description of the architecture, including information on the layers, their output forms, and the total number of trainable parameters, is given in the model summary. The approach uses early halting callbacks and model checkpointing during training, and it lowers the learning rate to 0.0001 to achieve better convergence.

The resulting model is trained on the traingen and validgen generators for a predetermined number of epochs, and the training progress is displayed via a custom plot callback. The architecture is intended to improve accuracy and adaptability in the field of plant disease detection. It is distinguished by the combination of transfer learning and a refined classification head.

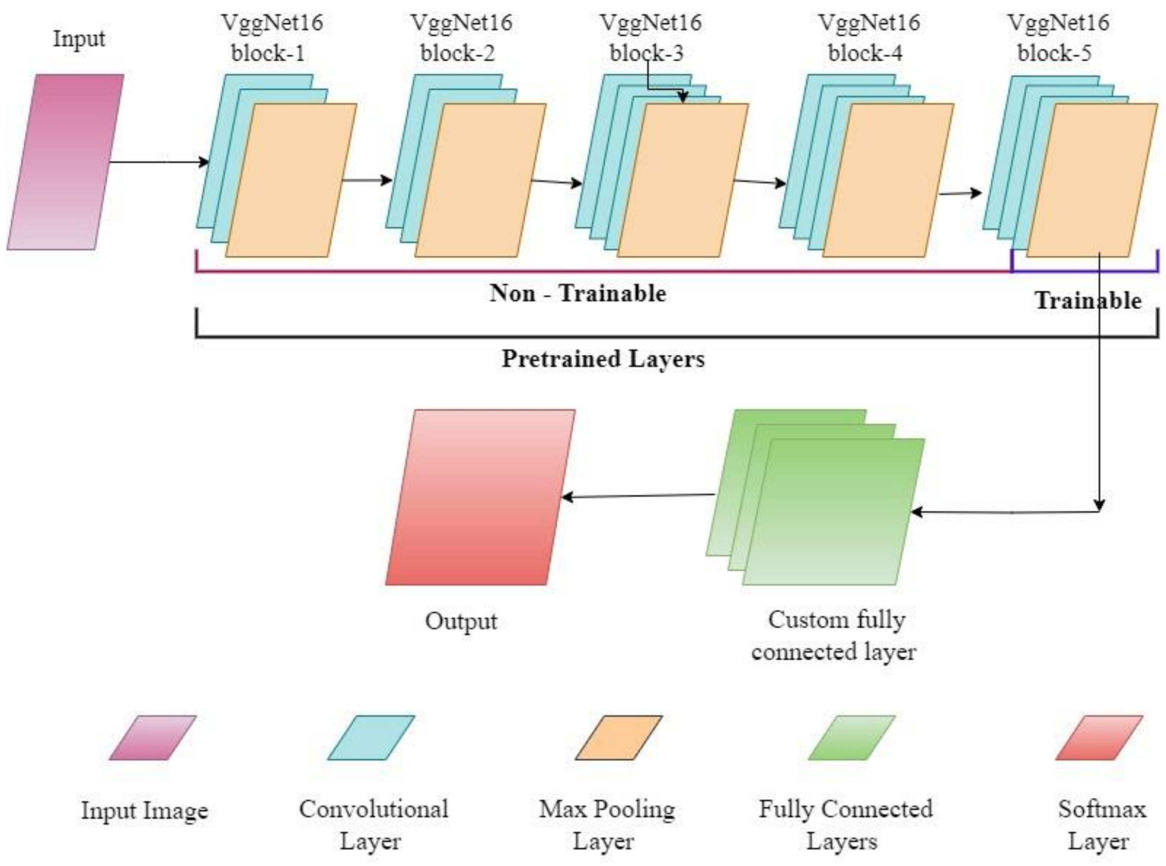


Fig. 3.2.2.2 Architecture of the VggNat16 Transfer Learning Model

3.2.2.3 Architecture of the InceptionNet V3 Transfer Learning model

Using pre-trained weights from the ImageNet dataset, the deep learning model for plant disease identification that has been constructed has the InceptionV3 architecture as its convolutional basis. By defining the number of layers to be trainable, the optional fine-tuning mechanism allows customization of the training procedure, striking a balance between adapting to the subtleties of plant disease datasets and making use of pre-trained knowledge. A global average pooling layer effectively reduces spatial dimensions after the convolutional base, and a densely linked layer with 256 units with ReLU activation captures complex patterns. A dropout layer with a 50% deactivation rate is added to reduce overfitting. The final output layer classifies the model's outputs into probability distributions across different plant disease classes by using softmax activation. With the Adam optimizer, categorical crossentropy loss, and accuracy as the evaluation metric, the model is assembled for training.

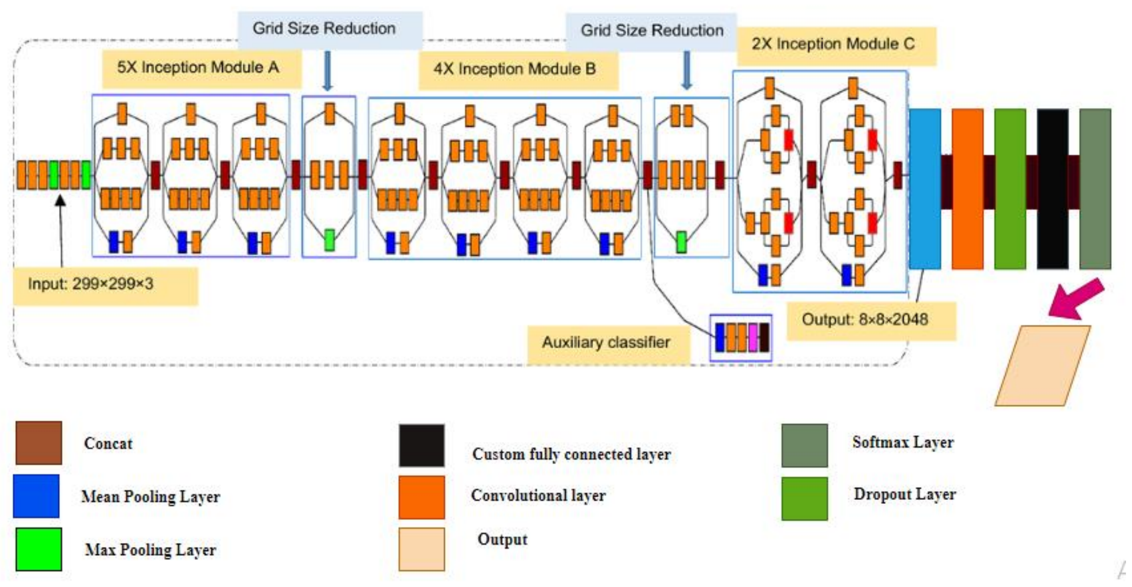


Fig. 3.2.2.3 Architecture of the InceptionNet V3 Transfer Learning Model

3.2.3 Advantages of Proposed System

With many benefits over current methods, the proposed plant disease detection system introduces a paradigm change in image-based agricultural diagnostics. The system offers enhanced generalization by leveraging transfer learning from popular architectures including as VGG16, ResNet50, and InceptionV3. This allows the system to adapt to a wide range of plant species and disease manifestations. The optimized bespoke classification head improves robustness and accuracy while resolving constraints brought on by a range of environmental factors and complex symptomologies. A key advantage is the addition of a larger and more varied dataset, which guarantees a thorough comprehension of illness trends. Through optimized model setups, real-time detection capabilities—a crucial component for prompt interventions—are smoothly incorporated. The system's scalability allows for the expansion of illness classifications and a rising dataset, and its emphasis on transparency guarantees the reliability and interpretability of decision-making procedures. The interpretability of the model architecture is further improved by visual representations, which give a clear picture of the inner workings of the system. To sum up, the approach that has been suggested is very versatile, scalable, and interpretable. This is a noteworthy development in the field of plant disease identification and holds great potential for precision agriculture.

3.3 Description of the used algorithm

3.3.1 ResNet50 Transfer Learning Model

The algorithm initializes and trains a deep learning model for plant disease identification in a methodical manner. Using the Keras package, a Sequential model is first initialized. Then, the original fully connected layers are removed to use the ResNet50 architecture as the convolutional base, which is then utilized as a feature extractor. A Global Average Pooling (GAP) layer is added after the ResNet50 base to provide a spatial summary of the learned characteristics. For the purpose of final classification into a predetermined number of classes, a Dense layer with a Softmax activation function is added. ResNet50 base weights are frozen, so pre-trained information is retained for further training. Using categorical crossentropy as the loss function, the Stochastic Gradient Descent (SGD) optimizer, and pertinent evaluation metrics, the model is assembled. For training and validation data, generators are used, together with callbacks for early halting and model weight saving. To put it simply, the approach uses transfer learning to refine the model for plant disease identification while maintaining important pre-trained characteristics. It does this by using ResNet50, which has already been trained on ImageNet. Plant disease picture classification becomes more effective when the best-performing weights are stored for later use, a result of the rigorous training procedure that produces a well-monitored model.

3.3.2 VggNet16 Transfer Learning Model

The algorithm used is a VGG16 convolutional neural network (CNN)-based transfer learning method for classifying plant diseases. One of the most important components in building a convolutional neural network (CNN) specifically designed for plant disease classification is the model generation function, often known as create_model. This flexible feature enables modification of important factors like the amount of fine-tuning done during training, the number of classes to be classified, the type of optimization technique, and the dimensions of the input. With the VGG16 architecture serving as the foundation, fully connected layers are excluded in favor of the inclusion of pretrained weights from the ImageNet dataset when a custom classification head is later added. Setting a lower learning rate and generating a fresh model instance with the final two layers unfrozen for additional training are the steps involved in being ready for fine-tuning. The subsequent fine-tuning stage is carried out with the help of designated data generators, early halting methods, and model checkpointing. The VGG16 base is augmented with a bespoke classification head that includes layers for flattening, dense units activated by rectified linear units (ReLUs), dropout for regularization, and a final dense output layer activated by softmax for multi-class probabilities. Accuracy measures, an optimizer (with 'rmsprop' as the default), and categorical cross entropy loss are used in the compilation of the model. The model's performance is carefully assessed after training, and the best weights are stored for possible later use. With this architecture, the pre-trained VGG16 model's strengths are combined with a customized classification head, making it an excellent choice for picture classification applications, especially those involving the identification of plant diseases. The ability to fine-tune is optional and improves adaptability to a variety of datasets, demonstrating the efficacy and versatility of the model building function.

3.3.3 InceptionNet V3 Transfer Learning Model

The plant disease identification technique that combines transfer learning with deep learningwith the InceptionV3 architecture. The method starts with careful data preparation, dividing the dataset into different classes, each of which stands for a different plant disease. Using an ImageDataGenerator on the training set, data augmentation methods including flips, brightness adjustments, and rotations are done to improve the robustness of the model. The InceptionV3 architecture, which has been pre-trained on the ImageNet dataset, is used for feature extraction. The final 20 layers of the architecture are fine-tuned to better fit the features of datasets related to plant diseases. The architecture of the model consists of a Dropout layer to reduce overfitting, a Dense layer to transform features, and a Global Average Pooling layer to reduce spatial dimension. Softmax activation is used in the final output layer to multi-class classify plant disease into 20 classes. Following compilation using the categorical crossentropy loss and Adam optimizer, the model is trained on the augmented set with validation at every epoch. Model checkpoints are saved in accordance with validation loss, and early termination is used in the event that no progress is seen. Metrics including accuracy, precision, recall, and F1 score are included in the model evaluation on the validation set and are displayed using the PlotLossesCallback. Lastly, the trained model is stored for usage and deployment at a later time. This method demonstrates its robustness and adaptability in agricultural settings by skillfully combining data augmentation, validation procedures, and transfer learning to produce an efficient and dependable solution for plant disease diagnosis.

3.4 Module Description

3.4.1 ResNet50 Transfer Learning model

Utilizing the sturdy ResNet50 architecture as a feature extractor, the constructed CNN model for plant disease detection first integrates the previously trained ResNet50 base from ImageNet.

After that, spatial features are condensed by a Global Average Pooling (GAP) layer, and the final classification into NUM_CLASSES is facilitated by a Dense layer with Softmax activation. For the best feature extraction, the model maintains essential pre-trained knowledge by designating the ResNet50 base layer as non-trainable. Multi-class classification is improved by stochastic gradient descent (SGD) optimization with adjusted parameters and categorical crossentropy as the loss function. The training process uses traingen and validgen as generators, and the callbacks cb_checkpointer and cb_early_stopper ensure that the model does not overfit by saving weight. Transfer learning and fine-tuning are successfully used in this holistic architecture to produce a flexible and precise model for plant disease identification.

3.4.2 VggNet16 Transfer Learning model

Create_model, a flexible CNN construction function designed for plant disease detection, is contained in the Python module. It uses the VGG16 base for feature extraction and accepts input parameters including picture size, output classes, optimizer selection, and fine-tuning choices. This enables selective fine-tuning based on user input. Layers like Flatten, Dense with ReLU activation, Dropout for regularization, and a final Dense layer for class probabilities are included in the custom classification head, which is attached to the VGG16 base. Accuracy is the evaluation metric in model compilation, along with the designated optimizer and categorical crossentropy loss. The model.summary() function provides a brief overview of the architecture, including layer types, output shapes, and trainable parameters. Because of the module's modular nature, creating flexible CNN architectures for plant disease detection that meet the needs of various datasets and users is made simple.

3.4.3 InceptionNet V3Transfer Learning model

For plant disease identification, the model combines transfer learning with InceptionV3 architecture, using image data generators for preprocessing and real-time augmentation. It creates generators for smooth data loading and creates directories for training and testing datasets. The architecture is defined by the create_model function, which allows for trainable layers to be customized and fine-tuned. Configurable parameters optimize training, while the Adam optimizer and categorical crossentropy loss are used in model compilation. Callbacks improve early training stoppage, live loss visualization, and model checkpointing. Loss visualization for dynamic training and validation is made easier by the livelossplot package. For efficient plant disease identification, the model offers a comprehensive approach that includes data preparation, model construction, and training.

3.5 Dataset Description

3.5.1 Data Acquisition

The study uses the PlantVillage dataset, which consists of 19,458 photos that have been hand-picked to improve plant disease identification through computer vision and deep learning. Data augmentation methods and a Generative Adversarial Network (GAN) model provide an extra 9,973 augmented photos in order to improve dataset variety. 29,928 photos in all, divided into 20 different classifications that correspond to various plant health and disease categories, make up the full dataset. The dataset provides a rich depiction of plant conditions and spans a variety of crops, including blueberries, apples, cherries, corn, grapes, peppers, potatoes, and strawberries.

Class No.	Class Names	Number of Images
0	Blueberry___healthy	2002
1	Apple___Cedar_apple_rust	777
2	Apple___healthy	2146
3	Cherry_(including_sour)___healthy	1363
4	Cherry_(including_sour)___Powdery_mildew	1564
5	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	1015
6	Corn_(maize)___Common_rust_	1707
7	Corn_(maize)___healthy	1662
8	Corn_(maize)___Northern_Leaf_Blight	1485
9	Grape___Black_rot	1681
10	Grape___Esca_(Black_Measles)	1884
11	Grape___healthy	916
12	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1577
13	Pepper,_bell___Bacterial_spot	1487
14	Pepper,_bell___healthy	1976
15	Potato___Early_blight	1476
16	Potato___healthy	552
17	Potato___Late_blight	1496
18	Strawberry___healthy	956
19	Strawberry___Leaf_scorch	1609

Table 3.5.1 Dataset Details

3.5.2 Data Preparation

In this part, the dataset containing 29,928 images of leafs are splitted into 65 and 35 percent as training dataset and test dataset.

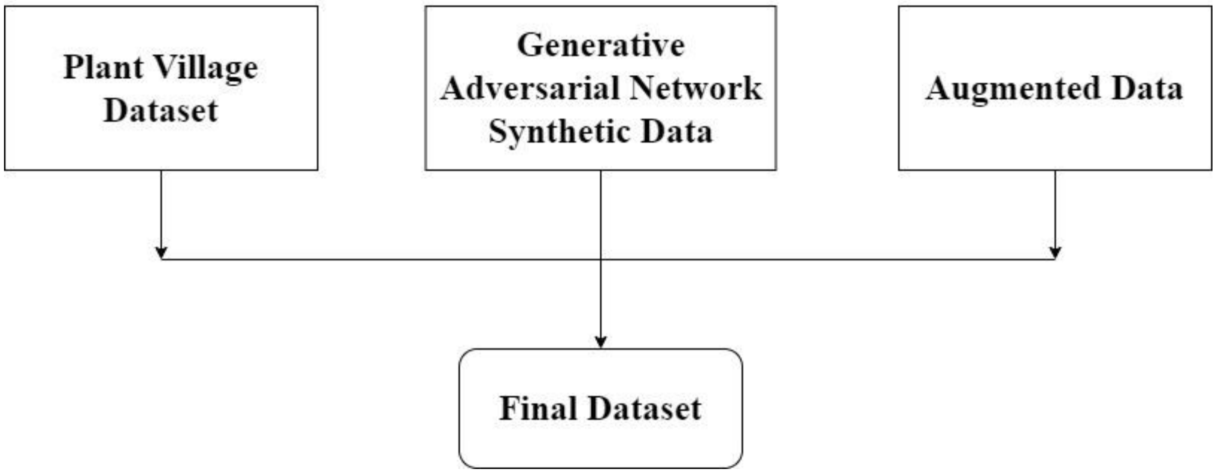


Fig. 3.5.2 Dataset source

3.5.2.1 Training and Validation Dataset

The training set comprises 19,458 plant leaf images, categorized into 20 classes, representing diverse plant species and health or disease conditions. This dataset is strategically divided, with 85% (16,545 images) allocated for training and 15% (2,911 images) for validation, ensuring robust model development and evaluation.

Class No.	Class Names	Number of Images
0	Blueberry___healthy	1502
1	Apple___Cedar_apple_rust	277
2	Apple___healthy	1646
3	Cherry_(including_sour)___healthy	863
4	Cherry_(including_sour)___Powdery_mildew	1064
5	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	515
6	Corn_(maize)___Common_rust_	1200
7	Corn_(maize)___healthy	1162
8	Corn_(maize)___Northern_Leaf_Blight	985
9	Grape___Black_rot	1181
10	Grape___Esca_(Black_Measles)	1383
11	Grape___healthy	423
12	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	1077
13	Pepper,_bell___Bacterial_spot	987
14	Pepper,_bell___healthy	1476
15	Potato___Early_blight	1000
16	Potato___healthy	152
17	Potato___Late_blight	1000
18	Strawberry___healthy	456
19	Strawberry___Leaf_scorch	1109

Table 3.5.2.1 Training dataset details

3.5.2.2 Testing Dataset

The testing set of the dataset comprises a subset of 9,973 images distributed across 20 different classes, with each class representing a specific category of plant health or disease. The distribution of images per class is as follows:

Class No.	**Class Name**	**Number of Images**
0	Corn_(maize)___Common_rust_	500
1	Grape___Black_rot	500
2	Corn_(maize)___healthy	500
3	Grape___Esca_(Black_Measles),	500
4	Pepper,_bell___healthy	500
5	Apple___healthy	500
6	Apple___Cedar_apple_rust	507
7	Cherry_(including_sour)___healthy	500
8	Cherry_(including_sour)___Powdery_mildew	500
9	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	500
10	Corn_(maize)___Northern_Leaf_Blight	501
11	Grape___healthy	493
12	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	500
13	Pepper,_bell___Bacterial_spot	500
14	Potato___Early_blight	500
15	Potato___healthy	476
16	Potato___Late_blight	500
17	Strawberry___healthy	496
18	Strawberry___Leaf_scorch	500
19	Blueberry___healthy	500

Table 3.5.2.2 Test dataset details

3.5.3 Image preprocessing and Data Augmentation

A batch size of 64 is specified for processing a batch of images in each iteration during model training. Two image data generators are defined, one for training and validation, and the other for testing. These generators apply various transformations to the images, enhancing the model's ability to generalize and perform well on diverse datasets.

3.5.3.1 Training and Validation Data Generator

The training and validation data generator, train_generator, is configured with the following transformations:

3.5.3.1.1 Rotation range of 90 degrees.

3.5.3.1.2 Brightness range varied between 0.1 and 0.7.

3.5.3.1.3 Horizontal and vertical shifts with a range of 0.5.

3.5.3.1.4 Horizontal and vertical flips.

3.5.3.1.5 Validation split of 15% to allocate a portion of the training data for validation.

3.5.3.1.6 VGG16 preprocessing function is applied.

The original and enhanced datasets are pointed to by the training and validation data folders, which are designated as train_data_dir and test_data_dir, respectively. Class_subset, the list of class names, is retrieved from the original dataset directory. The next step is to create batches of training and validation data using the flow_from_directory technique. The photos are enlarged to (299, 299) for both sets, and the categorical class mode—which indicates that the labels are one-hot encoded—is applied. 'training' for the training generator and 'validation' for the validation generator are the subset parameters that are supplied. For reproducibility, a seed is set and the data is shuffled.

3.5.3.2 Testing Data Generator:

Test_generator, the testing data generator, is set up to use the VGG16 preprocessing function. It pulls test data in batches from the upgraded dataset directory. The class mode is set to None and the photos are resized to (299, 299) because this generator is not used for validation or training. Since the batch size is set to 1, every image is processed independently. For consistent evaluation, the data is not scrambled throughout testing, and reproducibility is ensured by using a seed. The model's capacity to recognize a variety of patterns and generalize effectively to new data is improved by the application of data augmentation techniques.

Chapter 4

ANALYSIS AND DESIGN

4.1 Brief Introduction

4.1.1 Problem Statement

Plant diseases can be difficult to identify in a timely and accurate manner, which has an influence on crop quality and output. Conventional approaches can take a long time and may not react quickly to diseases that are changing. The suggested remedy entails using transfer learning to create and apply the deep learning models VGG16, ResNet50, and InceptionV3. By automating the identification process through the analysis of leaf pictures, these models seek to revolutionize the detection of plant diseases and support precision agriculture and efficient disease management.

4.1.2 Design Overview

The foundation of our strategy for plant disease identification is the application of transfer learning to the VGG16, ResNet50, and InceptionV3 architectures. We take advantage of these models' shown ability to capture complex visual cues associated with various plant diseases by initializing them with pre-trained weights from ImageNet. By enabling the customisation of trainable layers, the fine-tuning capability achieves a balance between utilizing pre-trained information and adjusting to the unique peculiarities of plant disease datasets. Common elements like dropout, dense layers, and global average pooling are incorporated to further improve feature extraction and transformation while reducing overfitting during training. The output layer effectively classifies plant disease classes by converting model outputs into probability distributions using softmax activation. The all-encompassing model compilation, which makes use of the Adam optimizer, the category crossentropy loss function, and accuracy measures, emphasizes a comprehensive approach to effective training and assessment. Our plant disease detection models are positioned as strong tools for precision agriculture by this methodology, which is based on cutting-edge methods and technical jargon. It offers the possibility of precise and dependable identification of plant health issues that are essential for sustainable crop management.

4.1.3 Analysis

The suggested models use deep learning and transfer learning to tackle the problem of plant disease detection in agriculture. A variety of architectures are offered by the selection of VGG16, ResNet50, and InceptionV3, each of which is renowned for having particular advantages in feature representation. Fine-tuning enables flexibility to particular datasets, and dropout and global average pooling are common elements that improve the resilience and efficiency of the model. The model compilation shows a thorough approach to efficient training by employing the Adam optimizer and category cross entropy loss. The research emphasizes the advanced strategies used to address the complexities of plant disease detection in agriculture, while also acknowledging the importance of technical words. Overall, these models represent a cutting-edge approach to precision agriculture, offering the potential for accurate and rapid identification of plant diseases crucial for sustainable crop management.

4.2 Requirement Analysis

4.2.1 Software Requirement

To generate the VGG16, ResNet50, and InceptionV3 models for plant disease detection, a specific set of software requirements is essential. The following list outlines the necessary software components and tools:

Deep Learning Frameworks	TensorFlow : A popular open-source deep learning framework
Machine Learning Libraries	Keras: An high-level neural networks API
Python	Python is the primary programming language used for deep learning model development.
Image Processing Libraries	OpenCV: A powerful computer vision library
Data Manipulation and Analysis	NumPy: A fundamental package for scientific computing with Python, providing support for large, multi-dimensional arrays and matrices.
Additional Libraries	Matplotlib: A plotting library for creating visualizations, which can be helpful for analyzing model performance and results. Scikit-learn: A machine learning library that provides simple and efficient tools for data analysis and modeling.
Development Environment	Jupyter Notebooks or an Integrated Development Environment (IDE) like Visual Studio Code or PyCharm for interactive development, experimentation, and model training.

Table 4.2.1 Software Requirement

4.2.2 Hardware Requirement

The hardware requirements for generating and training deep learning models, including VGG16, ResNet50, and InceptionV3 for plant disease detection, can vary based on the size of the dataset, the complexity of the models, and the desired training speed.

Processor (CPU)	A multi-core processor with a clock speed of at least 2.5 GHz is recommended.
Graphics Processing Unit (GPU)(optional)	For faster model training, especially when working with large datasets, a dedicated GPU is highly recommended.
Memory (RAM)	A minimum of 2 GB of RAM.
Storage	SSD storage is preferred over HDD for faster data access, especially when loading and preprocessing large image datasets. Ensure sufficient storage space for datasets, model checkpoints, and intermediate files.
CUDA and cuDNN (Optional)	If using NVIDIA GPUs, installing CUDA and cuDNN can significantly accelerate deep learning computations. These libraries are essential for GPU acceleration and are compatible with frameworks like TensorFlow.
Networking	A stable internet connection is important for downloading datasets, pre-trained models, and necessary libraries.
Power Supply	Ensure that the power supply is sufficient for the CPU, GPU, and other components.

Table 4.2.2 Hardware Requirement

4.3 Detailed Design

4.3.1 System Design

The system design for plant disease detection incorporates a structured pipeline encompassing key stages, from loading the dataset to training and prediction using deep learning models with transfer learning.

4.3.1.1 Load Data

Getting a varied dataset with photos of different plant species afflicted with illnesses is the first stage. An essential part of training and testing the deep learning models is this dataset. This dataset will be loaded quickly by the system, guaranteeing that it includes a representative sample of plant diseases.

4.3.1.2 Image Preprocessing

After the dataset is loaded, there is a thorough image preparation step. To improve the robustness of the model, this involves actions including scaling photos to a standard dimension, normalizing pixel values to a range, and using augmentation techniques. In order to prepare the dataset in a format that is useful for efficient model training, image preprocessing is essential.

4.3.1.3 Deep Learning Pretrained Model

The model design is based on the selected deep learning architectures, namely VGG16, ResNet50, and InceptionV3. Because these designs have been pre-trained on massive datasets such as ImageNet, they can recognize complex aspects in visual patterns. Plant disease identification is made possible by the system's configuration, which uses these trained models as feature extractors and makes use of their acquired representations.

4.3.1.4 Transfer Learning Implementation

Integrating transfer learning is a crucial approach in the system architecture. The system leverages the knowledge that pre-trained models have amassed from a variety of datasets by employing them. With the transfer learning approach, specific layers of the pre-trained models are fine-tuned to adapt to the subtleties of the plant disease dataset, while other levels are frozen to keep their learned characteristics. The system can benefit from both dataset-specific adaptations and generalizable features thanks to this approach.

4.3.1.5 Training and Prediction

The preprocessed dataset is fed into the deep learning models during the model training phase. Training settings including batch size, learning rate, and epoch count can be managed by the system. The model gains the ability to discriminate between several plant disease classes during training. The trained model is then used to make predictions on fresh, unobserved data. The trained model can be used by users to identify diseases in plant photos, which will help with quick and accurate identification.

This holistic system design encapsulates the complete workflow, ensuring a seamless and effective process from loading the dataset to training and deploying deep learning models for plant disease detection. It aligns the strengths of transfer learning with the specific requirements of agricultural applications, providing a powerful tool for crop health assessment and disease management.

4.3.2 Dataflow Diagram

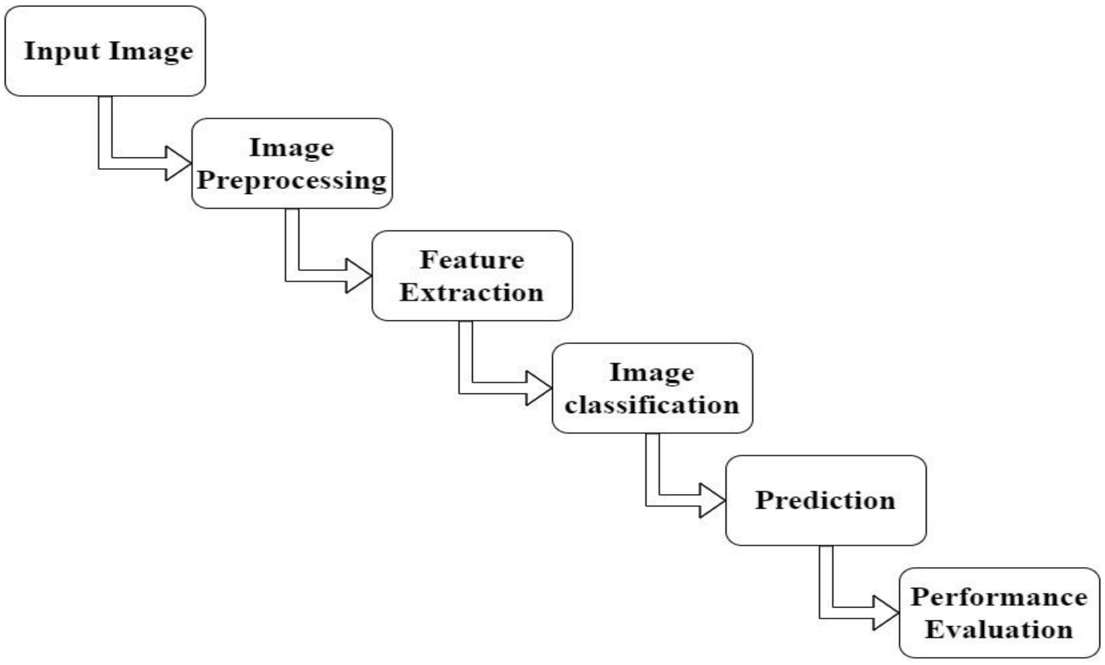


Fig. 4.3.2 Dataflow Diagram

4.3.3 Class Diagram

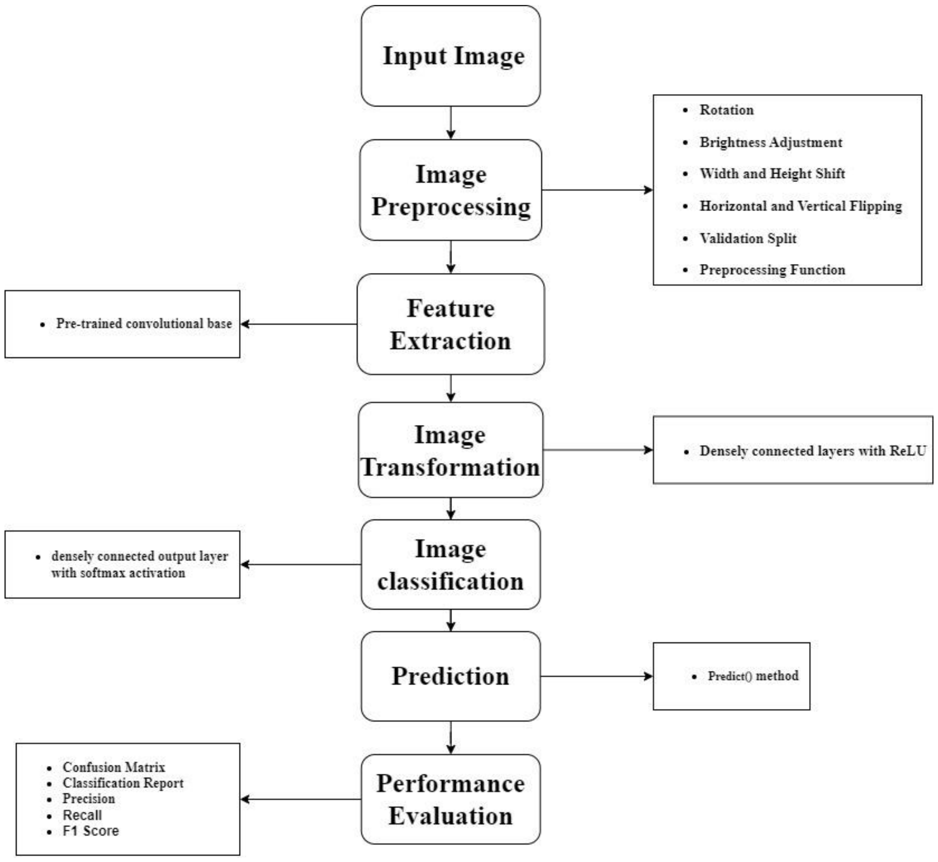


Fig. 4.3.3 Class Diagram

4.3.4 Use Case Diagram

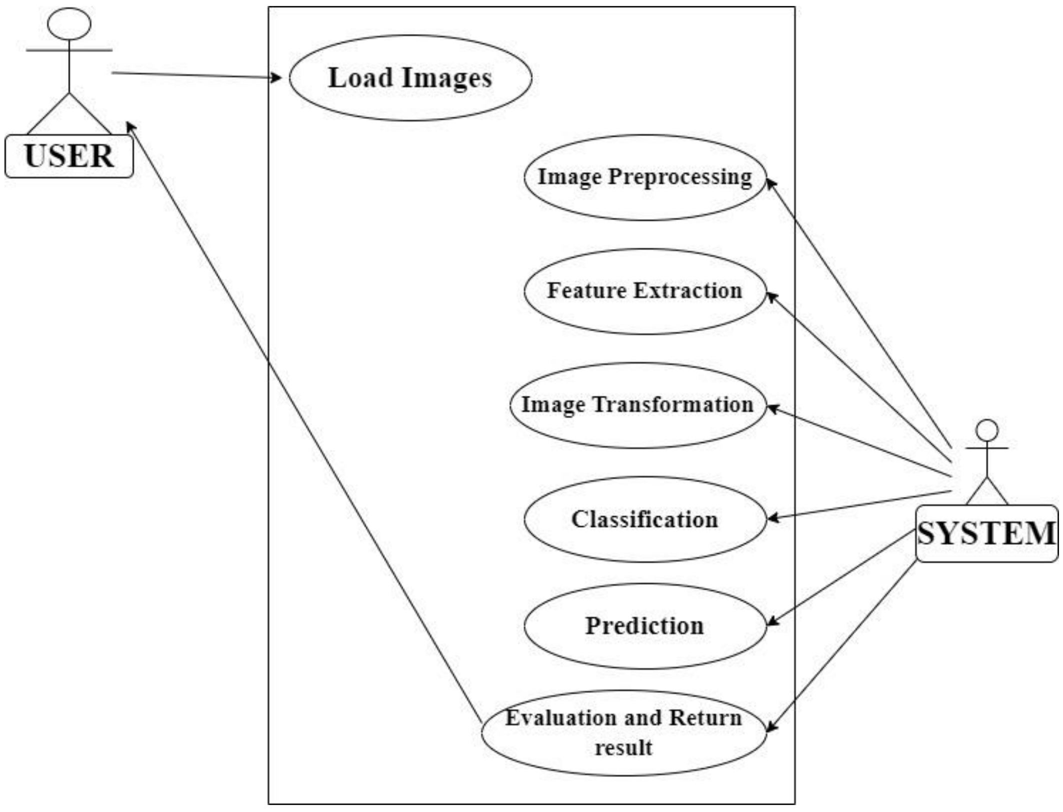


Fig. 4.3.4 Use case Diagram

4.3.5 Sequence Diagram

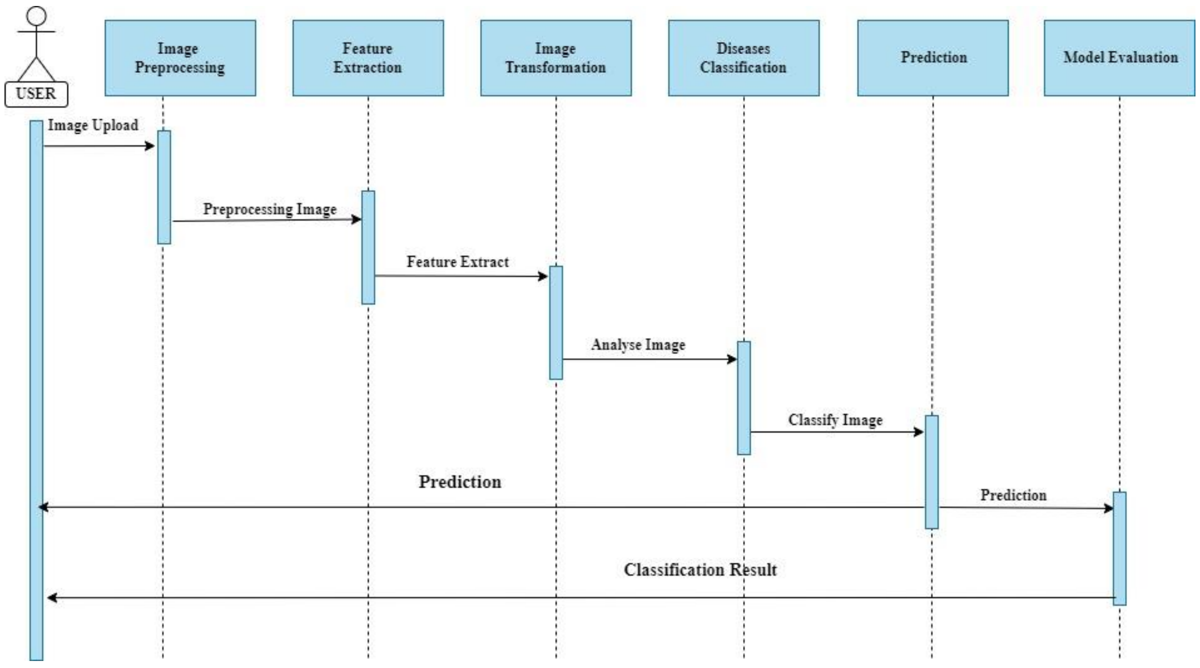


Fig. 4.3.5 Sequence Diagram

Chapter 5

IMPLEMENTATION & RESULT ANALYSIS

5.1 Tools Used

The provided code uses various tools and libraries for building, training, and evaluating deep learning models for plant disease detection. Here's a list of the tools used:
Google Colab, Keras, scikit-image (skimage), NumPy, PIL (Pillow), TensorFlow, Matplotlib, scikit-learn, ImageDataGenerator, ResNet50, VggNet16, InceptionV3, ModelCheckpoint, EarlyStopping, SGD (Stochastic Gradient Descent), Confusion Matrix, Classification Report, ImageDataGenerator, matplotlib.pyplot, Seaborn, Image Preprocessing and Augmentation, Various Callbacks, ModelCheckpoint and EarlyStopping
These tools collectively provide a comprehensive environment for implementing, training, and evaluating deep learning models for plant disease detection.

5.2 Methodology

The methodology for implementing and training deep learning models for plant disease detection involves several key steps.
First, the issue is clearly stated, highlighting how important it is for farmers to identify diseases early on. Next, an appropriate dataset is obtained that includes a wide variety of photos depicting different plant illnesses. The data is carefully preprocessed, with pixel values normalized and resized for homogeneity. The process of selecting a model entails selecting well-known deep learning architectures that have demonstrated efficacy in image classification tasks, such as VGG16, ResNet50, or InceptionV3. By altering the dense layers of pre-trained models, transfer learning is utilized to customize them for the categorization of plant diseases. Defined optimization algorithms, loss functions, and assessment metrics are included in the model compilation. The provided dataset is used for training, which includes early pausing and monitoring measures to avoid overfitting. On a different test dataset, the trained model is rigorously assessed using confusion matrices and performance measures to ensure in-depth examination. The effectiveness of the model is evaluated by carefully examining the results, with an emphasis on finding misclassifications and possible areas for improvement. Fine-tuning, which includes modifying the hyperparameters or model architecture, is taken into consideration when needed. A complete report that details the whole approach, model

architecture, training process, and evaluation results is crucial, as are visualizations of training history. Using deep learning models for plant disease detection in a methodical manner guarantees a solid and well-informed deployment.

5.3 Graphs / Tables

5.3.1 ResNet50 with Transfer learning model With Fine tune

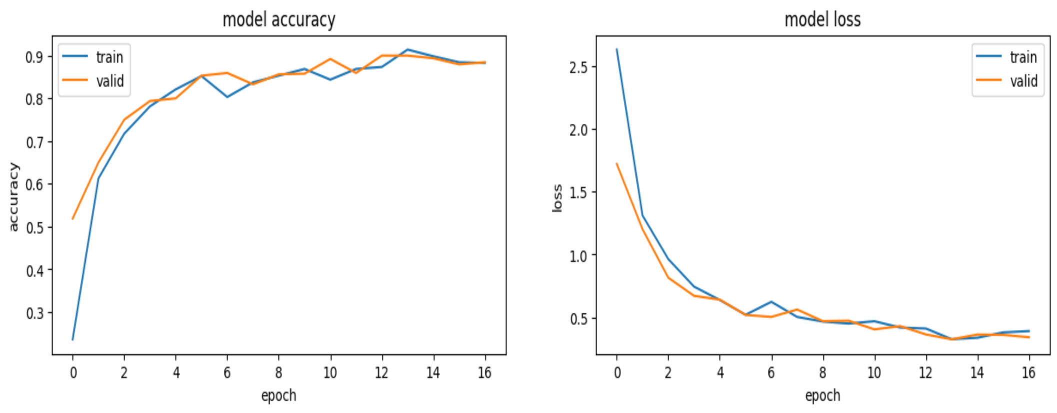


Fig. 5.3.1 ResNat50 Training and Validation Accuracy and Loss over Epochs

Parameter name	Accuracy	Loss
Training	88.28	0.3898
Validation	88.44	0.3409
Test	83.23	--------

Table 5.3.1 Accuracy and loss result of ResNet50 model

Classification Report:

Class	Precision	Recall	F1-score	Support
0	0.82	0.99	0.90	500
1	0.95	0.88	0.91	597
2	0.95	0.82	0.89	500
3	0.89	0.94	0.92	500
4	0.87	0.96	0.92	500
5	0.73	0.70	0.71	500
6	0.48	0.99	0.64	500
7	1.00	0.84	0.91	500
8	0.76	0.37	0.50	1000

9	0.84	0.94	0.89	500
10	0.99	0.78	0.87	500
11	0.98	0.83	0.90	493
12	0.98	0.92	0.95	500
13	0.85	0.82	0.84	500
14	0.55	0.98	0.71	500
15	0.91	0.94	0.93	500
16	0.97	0.56	0.71	476
17	0.94	0.90	0.92	500
18	0.98	0.96	0.97	496
19	0.94	0.97	0.95	500

Table 5.3.1 Classification report

Confusion Matrix

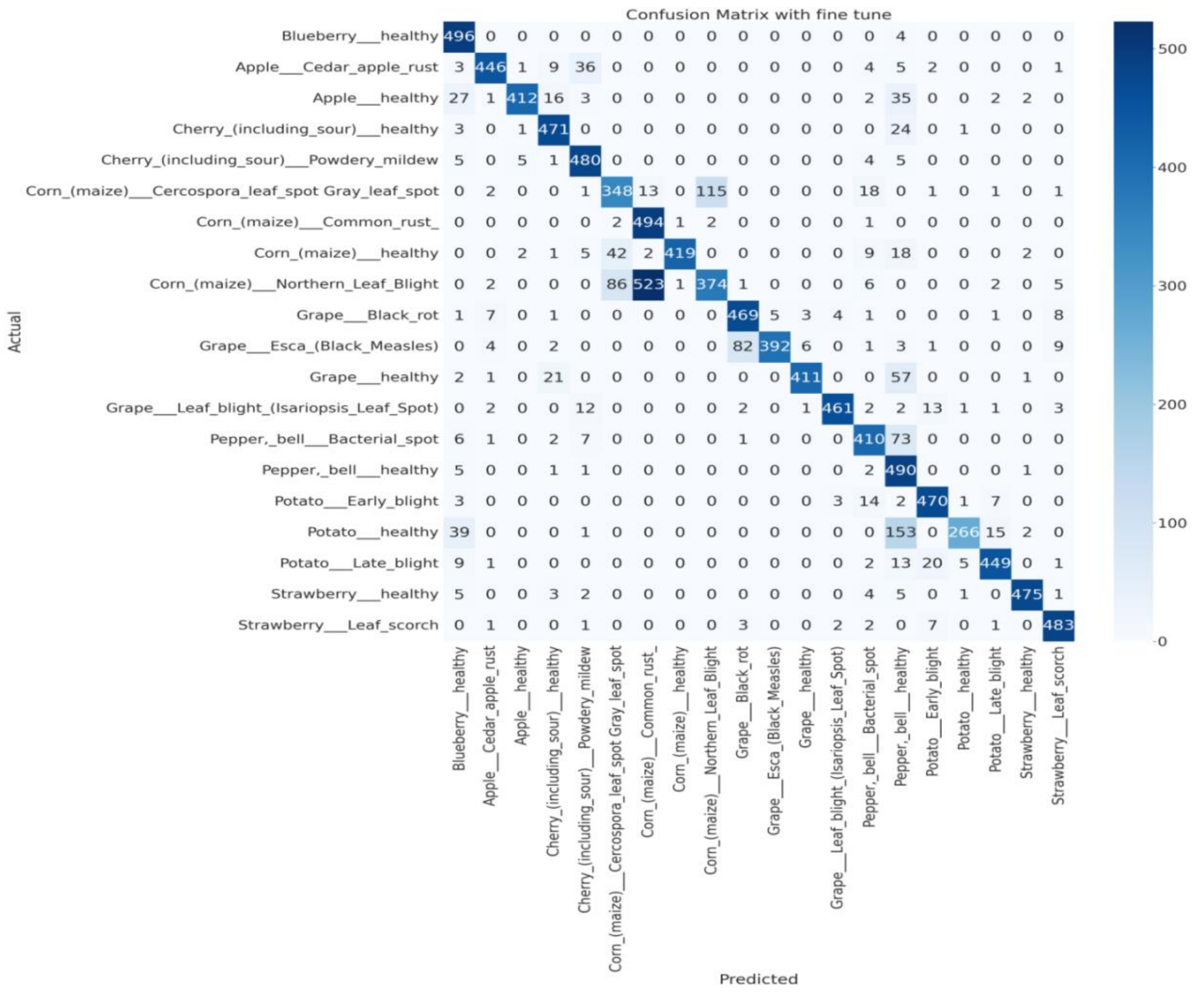


Fig. 5.3.1 Confusion matrix

5.3.2 VggNet16 with Transfer Learning model without fine tune

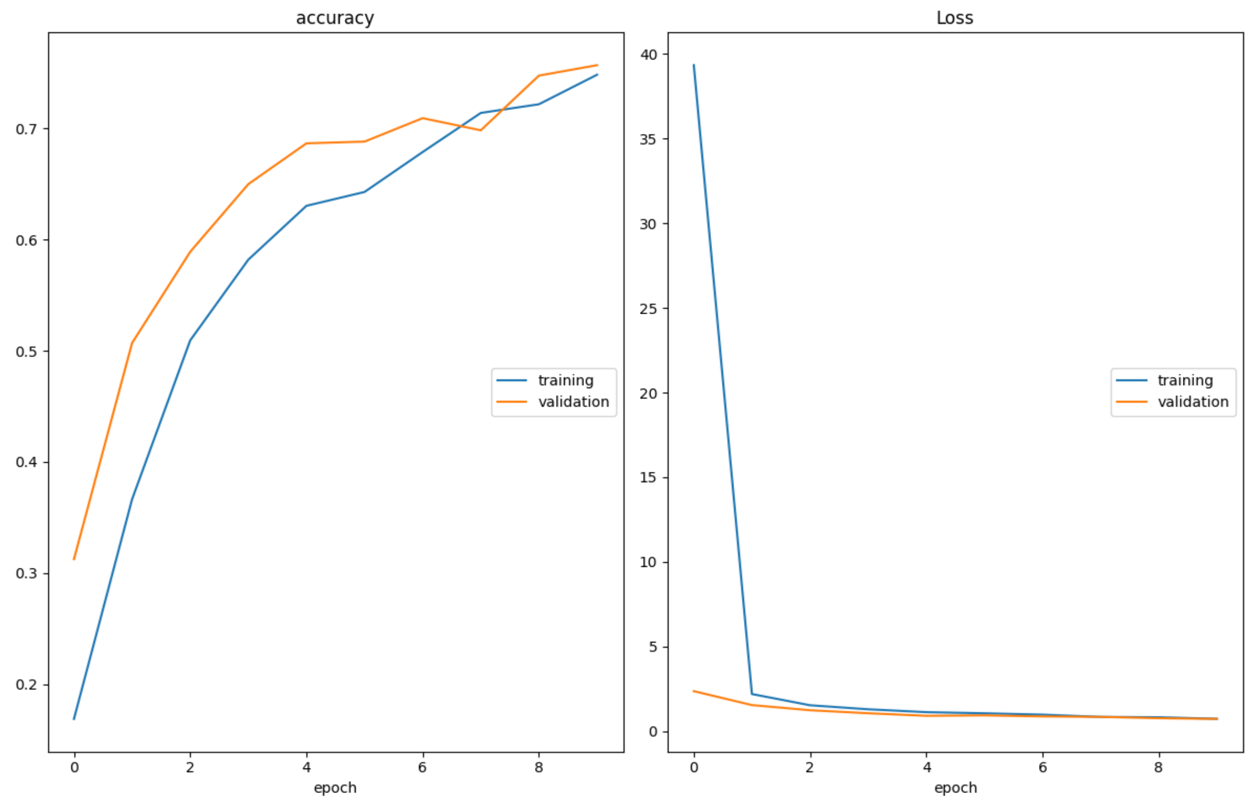


Fig. 5.3.2 Accuracy and loss plot diagram

Parameter name	Accuracy	Loss
Training	74.84	73.89
Validation	75.70	73.19
Test	72.04	----------

Fig. 5.3.2 Accuracy and loss result

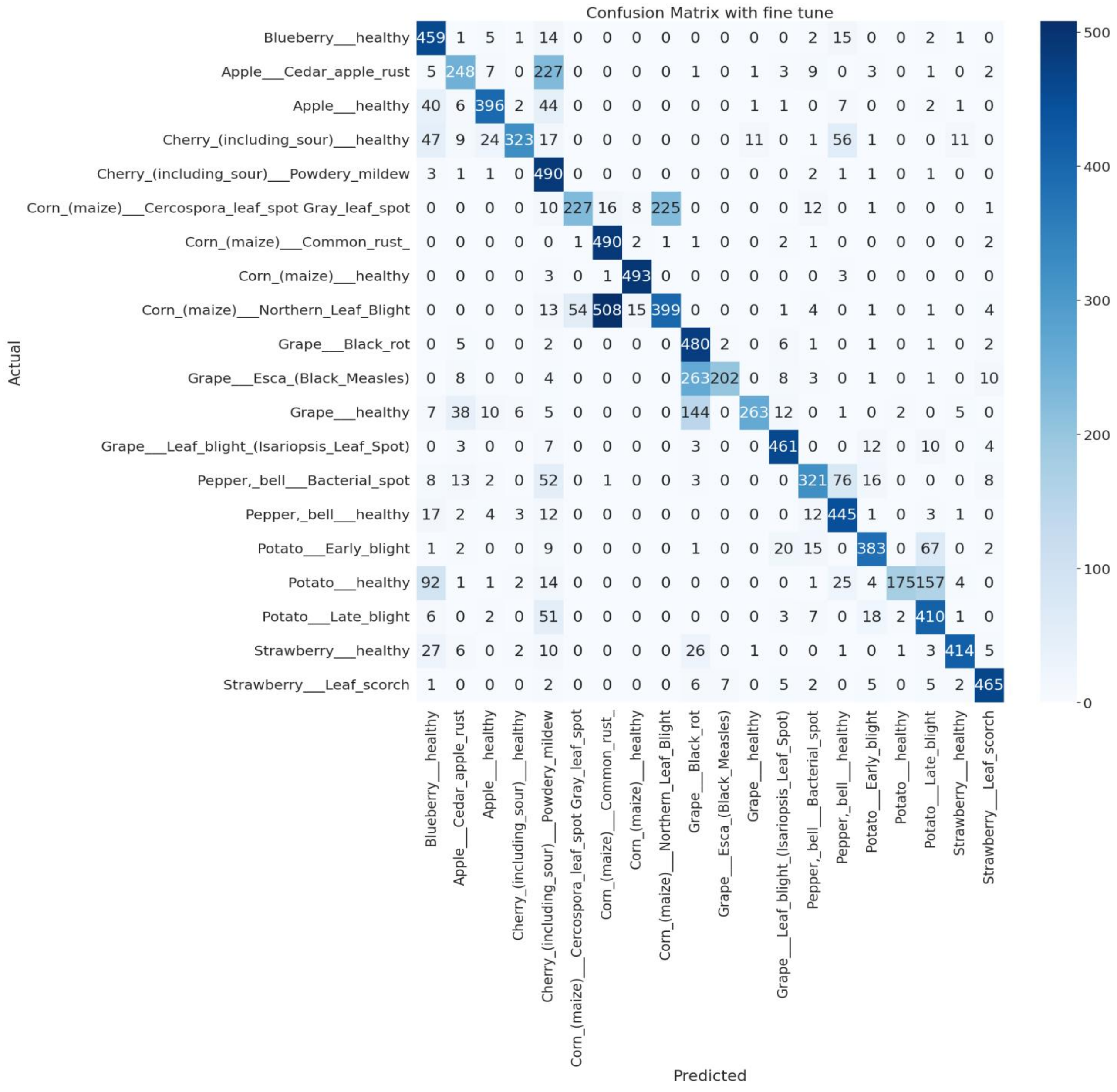
Classification Report

Class No.	Precision	Recall	f1-score	Support
0	0.64	0.92	0.76	500
1	0.72	0.49	.58	507
2	0.88	0.79	0.83	500
3	0.95	0.65	0.77	500
4	0.50	0.98	0.66	500
5	0.80	0.45	0.58	500
6	0.48	0.98	0.65	500
7	0.95	0.99	0.97	500
8	0.64	0.40	0.49	1000

9	0.52	0.96	0.67	500
10	0.96	0.40	0.57	500
11	0.95	0.53	0.68	493
12	0.88	0.92	0.90	500
13	0.82	0.64	0.72	500
14	0.71	0.89	0.79	500
15	0.85	0.77	0.81	500
16	0.97	0.37	0.53	476
17	0.62	0.82	0.70	500
18	0.94	0.83	0.88	496
19	0.92	0.93	0.93	500

Table 5.3.2 Classification Report

Confusion Matrix



Actual \ Predicted	Blueberry___healthy	Apple___Cedar_apple_rust	Apple___healthy	Cherry_(including_sour)___healthy	Cherry_(including_sour)___Powdery_mildew	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	Corn_(maize)___Common_rust_	Corn_(maize)___healthy	Corn_(maize)___Northern_Leaf_Blight	Grape___Black_rot	Grape___Esca_(Black_Measles)	Grape___healthy	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	Pepper,_bell___Bacterial_spot	Pepper,_bell___healthy	Potato___Early_blight	Potato___healthy	Potato___Late_blight	Strawberry___healthy	Strawberry___Leaf_scorch
Blueberry___healthy	459	1	5	1	14	0	0	0	0	0	0	0	0	2	15	0	0	2	1	0
Apple___Cedar_apple_rust	5	248	7	0	227	0	0	0	0	1	0	1	3	9	0	3	0	1	0	2
Apple___healthy	40	6	396	2	44	0	0	0	0	0	0	1	1	0	7	0	0	2	1	0
Cherry_(including_sour)___healthy	47	9	24	323	17	0	0	0	0	0	0	11	0	1	56	1	0	0	11	0
Cherry_(including_sour)___Powdery_mildew	3	1	1	0	490	0	0	0	0	0	0	0	0	2	1	1	0	1	0	0
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0	0	0	0	10	227	16	8	225	0	0	0	0	12	0	1	0	0	0	1
Corn_(maize)___Common_rust_	0	0	0	0	0	1	490	2	1	1	0	0	2	1	0	0	0	0	0	2
Corn_(maize)___healthy	0	0	0	0	3	0	1	493	0	0	0	0	0	0	3	0	0	0	0	0
Corn_(maize)___Northern_Leaf_Blight	0	0	0	0	13	54	508	15	399	0	0	0	1	4	0	1	0	1	0	4
Grape___Black_rot	0	5	0	0	2	0	0	0	0	480	2	0	6	1	0	1	0	1	0	2
Grape___Esca_(Black_Measles)	0	8	0	0	4	0	0	0	0	263	202	0	8	3	0	1	0	1	0	10
Grape___healthy	7	38	10	6	5	0	0	0	0	144	0	263	12	0	1	0	2	0	5	0
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0	3	0	0	7	0	0	0	0	3	0	0	461	0	0	12	0	10	0	4
Pepper,_bell___Bacterial_spot	8	13	2	0	52	0	1	0	0	3	0	0	0	321	76	16	0	0	0	8
Pepper,_bell___healthy	17	2	4	3	12	0	0	0	0	0	0	0	0	12	445	1	0	3	1	0
Potato___Early_blight	1	2	0	0	9	0	0	0	0	1	0	0	20	15	0	383	0	67	0	2
Potato___healthy	92	1	1	2	14	0	0	0	0	0	0	0	0	1	25	4	175	157	4	0
Potato___Late_blight	6	0	2	0	51	0	0	0	0	0	0	0	3	7	0	18	2	410	1	0
Strawberry___healthy	27	6	0	2	10	0	0	0	0	26	0	1	0	0	1	0	1	3	414	5
Strawberry___Leaf_scorch	1	0	0	0	2	0	0	0	0	6	7	0	5	2	0	5	0	5	2	465

Fig. 5.3.2 Confusion matrix

5.3.3 VggNet16 transfer learning model with fine tune

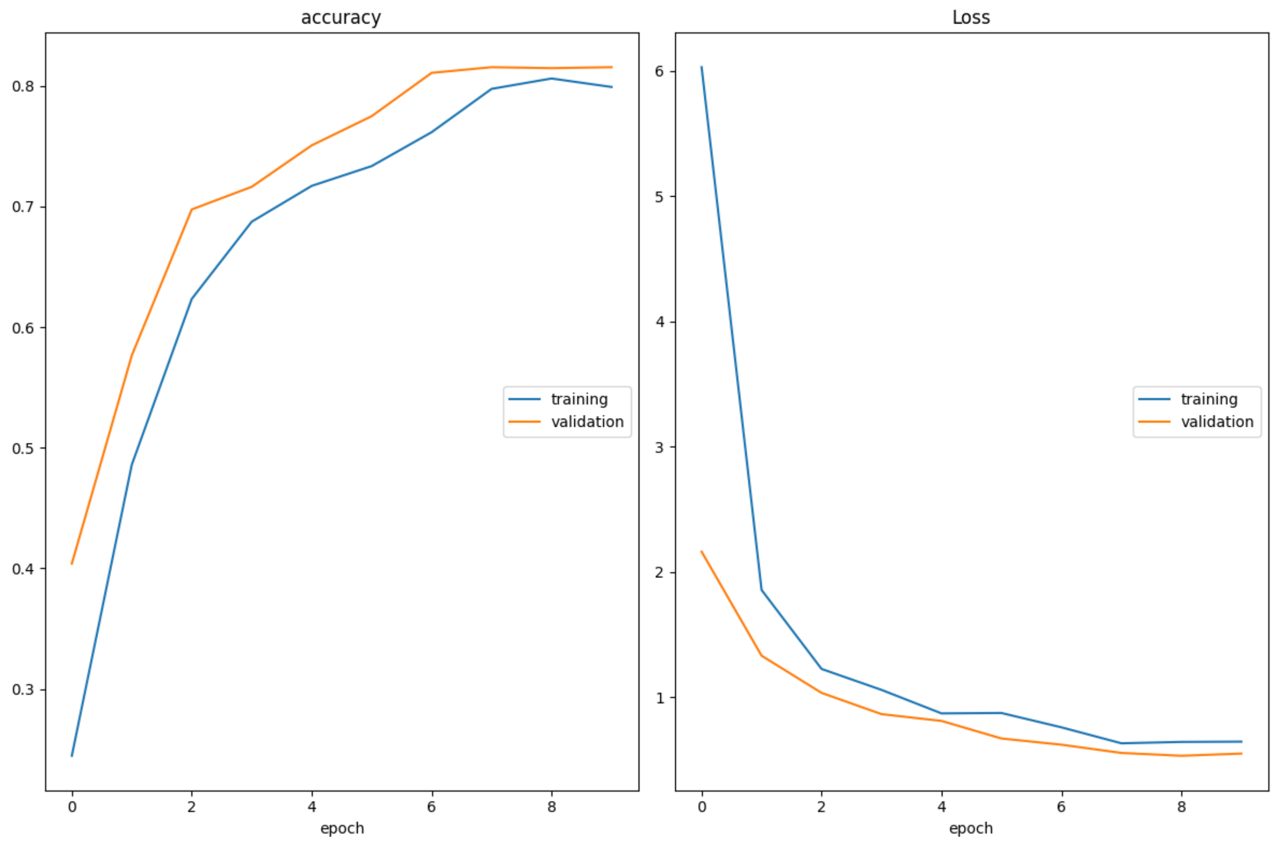
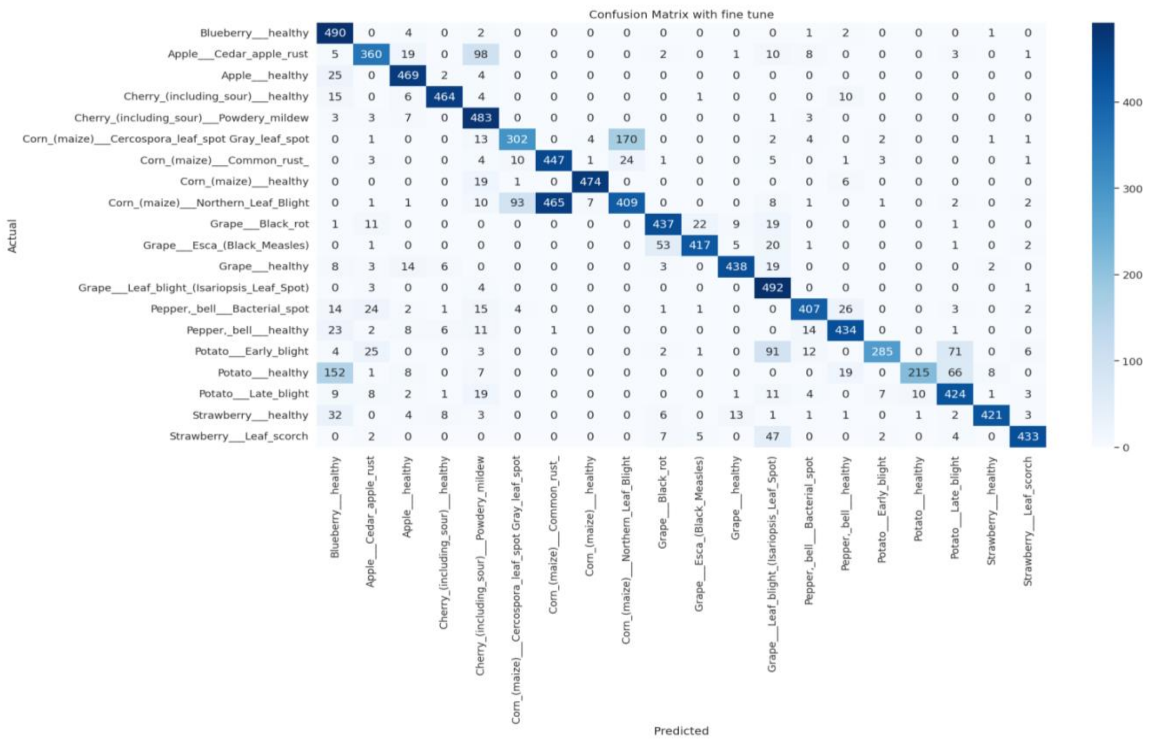


Fig. 5.3.3 Accuracy and loss plot diagram

Confusion Matrix



Actual \ Predicted	Blueberry___healthy	Apple___Cedar_apple_rust	Apple___healthy	Cherry_(including_sour)___healthy	Cherry_(including_sour)___Powdery_mildew	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	Corn_(maize)___Common_rust_	Corn_(maize)___healthy	Corn_(maize)___Northern_Leaf_Blight	Grape___Black_rot	Grape___Esca_(Black_Measles)	Grape___healthy	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	Pepper,_bell___Bacterial_spot	Pepper,_bell___healthy	Potato___Early_blight	Potato___healthy	Potato___Late_blight	Strawberry___healthy	Strawberry___Leaf_scorch
Blueberry___healthy	490	0	4	0	2	0	0	0	0	0	0	0	0	1	2	0	0	0	1	0
Apple___Cedar_apple_rust	5	360	19	0	98	0	0	0	0	2	0	1	10	8	0	0	0	3	0	1
Apple___healthy	25	0	469	2	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Cherry_(including_sour)___healthy	15	0	6	464	4	0	0	0	0	0	1	0	0	0	10	0	0	0	0	0
Cherry_(including_sour)___Powdery_mildew	3	3	7	0	483	0	0	0	0	0	0	0	1	3	0	0	0	0	0	0
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0	1	0	0	13	302	0	4	170	0	0	0	2	4	0	2	0	0	1	1
Corn_(maize)___Common_rust_	0	3	0	0	4	10	447	1	24	1	0	0	5	0	1	3	0	0	0	1
Corn_(maize)___healthy	0	0	0	0	19	1	0	474	0	0	0	0	0	0	6	0	0	0	0	0
Corn_(maize)___Northern_Leaf_Blight	0	1	1	0	10	93	465	7	409	0	0	0	8	1	0	1	0	2	0	2
Grape___Black_rot	1	11	0	0	0	0	0	0	0	437	22	9	19	0	0	0	0	1	0	0
Grape___Esca_(Black_Measles)	0	1	0	0	0	0	0	0	0	53	417	5	20	1	0	0	0	1	0	2
Grape___healthy	8	3	14	6	0	0	0	0	0	3	0	438	19	0	0	0	0	0	2	0
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0	3	0	0	4	0	0	0	0	0	0	0	492	0	0	0	0	0	0	1
Pepper,_bell___Bacterial_spot	14	24	2	1	15	4	0	0	0	1	1	0	0	407	26	0	0	3	0	2
Pepper,_bell___healthy	23	2	8	6	11	0	1	0	0	0	0	0	0	14	434	0	0	1	0	0
Potato___Early_blight	4	25	0	0	3	0	0	0	0	2	1	0	91	12	0	285	0	71	0	6
Potato___healthy	152	1	8	0	7	0	0	0	0	0	0	0	0	0	19	0	215	66	8	0
Potato___Late_blight	9	8	2	1	19	0	0	0	0	0	0	1	11	4	0	7	10	424	1	3
Strawberry___healthy	32	0	4	8	3	0	0	0	0	6	0	13	1	1	1	0	1	2	421	3
Strawberry___Leaf_scorch	0	2	0	0	0	0	0	0	0	7	5	0	47	0	0	2	0	4	0	433

Fig. 5.3.3 Confusion matrix

VGGNET_16 model with fine tune

Parameter name	Accuracy	Loss
Training	79.92	64.67
Validation	81.56	55.14
Test	79.27	**-**

Table 5.3.3 Accuracy and loss result

Classification Report:

Name	Precision	Recall	f1-score	Support
0	0.63	0.98	0.77	500
1	0.80	0.71	0.75	507
2	0.86	0.94	0.90	500
3	0.95	0.93	0.94	500
4	0.69	0.97	0.81	500
5	0.74	0.60	0.66	500
6	0.49	0.89	0.63	500
7	0.98	0.95	0.96	500
8	0.68	0.41	0.51	1000
9	0.85	0.87	0.86	500
10	0.93	0.83	0.88	500
11	0.94	0.89	0.91	493
12	0.68	0.98	0.80	500
13	0.89	0.81	0.85	500
14	0.87	0.87	0.87	500
15	0.95	0.57	0.71	500
16	0.95	0.45	0.61	476
17	0.73	0.85	0.79	500
18	0.97	0.85	0.91	496
19	0.95	0.87	0.91	500

Table 5.3.3 Accuracy and loss result

5.3.4 InceptionNet V3 with Transfer learning model with fine tune

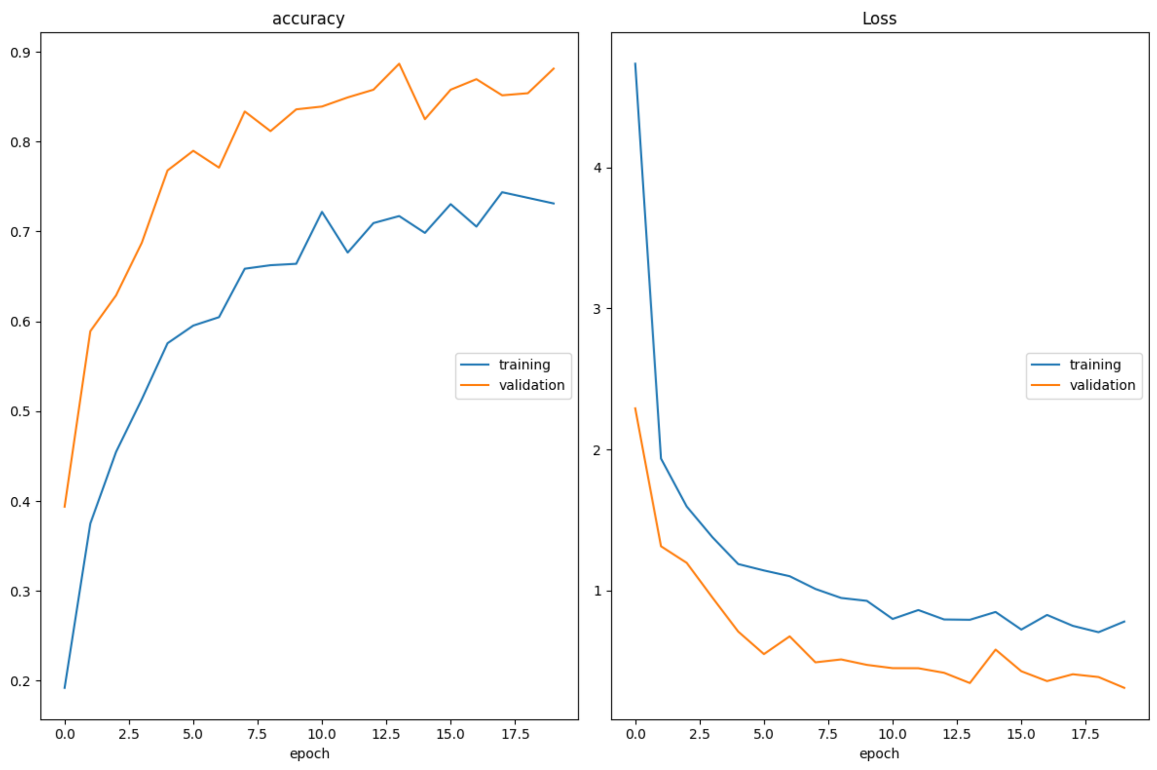


Fig. 5.3.4 Accuracy and loss plot

Confusion Matrix

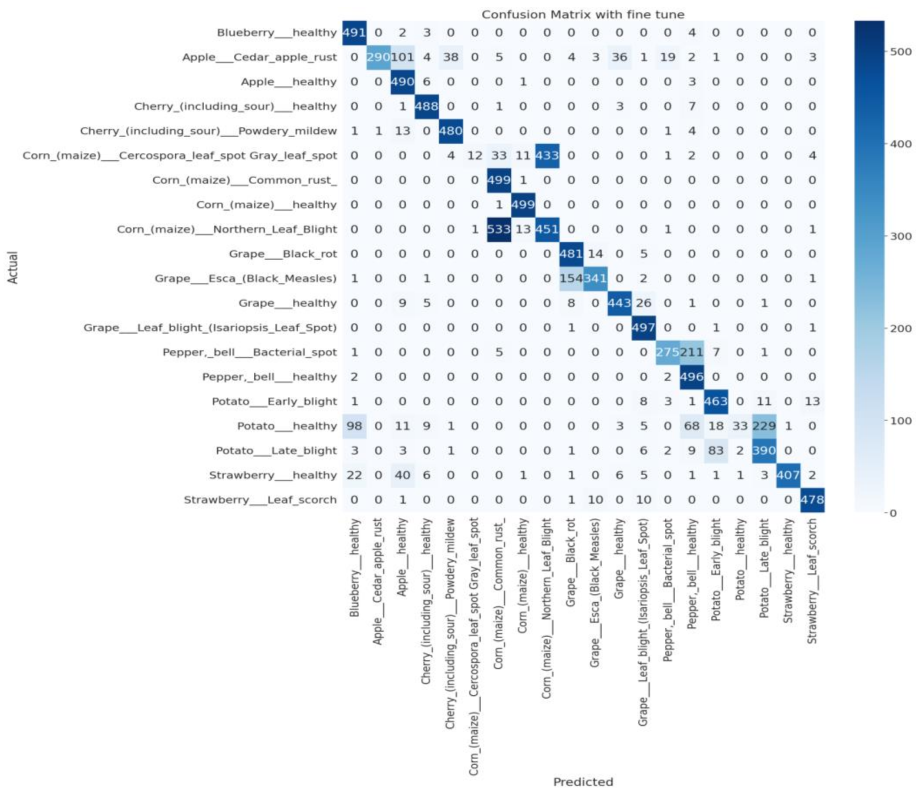


Fig. 5.3.4 confusion matrix

Inceptionnet model with fine tune

Parameter name	Accuracy	Loss
Training	73.12	0.7790
Validation	88.13	0.3089
Test	76.43	---------

Table 5.3.4 Accuracy and loss result

Classification Report:

Class	Precision	Recall	F1-score	Support
0	0.79	0.98	0.88	500
1	1.00	0.57	0.73	507
2	0.73	0.98	0.84	500
3	0.93	0.98	0.95	500
4	0.92	0.96	0.94	500
5	0.92	0.02	0.05	500
6	0.46	1.00	0.63	500
7	0.95	1.00	0.97	500
8	0.51	0.45	0.48	1000
9	0.74	0.96	0.84	500
10	0.93	0.68	0.79	500
11	0.90	0.90	0.90	493
12	0.88	0.99	0.93	500
13	0.90	0.55	0.68	500
14	0.61	0.99	0.76	500
15	0.81	0.93	0.86	500
16	0.92	0.07	0.13	476
17	0.61	0.78	0.69	500
18	1.00	0.82	0.90	496
19	0.95	0.96	0.95	500

Table 5.3.4 Classification report

5.3.5 Inceptionnet v3 without transfer learning

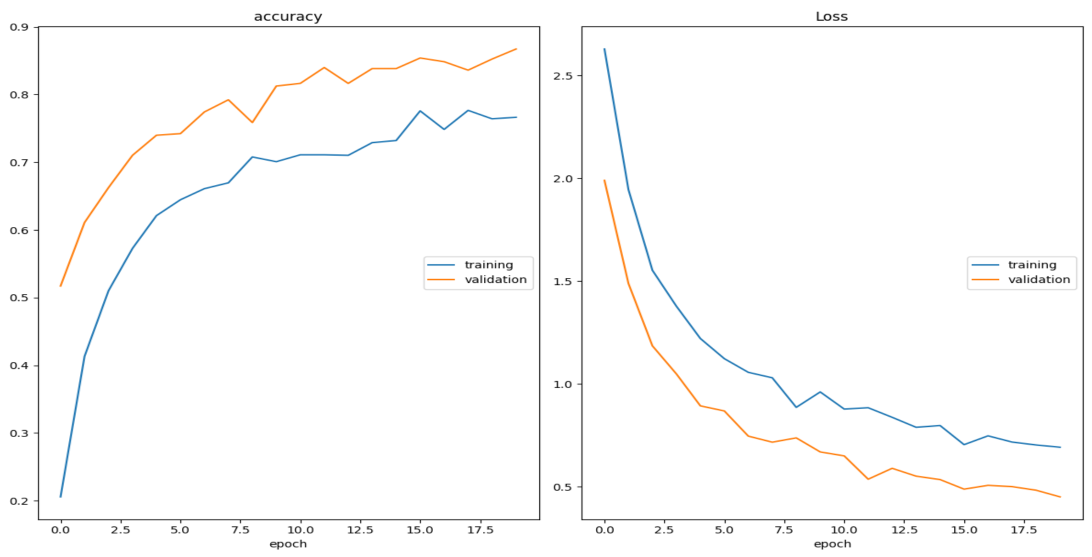
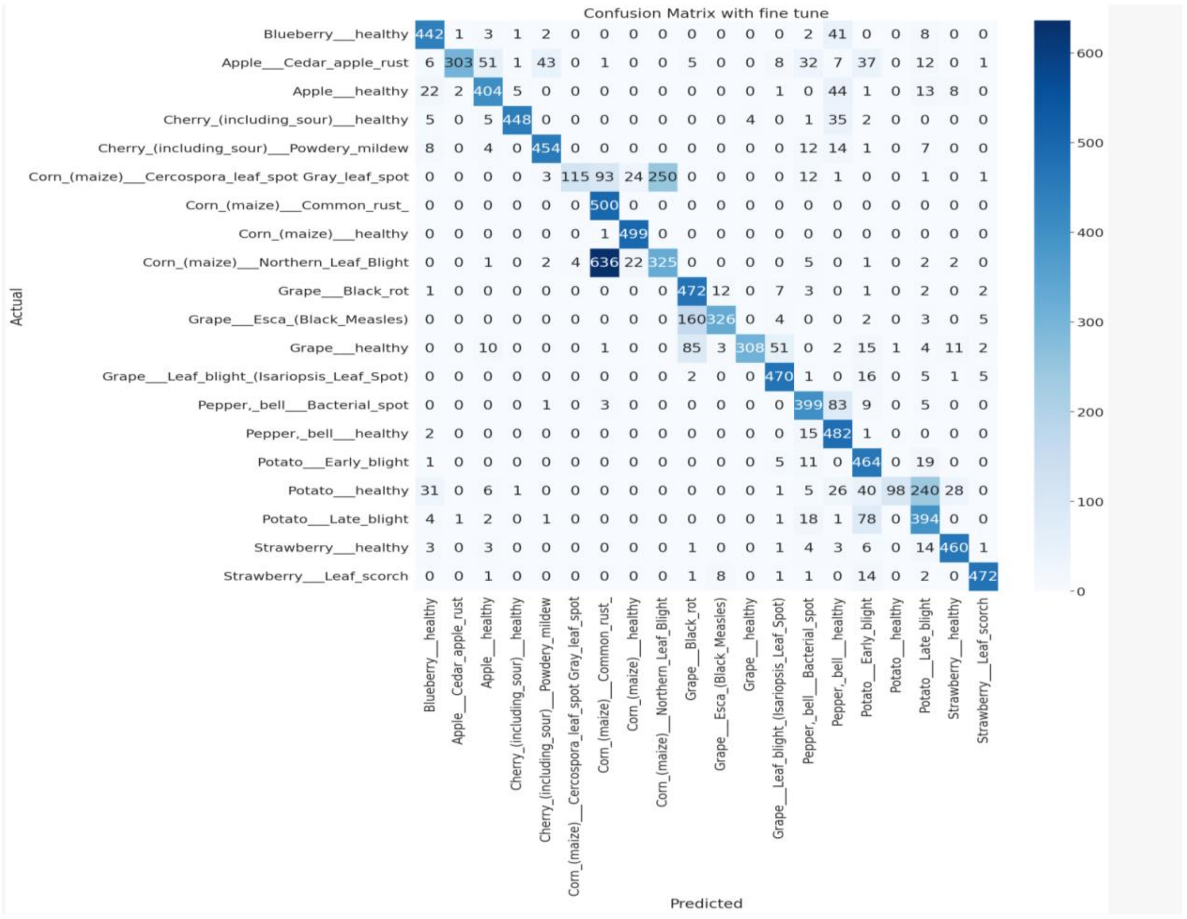


Fig. 5.3.5 Accuracy and loss plot diagram

Confusion Matrix



Actual \ Predicted	Blueberry___healthy	Apple___Cedar_apple_rust	Apple___healthy	Cherry_(including_sour)___healthy	Cherry_(including_sour)___Powdery_mildew	Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	Corn_(maize)___Common_rust_	Corn_(maize)___healthy	Corn_(maize)___Northern_Leaf_Blight	Grape___Black_rot	Grape___Esca_(Black_Measles)	Grape___healthy	Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	Pepper,_bell___Bacterial_spot	Pepper,_bell___healthy	Potato___Early_blight	Potato___healthy	Potato___Late_blight	Strawberry___healthy	Strawberry___Leaf_scorch
Blueberry___healthy	442	1	3	1	2	0	0	0	0	0	0	0	0	2	41	0	0	8	0	0
Apple___Cedar_apple_rust	6	303	51	1	43	0	1	0	0	5	0	0	8	32	7	37	0	12	0	1
Apple___healthy	22	2	404	5	0	0	0	0	0	0	0	0	1	0	44	1	0	13	8	0
Cherry_(including_sour)___healthy	5	0	5	448	0	0	0	0	0	0	0	4	0	1	35	2	0	0	0	0
Cherry_(including_sour)___Powdery_mildew	8	0	4	0	454	0	0	0	0	0	0	0	0	12	14	1	0	7	0	0
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0	0	0	0	3	115	93	24	250	0	0	0	0	12	1	0	0	1	0	1
Corn_(maize)___Common_rust_	0	0	0	0	0	0	500	0	0	0	0	0	0	0	0	0	0	0	0	0
Corn_(maize)___healthy	0	0	0	0	0	0	1	499	0	0	0	0	0	0	0	0	0	0	0	0
Corn_(maize)___Northern_Leaf_Blight	0	0	1	0	2	4	636	22	325	0	0	0	0	5	0	1	0	2	2	0
Grape___Black_rot	1	0	0	0	0	0	0	0	0	472	12	0	7	3	0	1	0	2	0	2
Grape___Esca_(Black_Measles)	0	0	0	0	0	0	0	0	0	160	326	0	4	0	0	2	0	3	0	5
Grape___healthy	0	0	10	0	0	0	1	0	0	85	3	308	51	0	2	15	1	4	11	2
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)	0	0	0	0	0	0	0	0	0	2	0	0	470	1	0	16	0	5	1	5
Pepper,_bell___Bacterial_spot	0	0	0	0	1	0	3	0	0	0	0	0	0	399	83	9	0	5	0	0
Pepper,_bell___healthy	2	0	0	0	0	0	0	0	0	0	0	0	0	15	482	1	0	0	0	0
Potato___Early_blight	1	0	0	0	0	0	0	0	0	0	0	0	5	11	0	464	0	19	0	0
Potato___healthy	31	0	6	1	0	0	0	0	0	0	0	0	1	5	26	40	98	240	28	0
Potato___Late_blight	4	1	2	0	1	0	0	0	0	0	0	0	1	18	1	78	0	394	0	0
Strawberry___healthy	3	0	3	0	0	0	0	0	0	1	0	0	1	4	3	6	0	14	460	1
Strawberry___Leaf_scorch	0	0	1	0	0	0	0	0	0	1	8	0	1	1	0	14	0	2	0	472

Fig. 5.3.5 confusion matrix

Inceptionnet model without fine tune

Parameter name	**Accuracy**	**Loss**
Training	76.64	0.6915
Validation	86.72	0.4501
Test	74.82	---------

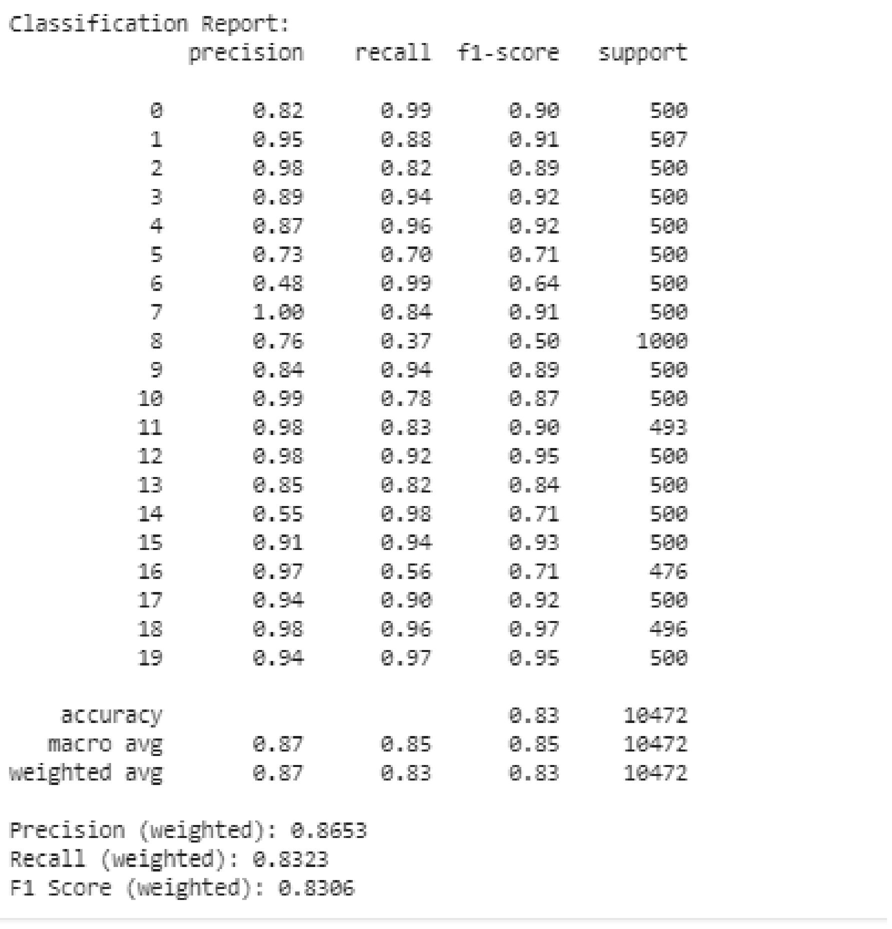
Table 5.3.5 Accuracy and loss result

Class	**Precision**	**Recall**	**F1-score**	**Support**
0	0.84	0.88	0.86	500
1	0.99	0.60	0.74	507
2	0.82	0.81	0.82	500
3	0.98	0.90	0.94	500
4	0.90	0.91	0.90	500
5	0.97	0.23	0.37	500
6	0.40	1.00	0.58	500
7	0.92	1.00	0.96	500
8	0.57	0.33	0.41	1000
9	0.65	0.94	0.77	500
10	0.93	0.65	0.77	500
11	0.99	0.62	0.77	493
12	0.85	0.94	0.90	500
13	0.77	0.80	0.78	500
14	0.65	0.96	0.78	500
15	0.67	0.93	0.78	500
16	0.99	0.21	0.34	476
17	0.54	0.79	0.64	500
18	0.90	0.93	0.91	496
19	0.97	0.94	0.95	500

Table 5.3.5 Classification report

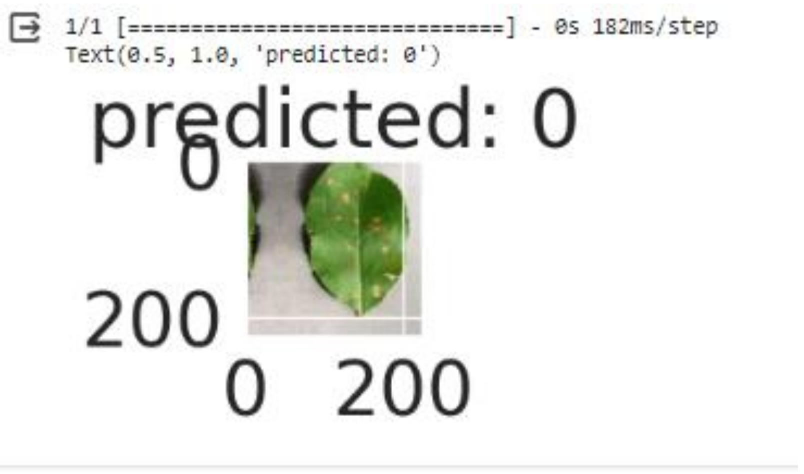
5.4 Output Screens

5.4.1 ResNet50 with Transfer learning model Without Fine tune



```
Classification Report:
              precision    recall  f1-score   support

           0       0.82      0.99      0.90       500
           1       0.95      0.88      0.91       507
           2       0.98      0.82      0.89       500
           3       0.89      0.94      0.92       500
           4       0.87      0.96      0.92       500
           5       0.73      0.70      0.71       500
           6       0.48      0.99      0.64       500
           7       1.00      0.84      0.91       500
           8       0.76      0.37      0.50      1000
           9       0.84      0.94      0.89       500
          10       0.99      0.78      0.87       500
          11       0.98      0.83      0.90       493
          12       0.98      0.92      0.95       500
          13       0.85      0.82      0.84       500
          14       0.55      0.98      0.71       500
          15       0.91      0.94      0.93       500
          16       0.97      0.56      0.71       476
          17       0.94      0.90      0.92       500
          18       0.98      0.96      0.97       496
          19       0.94      0.97      0.95       500

    accuracy                           0.83     10472
   macro avg       0.87      0.85      0.85     10472
weighted avg       0.87      0.83      0.83     10472

Precision (weighted): 0.8653
Recall (weighted): 0.8323
F1 Score (weighted): 0.8306
```

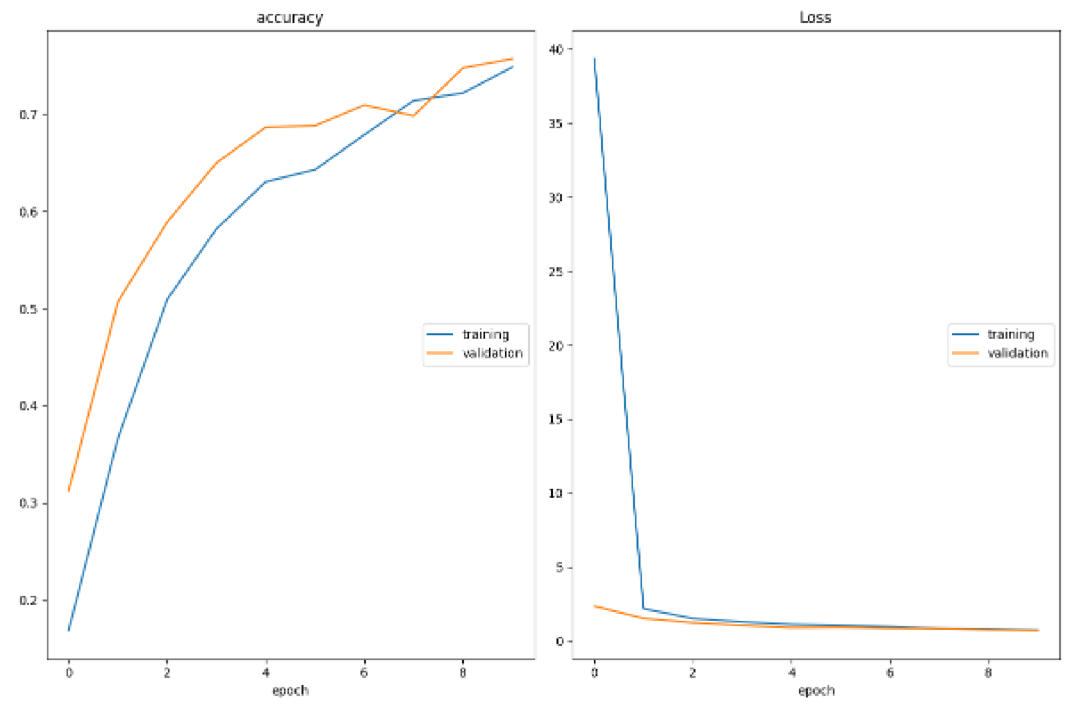
Table 5.4.1 output



```
1/1 [==============================] - 0s 182ms/step
Text(0.5, 1.0, 'predicted: 0')
```

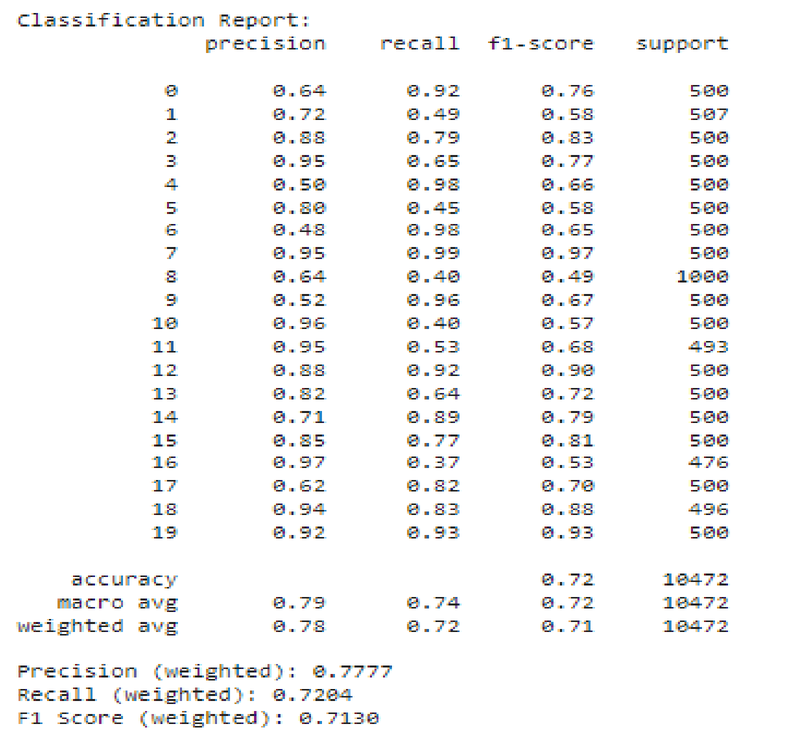
Fig. 5.4.1 output

5.4.2 VggNet16 transfer learning model without fine tune



```
accuracy
        training                  (min:    0.169, max:    0.748, cur:    0.748)
        validation                (min:    0.312, max:    0.757, cur:    0.757)
Loss
        training                  (min:    0.739, max:   39.329, cur:    0.739)
        validation                (min:    0.732, max:    2.370, cur:    0.732)
20/20 [==============================] - 1094s 56s/step - loss: 0.7389 - accuracy: 0.7484 - val_loss: 0.7319 - val_accuracy: 0.7570
```

Fig. 5.4.2 output



```
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.92      0.76       500
           1       0.72      0.49      0.58       507
           2       0.88      0.79      0.83       500
           3       0.95      0.65      0.77       500
           4       0.50      0.98      0.66       500
           5       0.80      0.45      0.58       500
           6       0.48      0.98      0.65       500
           7       0.95      0.99      0.97       500
           8       0.64      0.40      0.49      1000
           9       0.52      0.96      0.67       500
          10       0.96      0.40      0.57       500
          11       0.95      0.53      0.68       493
          12       0.88      0.92      0.90       500
          13       0.82      0.64      0.72       500
          14       0.71      0.89      0.79       500
          15       0.85      0.77      0.81       500
          16       0.97      0.37      0.53       476
          17       0.62      0.82      0.70       500
          18       0.94      0.83      0.88       496
          19       0.92      0.93      0.93       500

    accuracy                           0.72     10472
   macro avg       0.79      0.74      0.72     10472
weighted avg       0.78      0.72      0.71     10472

Precision (weighted): 0.7777
Recall (weighted): 0.7204
F1 Score (weighted): 0.7130
```

Table 5.4.2 Output

```
1/1 [==============================] - 1s 820ms/step
Text(0.5, 1.0, 'predicted: 0')
```

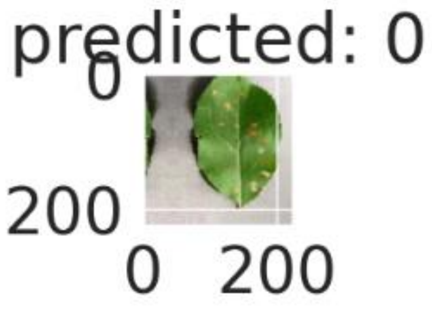
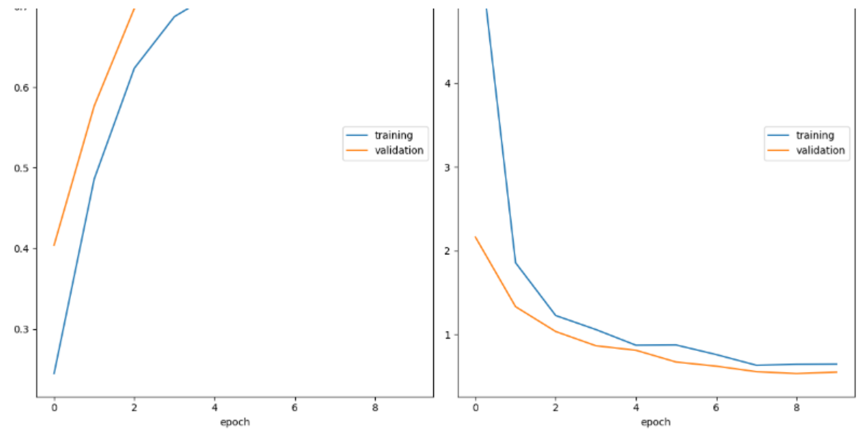


Fig. 5.4.2 output 1

5.4.3 VggNet16 transfer learning model with fine tune



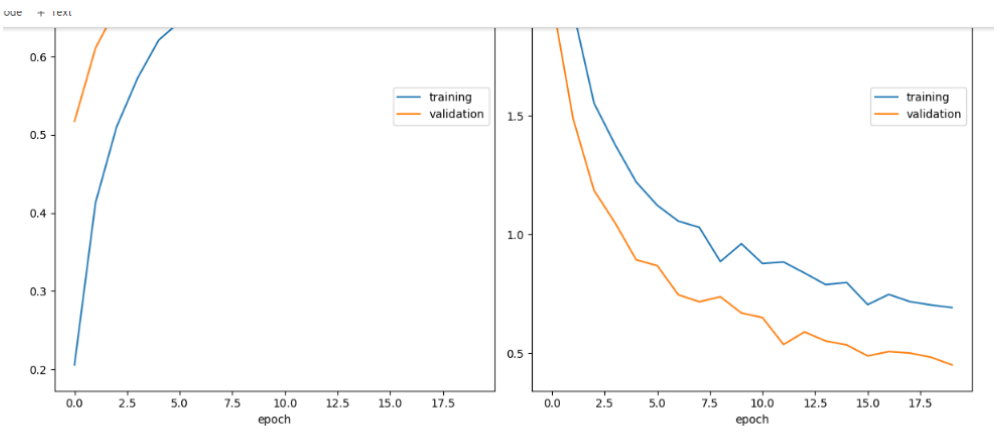
```
accuracy
        training             (min:    0.245, max:    0.806, cur:    0.799)
        validation           (min:    0.404, max:    0.816, cur:    0.816)
Loss
        training             (min:    0.634, max:    6.029, cur:    0.647)
        validation           (min:    0.534, max:    2.162, cur:    0.551)
20/20 [==============================] - 1560s 80s/step - loss: 0.6467 - accuracy: 0.7992 - val_loss: 0.5514 - val_accuracy: 0.8156
```

Fig. 5.4.3 output

```
Classification Report:
              precision    recall  f1-score   support

           0       0.63      0.98      0.77       500
           1       0.80      0.71      0.75       507
           2       0.86      0.94      0.90       500
           3       0.95      0.93      0.94       500
           4       0.69      0.97      0.81       500
           5       0.74      0.60      0.66       500
           6       0.49      0.89      0.63       500
           7       0.98      0.95      0.96       500
           8       0.68      0.41      0.51      1000
           9       0.85      0.87      0.86       500
          10       0.93      0.83      0.88       500
          11       0.94      0.89      0.91       493
          12       0.68      0.98      0.80       500
          13       0.89      0.81      0.85       500
          14       0.87      0.87      0.87       500
          15       0.95      0.57      0.71       500
          16       0.95      0.45      0.61       476
          17       0.73      0.85      0.79       500
          18       0.97      0.85      0.91       496
          19       0.95      0.87      0.91       500

    accuracy                           0.79     10472
   macro avg       0.83      0.81      0.80     10472
weighted avg       0.82      0.79      0.79     10472

Precision (weighted): 0.8193
Recall (weighted): 0.7927
F1 Score (weighted): 0.7881
```

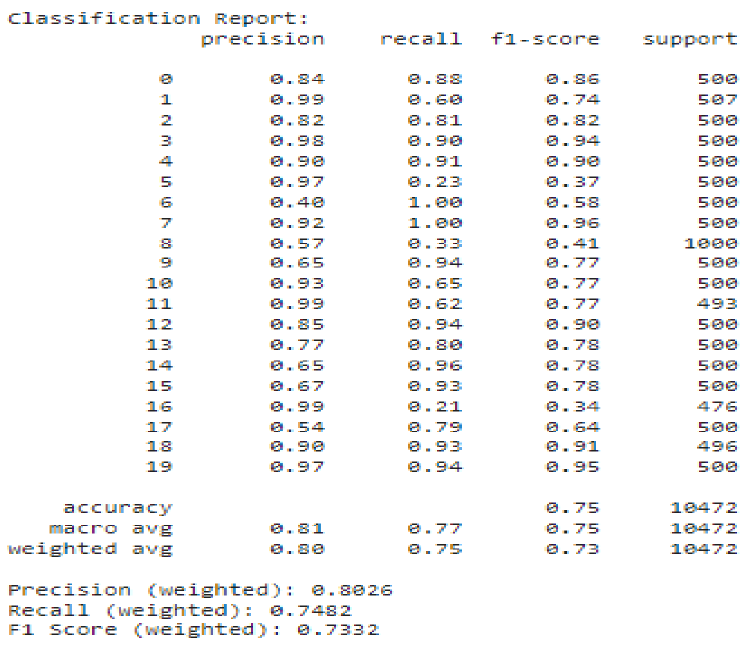
Fig. 5.4.3 output 1

5.4.4 InceptionNet V3 transfer learning model without fine tune



```
accuracy
        training                (min:    0.206, max:    0.777, cur:    0.766)
        validation              (min:    0.517, max:    0.867, cur:    0.867)
Loss
        training                (min:    0.691, max:    2.629, cur:    0.691)
        validation              (min:    0.450, max:    1.990, cur:    0.450)
20/20 [==============================] - 724s 37s/step - loss: 0.6915 - accuracy: 0.7664 - val_loss: 0.4501 - val_accuracy: 0.8672
```

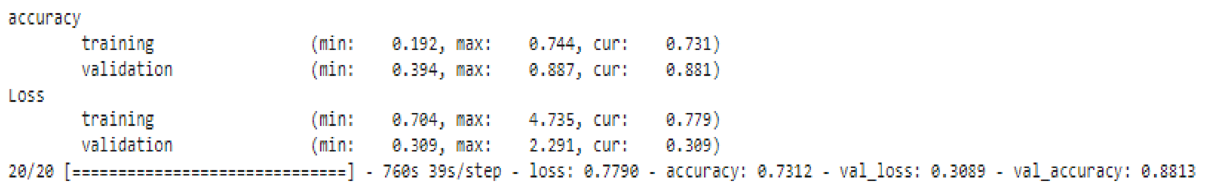
Fig. 5.4.4 output



```
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       500
           1       0.99      0.60      0.74       507
           2       0.82      0.81      0.82       500
           3       0.98      0.90      0.94       500
           4       0.90      0.91      0.90       500
           5       0.97      0.23      0.37       500
           6       0.40      1.00      0.58       500
           7       0.92      1.00      0.96       500
           8       0.57      0.33      0.41      1000
           9       0.65      0.94      0.77       500
          10       0.93      0.65      0.77       500
          11       0.99      0.62      0.77       493
          12       0.85      0.94      0.90       500
          13       0.77      0.80      0.78       500
          14       0.65      0.96      0.78       500
          15       0.67      0.93      0.78       500
          16       0.99      0.21      0.34       476
          17       0.54      0.79      0.64       500
          18       0.90      0.93      0.91       496
          19       0.97      0.94      0.95       500

    accuracy                           0.75     10472
   macro avg       0.81      0.77      0.75     10472
weighted avg       0.80      0.75      0.73     10472

Precision (weighted): 0.8026
Recall (weighted): 0.7482
F1 Score (weighted): 0.7332
```

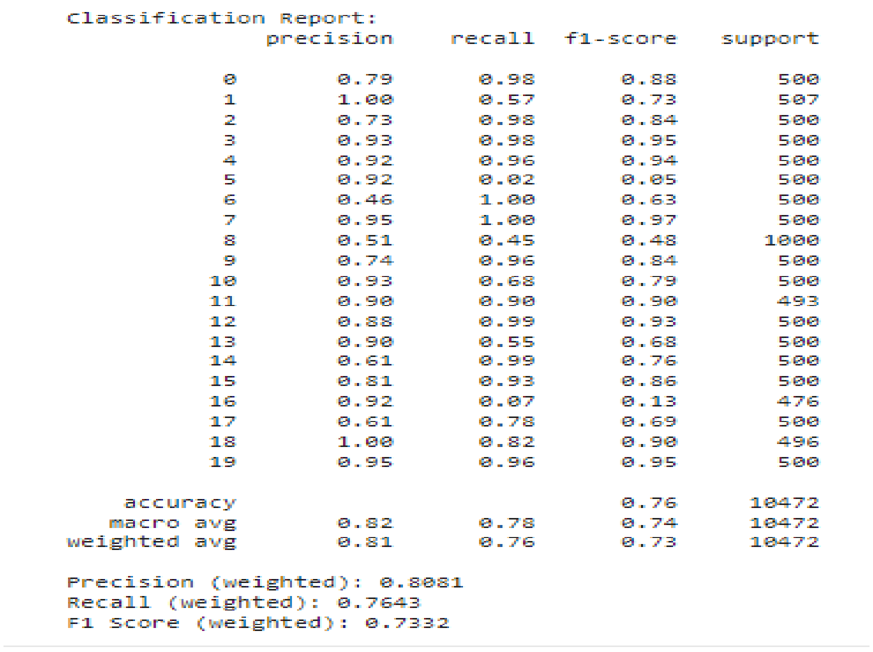
Fig. 5.4.4 output 1

5.4.5 InceptionNet V3 transfer learning model with fine tune



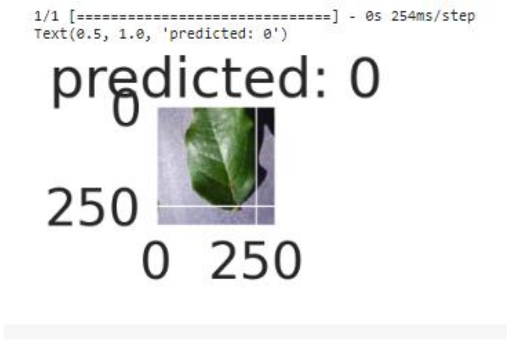
```
accuracy
        training                (min:    0.192, max:    0.744, cur:    0.731)
        validation              (min:    0.394, max:    0.887, cur:    0.881)
Loss
        training                (min:    0.704, max:    4.735, cur:    0.779)
        validation              (min:    0.309, max:    2.291, cur:    0.309)
20/20 [==============================] - 760s 39s/step - loss: 0.7790 - accuracy: 0.7312 - val_loss: 0.3089 - val_accuracy: 0.8813
```

Fig. 5.4.5 output

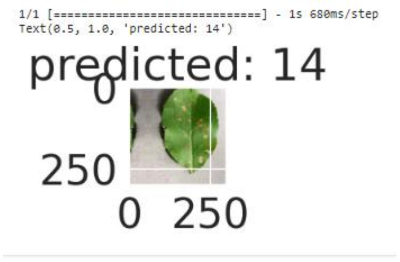


```
Classification Report:
              precision    recall  f1-score   support

           0       0.79      0.98      0.88       500
           1       1.00      0.57      0.73       507
           2       0.73      0.98      0.84       500
           3       0.93      0.98      0.95       500
           4       0.92      0.96      0.94       500
           5       0.92      0.02      0.05       500
           6       0.46      1.00      0.63       500
           7       0.95      1.00      0.97       500
           8       0.51      0.45      0.48      1000
           9       0.74      0.96      0.84       500
          10       0.93      0.68      0.79       500
          11       0.90      0.90      0.90       493
          12       0.88      0.99      0.93       500
          13       0.90      0.55      0.68       500
          14       0.61      0.99      0.76       500
          15       0.81      0.93      0.86       500
          16       0.92      0.07      0.13       476
          17       0.61      0.78      0.69       500
          18       1.00      0.82      0.90       496
          19       0.95      0.96      0.95       500

    accuracy                           0.76     10472
   macro avg       0.82      0.78      0.74     10472
weighted avg       0.81      0.76      0.73     10472

Precision (weighted): 0.8081
Recall (weighted): 0.7643
F1 Score (weighted): 0.7332
```

Fig. 5.4.5 output 1



```
1/1 [==============================] - 0s 254ms/step
Text(0.5, 1.0, 'predicted: 0')
```



```
1/1 [==============================] - 1s 680ms/step
Text(0.5, 1.0, 'predicted: 14')
```

Fig. 5.4.5 output 2

Fig. 5.4.5 output 3

5.5 Results

The experimental findings show that the assessed pre-trained models applying transfer learning for image classification have different performance traits. ResNet50 was the most impressive performer, with the best accuracy of 83.23%. This conclusion is consistent with the literature's general trend, which highlights the effectiveness of deeper architectures—especially those that incorporate residual learning—for better feature representation and classification accuracy. The accuracy of the VGG16 model was improved to a notable 79.27% after fine-tuning. This result emphasizes how flexible VGG16 is when task-specific modifications are implemented, corroborating earlier studies that promote fine-tuning as an effective method for improving trained models.

InceptionV3 showed competitive performance with and without fine-tuning. With an accuracy of 76.43%, the untuned InceptionV3 demonstrated a trade-off between computing efficiency and accuracy. A further indication of the refined InceptionV3's adaptability in attaining a trade-off between model complexity and performance is its accuracy of 74.82%. These findings give practitioners practical advice on how to choose a model depending on the demands of a given activity and the computational resources at hand. ResNet50 performs well in situations where high accuracy is critical, while VGG16 is a strong option for task-specific optimization due to its fine-tuning-capable flexibility. Efficiency-wise, InceptionV3 shows to be a reasonable choice because to its well-balanced performance. All things considered, this thorough research adds significant knowledge to the larger field of pre-trained models, helping practitioners make judgments that are appropriate for their practical image classification applications.

5.6 Comparative Analysis with Existing works

The comparative examination of several pre-trained models offers important insights into how well they perform on real-world tasks within the framework of current picture classification research and applications. With the best accuracy and great precision, recall, and F1 scores, the ResNet50-based model stands out as a formidable competitor. This is consistent with a larger body of literature highlighting the superior performance of deeper architectures like ResNet50 in image identification tasks. When calibrated, the VGG16 model exhibits significant gains in accuracy and provides a workable alternative for situations where model refinement is essential. The results of this investigation show that fine-tuning is effective in modifying pre-trained models to fit particular domains, which is in line with other studies' conclusions. The InceptionV3 models offer a trade-off between computational efficiency and accuracy, both with and without fine-tuning. This is consistent with the literature, where InceptionV3 has been acknowledged for its capacity to strike a balance between resource usage and performance, qualifying it for implementation in a range of real-world applications.In actuality, the selection of a pre-trained model is contingent upon the particular demands of the picture classification task as well as the computational resources at hand. The results indicate that the ResNet50-based model can be used by practitioners for jobs requiring a high degree of accuracy, while the fine-tuned VGG16 model provides a flexible trade-off between performance and flexibility. These observations add to the continuing discussion in the field and help practitioners choose models that meet the requirements and limitations of their practical uses.

Chapter 6

CONCLUSION AND FUTURE ENHANCEMENTS

In conclusion, a comparison study of transfer learning-based pre-trained models for plant disease detection classification, such as ResNet50, VGG16, and InceptionV3, has shed light on how well each model performs. The most accurate model was ResNet50 transfer learning with its deep residual learning architecture, demonstrating the effectiveness of using pre-trained features for better image classification. VGG16's adaptability was shown through fine-tuning, which led to a notable increase in accuracy. InceptionV3 demonstrated competitive performance with and without fine-tuning, demonstrating its adaptability in striking a compromise between computing economy and accuracy. The analysis's findings advance our knowledge of how transfer learning affects picture classification tasks and provide practitioners with useful direction when choosing models that meet their unique needs. There are a number of directions that could be taken to improve image classification in the future. Improving model flexibility may be possible by investigating self-supervised learning and domain-specific pre-training strategies. Strategies for fine-tuning, such as layer-wise modifications and dynamic adaptation, offer chances to maximize transfer learning methodologies. Models' generalization skills will be better understood by evaluating them on a variety of datasets and multimodal tasks. A deeper understanding of model behavior will also be provided by combining quantitative and qualitative evaluations, such as representations of learned features. These future directions have the potential to advance transfer learning approaches and improve the applicability of pre-trained models to a wider range of real-world settings, which will be beneficial to the discipline.

REFERENCES

[1] N, Banupriya & chowdry, Rajaneni & Yogeshwari, & V,Varsha,.(2022), ‘PLANT DISEASE DETECTION USING IMAGE PROCESSING AND MACHINE LEARNING ALGORITHM’, 10.37896/jxu14.7/012.

[2] Yan Guo, Jin Zhang, Chengxin Yin, Xiaonan Hu, Yu Zou, Zhipeng Xue, Wei Wang. (2020) "Plant Disease Identification Based on Deep Learning Algorithm in Smart Farming", Discrete Dynamics in Nature and Society, vol. 2020, Article ID 2479172, 11 pages,. https://doi.org/10.1155/2020/2479172

[3] J., Andrew, Jennifer Eunice, Daniela Elena Popescu, M. Kalpana Chowdary, and Jude Hemanth. (2022)."Deep Learning-Based Leaf Disease Detection in Crops Using Images for Agricultural Applications" *Agronomy* 12, no. 10: 2395. https://doi.org/10.3390/agronomy12102395.

[4] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016) “Using Deep Learning for Image-Based Plant Disease Detection”. *Frontiers in plant science*, *7,* 1419. https://doi.org/10.3389/fpls.2016.01419

[5] Chen, R., Qi, H., Liang, Y., & Yang, M. (2022). ”Identification of plant leaf diseases by deep learning based on channel attention and channel pruning”. *Frontiers in plant science*, *13,* 1023515. https://doi.org/10.3389/fpls.2022.1023515

[6] Jung, M., Song, J. S., Shin, A. Y., Choi, B., Go, S., Kwon, S. Y., Park, J., Park, S. G., & Kim, Y. M. (2023). “Construction of deep learning-based disease detection model in plants”. *Scientific reports*, *13*(1), 7331. https://doi.org/10.1038/s41598-023-34549-2

[7] Chohan, Murk & Khan, Adil & Chohan, Rozina & Katper, Saif & Mahar, Muhammad. (2020). “Plant Disease Detection using Deep Learning”. International Journal of Recent Technology and Engineering. 9. 909-914. 10.35940/ijrte.A2139.059120.

Appendices

Appendix A

SAMPLE CODE

1 - Data Augmentation section

```
from keras.preprocessing.image import ImageDataGenerator
from skimage import io
import numpy as np
import os
from PIL import Image
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from matplotlib.pyplot import imread, imshow, subplot, show
import numpy as np
import os
from PIL import Image
from skimage import io
datagen = ImageDataGenerator(
        featurewise_center=False,
        samplewise_center=False,
        featurewise_std_normalization=False,
        samplewise_std_normalization=False,
        zca_whitening=False,
#         zca_epsilon=1e-06,
        rotation_range=40,
        width_shift_range=0.2,
        height_shift_range=0.2,
#         brightness_range=None,
        shear_range=0.2,
        zoom_range=0.2,
        channel_shift_range=0.0,
        fill_mode="reflect",
        cval=0.0,
        horizontal_flip=True,
        vertical_flip=True,
        rescale=None,
        preprocessing_function=None,
        data_format=None,
#         validation_split=0.0,
#         dtype=None,
        )
image_directory = '/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/
DATASET/PlantVillage-Dataset/Apple___Apple_scab/'
SIZE = 224
dataset = []
my_images = os.listdir(image_directory)
for i, image_name in enumerate(my_images):
    if ((image_name.split('.')[1] == 'jpg') or (image_name.split('.')[1] == 'JPG')):
        image = io.imread(image_directory + image_name)
        image = Image.fromarray(image, 'RGB')
        image = image.resize((SIZE,SIZE))
        dataset.append(np.array(image))

x = np.array(dataset)
batch_size = 20 # how many images should be created at a time
num_of_count = 50
num_of_images = batch_size * num_of_count

count = 1
for batch in datagen.flow(x, batch_size=10, save_to_dir='/content/drive/MyDrive/VIT PROJECT
WORK/Mater Project VIT FINAL SEM/ DATASET/ Augmented Dataset/ Aug_Apple___Apple_scab',
                        save_prefix='aug', save_format='jpg'):
    count += 1
    if count > num_of_count:
        break

print("%d images are generated" % num_of_images)
```

2. Dataset Details

2.1 Training dataset

```
import os
# Specify the path to the directory containing your dataset
dataset_path = "/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/
DATASET/PlantVillage-Dataset/ dataset_original"
class_labels = os.listdir(dataset_path)   # Get a list of class labels (subdirectories in the dataset path)
class_counts = {}    # Initialize a dictionary to store the count for each class
total_count = 0      # Initialize a variable to store the total count

# Loop through each class label
for label in class_labels:
    class_path = os.path.join(dataset_path, label)        # Construct the path to the class directory
    num_images = len(os.listdir(class_path))         # Count the number of images in the class directory
    class_counts[label] = num_images                 # Store the count in the dictionary
    total_count += num_images                        # Increment the total count

for label, count in class_counts.items():            # Print the class labels and their respective counts
    print(f"Class: {label}, Number of Images: {count}")
print(f"Total Number of Images: {total_count}")       # Print the total number of images
```

2.2 Test dataset

```
# Specify the path to the directory containing your dataset
dataset_path = "/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/
DATASET/PlantVillage-Dataset/ Augmented Dataset"
class_labels = os.listdir(dataset_path)# Get a list of class labels (subdirectories in the dataset path)
class_counts = {}# Initialize a dictionary to store the count for each class
total_count = 0# Initialize a variable to store the total count

for label in class_labels:# Loop through each class label
    class_path = os.path.join(dataset_path, label)# Construct the path to the class directory
    num_images = len(os.listdir(class_path))
    class_counts[label] = num_images
    total_count += num_images
for label, count in class_counts.items():
    print(f"Class: {label}, Number of Images: {count}")
print(f"Total Number of Images: {total_count}")
```

1 - ResNet50 model with transfer learning without fine tuning for 20 classes of dataset

```
import os
from keras.models import Model
from keras.optimizers import Adam
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ModelCheckpoint, EarlyStopping
from keras.layers import Dense, Dropout, Flatten
from pathlib import Path
import numpy as np
from keras.applications.resnet50 import preprocess_input
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.python.keras.callbacks import EarlyStopping, ModelCheckpoint

NUM_CLASSES = 20
CHANNELS = 3
IMAGE_RESIZE = 224
RESNET50_POOLING_AVERAGE = 'avg'
DENSE_LAYER_ACTIVATION = 'softmax'
OBJECTIVE_FUNCTION = 'categorical_crossentropy'
LOSS_METRICS = ['accuracy']
NUM_EPOCHS = 20
EARLY_STOP_PATIENCE = 3
STEPS_PER_EPOCH_TRAINING = 10
STEPS_PER_EPOCH_VALIDATION = 10
BATCH_SIZE_TRAINING = 100
BATCH_SIZE_VALIDATION = 100
BATCH_SIZE_TESTING = 1
image_size = IMAGE_RESIZE
BATCH_SIZE = 64

train_generator = ImageDataGenerator(rotation_range=90,
                                     brightness_range=[0.1, 0.7],
                                     width_shift_range=0.5,
                                     height_shift_range=0.5,
                                     horizontal_flip=True,
                                     vertical_flip=True,
                                     validation_split=0.15,
                                     preprocessing_function=preprocess_input)

test_generator = ImageDataGenerator(preprocessing_function=preprocess_input)

download_dir = Path('/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/
DATASET/PlantVillage-Dataset')
```

```
train_data_dir = download_dir/' dataset_original'
test_data_dir = download_dir/' Augmented Dataset'

class_subset = os.listdir(download_dir/' dataset_original')
traingen = train_generator.flow_from_directory(train_data_dir,
                                               target_size=(224, 224),
                                               class_mode='categorical',
                                               classes=class_subset,
                                               subset='training',
                                               batch_size=BATCH_SIZE,
                                               shuffle=True,
                                               seed=42)

validgen = train_generator.flow_from_directory(train_data_dir,
                                               target_size=(224, 224),
                                               class_mode='categorical',
                                               classes=class_subset,
                                               subset='validation',
                                               batch_size=BATCH_SIZE,
                                               shuffle=True,
                                               seed=42)

testgen = test_generator.flow_from_directory(test_data_dir,
                                             target_size=(224, 224),
                                             class_mode=None,
                                             classes=class_subset,
                                             batch_size=1,
                                             shuffle=False,
                                             seed=42)

cb_early_stopper = EarlyStopping(monitor = 'val_loss', patience = EARLY_STOP_PATIENCE)
cb_checkpointer = ModelCheckpoint(filepath = '/content/drive/MyDrive/VIT PROJECT WORK/Mater
Project VIT FINAL SEM/model notebook/my_model_tl_ft_res333.keras', monitor = 'val_loss',
save_best_only = True,
                              mode='min',
                              save_freq=1,
                              verbose=1)

model = Sequential()

# NOTE that this layer will be set below as NOT TRAINABLE
model.add(ResNet50(include_top = False, pooling = RESNET50_POOLING_AVERAGE,
weights='imagenet'))
# 2nd layer as Dense for 20-class classification using SoftMax activation
model.add(Dense(NUM_CLASSES, activation = DENSE_LAYER_ACTIVATION))
# not to train first layer (ResNet) model as it is already trained
model.layers[0].trainable = False
model.summary()

from tensorflow.keras.optimizers import SGD
import tensorflow as tf
#sgd = SGD(learning_rate=0.01, decay=1e-6, momentum=0.9, nesterov=True)
```

```
sgd = tf.keras.optimizers.legacy.SGD(learning_rate = 0.01, decay = 1e-6, momentum = 0.9, nesterov =
True)
model.compile(optimizer = sgd, loss = OBJECTIVE_FUNCTION, metrics = LOSS_METRICS)
#train the model
fit_history = model.fit(
        traingen,
        steps_per_epoch=STEPS_PER_EPOCH_TRAINING,
        epochs = NUM_EPOCHS,
        validation_data=validgen,
        validation_steps=STEPS_PER_EPOCH_VALIDATION,
        callbacks=[cb_checkpointer, cb_early_stopper]
)
model.save("/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/model
notebook/my_model_tl_ft_res333.keras")

print(fit_history.history.keys())
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import precision_score, recall_score, f1_score

plt.figure(1, figsize = (15,8))

plt.subplot(221)
plt.plot(fit_history.history['accuracy'])
plt.plot(fit_history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'valid'])

plt.subplot(222)
plt.plot(fit_history.history['loss'])
plt.plot(fit_history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'valid'])

plt.show()
# Generate predictions
true_classes = testgen.classes
class_indices = traingen.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())

preds = model.predict(testgen)
pred_classes = np.argmax(preds, axis=1)
from sklearn.metrics import accuracy_score
acc = accuracy_score(true_classes, pred_classes)
print("Model Accuracy without Fine-Tuning: {:.2f}%".format(acc * 100))
# Compute the confusion matrix
cm = confusion_matrix(true_classes, pred_classes)
```

```
# Define class names if you have them
class_names = testgen.class_indices.keys()
# Create a heatmap
plt.figure(figsize=(20,10))
sns.set(font_scale=1.2)  # Adjust the font size as needed
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=class_names, yticklabels=class_names)
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix with fine tune')
plt.show()


y_pred = preds
y_true = testgen.classes
y_pred_classes = np.argmax(y_pred, axis=1)

# Calculate metrics
cm = confusion_matrix(y_true, y_pred_classes)
classification_rep = classification_report(y_true, y_pred_classes)
precision = precision_score(y_true, y_pred_classes, average='weighted')
recall = recall_score(y_true, y_pred_classes, average='weighted')
f1 = f1_score(y_true, y_pred_classes, average='weighted')

# Display the confusion matrix
print("Confusion Matrix:")
print(cm)

# Display classification report, precision, recall, and F1 score
print("\nClassification Report:")
print(classification_rep)
print(f"Precision (weighted): {precision:.4f}")
print(f"Recall (weighted): {recall:.4f}")
print(f"F1 Score (weighted): {f1:.4f}")

from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.resnet50 import preprocess_input, decode_predictions
import numpy as np

# Load an image from file
img_path = '/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/
DATASET/PlantVillage-Dataset/ Augmented Dataset/Apple___Cedar_apple_rust/aug_0_3915.jpg'
img = image.load_img(img_path, target_size=(224, 224))

# Convert the image to a numpy array
img_array = image.img_to_array(img)

# Add an extra dimension and preprocess the image
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
i=3
# Make the prediction
predictions = model.predict(img_array)
#print(predictions)
```

```
predicted = pred_classes[np.argmax(predictions)]
plt.subplot(240+1+i)
plt.imshow(img)
plt.title('predicted: ' + str(predicted))
```

2.1 - VGG16 model with transfer learning with fine tuning for 20 classes of dataset

```
def create_model(input_shape, n_classes, optimizer='rmsprop', fine_tune=0):
   conv_base = VGG16(include_top=False,
                weights='imagenet',
                input_shape=input_shape)
   if fine_tune > 0:
      for layer in conv_base.layers[:-fine_tune]:
         layer.trainable = False
   else:
      for layer in conv_base.layers:
         layer.trainable = False

   top_model = conv_base.output
   top_model = Flatten(name="flatten")(top_model)
   top_model = Dense(4096, activation='relu')(top_model)
   top_model = Dense(1072, activation='relu')(top_model)
   top_model = Dropout(0.2)(top_model)
   output_layer = Dense(n_classes, activation='softmax')(top_model)

   # Group the convolutional base and new fully-connected layers into a Model object.
   model = Model(inputs=conv_base.input, outputs=output_layer)

   # Compiles the model for training.
   model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
   model.summary()
   return model

input_shape = (224, 224, 3)
n_classes=20

n_steps =20
n_val_steps =20
n_epochs = 10
tl_checkpoint_1 = ModelCheckpoint(filepath='/content/drive/MyDrive/VIT PROJECT WORK/Mater
Project VIT FINAL SEM/model notebook/vggnet16_t2_another_2.h5',
                          save_best_only=True,
                          verbose=1)
# EarlyStopping
early_stop = EarlyStopping(monitor='val_loss',
                     patience=10,
                     restore_best_weights=True,
                     mode='min')

# Use a smaller learning rate
```

```
optim_2 = tf.keras.optimizers.legacy.Adam(learning_rate=0.0001)

# Re-compile the model, this time leaving the last 2 layers unfrozen for Fine-Tuning
vgg_model_2 = create_model(input_shape, n_classes, optim_2, fine_tune=2)
plot_loss_2 = PlotLossesCallback()

# Retrain model with fine-tuning
vgg_ft_history = vgg_model_2.fit(traingen,
                                  batch_size=BATCH_SIZE,
                                  epochs=n_epochs,
                                  validation_data=validgen,
                                  steps_per_epoch=n_steps,
                                  validation_steps=n_val_steps,
                                  callbacks=[tl_checkpoint_1, early_stop, plot_loss_2],
                                  verbose=1)
filepath="/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/model notebook/vggnet16_t2_another_2.h5"
vgg_model_2.save(filepath)
true_classes = testgen.classes
class_indices = traingen.class_indices
class_indices = dict((v,k) for k,v in class_indices.items())

vgg_preds_2 = vgg_model_2.predict(testgen)
vgg_pred_classes_2 = np.argmax(vgg_preds_2, axis=1)
from sklearn.metrics import accuracy_score
vgg_acc_2 = accuracy_score(true_classes, vgg_pred_classes_2)
print("VGG16 Model Accuracy with Fine-Tuning: {:.2f}%".format(vgg_acc_2 * 100))
```

3.1 - inceptionnet v3 transfer learning with fine tune

```
#Using Pre-trained Layers for Feature Extraction
def create_model(input_shape, n_classes, optimizer='adam', fine_tune=0):

    conv_base = InceptionV3(weights='imagenet', include_top=False, input_shape=(299, 299, 3))
    if fine_tune > 0:
        for layer in conv_base.layers[:-fine_tune]:
            layer.trainable = False
    else:
        for layer in conv_base.layers:
            layer.trainable = False

    top_model = conv_base.output
    #top_model = Flatten(name="flatten")(top_model)
    top_model = GlobalAveragePooling2D()(top_model)
    #top_model = Dense(1072, activation='relu')(top_model)
    top_model = Dense(256, activation='relu')(top_model)
    #top_model = Dropout(0.2)(top_model)
    top_model = Dropout(0.5)(top_model)
    output_layer = Dense(n_classes, activation='softmax')(top_model)

    #base_model,
    #layers.GlobalAveragePooling2D(),
    #layers.Dense(256, activation='relu'),
    #layers.Dropout(0.5),
    #layers.Dense(train_data.num_classes, activation='softmax')

    # Group the convolutional base and new fully-connected layers into a Model object.
    model = Model(inputs=conv_base.input, outputs=output_layer)

    # Compiles the model for training.
    model.compile(optimizer=optimizer,
                  loss='categorical_crossentropy',
                  metrics=['accuracy'])
    model.summary()
    return model
    #model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

input_shape = (299, 299, 3)
optim_1 = Adam(learning_rate=0.01)
n_classes=20
n_steps =20 # traingen.samples // BATCH_SIZE
n_val_steps =20 # validgen.samples // BATCH_SIZE
n_epochs = 20

model = create_model(input_shape, n_classes, optim_1, fine_tune=20)

!pip install livelossplot
from livelossplot.inputs.keras import PlotLossesCallback

plot_loss_1 = PlotLossesCallback()
tl_checkpoint_1 = ModelCheckpoint(filepath='/content/drive/MyDrive/VIT PROJECT WORK/Mater
Project VIT FINAL SEM/model notebook/16_t2_another_tl2_2.h5',
                                  save_best_only=True,
                                  verbose=1)
# EarlyStopping
early_stop = EarlyStopping(monitor='val_loss',
                           patience=10,
                           restore_best_weights=True,
                           mode='min')
#We can now train the model defined above:

history_0 = model.fit(traingen,
                      batch_size=BATCH_SIZE,
                      epochs=n_epochs,
                      validation_data=validgen,
                      steps_per_epoch=n_steps,
                      validation_steps=n_val_steps,
                      callbacks=[tl_checkpoint_1, early_stop, plot_loss_1],
                      verbose=1)

#Saving our model
filepath="/content/drive/MyDrive/VIT PROJECT WORK/Mater Project VIT FINAL SEM/model
notebook/16_t2_another_tl2_2.h5"
model.save(filepath)
```

PLANT DISEASES DETECTION USING DEEP LEARNING AND MACHINE VISION

ORIGINALITY REPORT

14%	9%	8%	6%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	medium.com Internet Source	2%
2	www.jesusutrera.com Internet Source	1%
3	spj.sciencemag.org Internet Source	1%
4	Ruzhun Zhao, Yuchang Zhu, Yuanhong Li. "CLA: A self-supervised contrastive learning method for leaf disease identification with domain adaptation", Computers and Electronics in Agriculture, 2023 Publication	1%