LIGHTWEIGHT AND FEW-SHOT IMAGE-BASED PLANT DISEASE DIAGNOSIS AND REMEDY RECOMMENDER SYSTEM

THESIS SUBMITTED TO THE UNIVERSITY OF DELHI
FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

PUSHKAR GOLE

UNDER THE SUPERVISION OF

SR. PROF. PUNAM BEDI

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF DELHI
DELHI - 110007
INDIA

SEPTEMBER 2024

LIGHTWEIGHT AND FEW-SHOT IMAGE-BASED PLANT DISEASE DIAGNOSIS AND REMEDY RECOMMENDER SYSTEM

THESIS SUBMITTED TO THE UNIVERSITY OF DELHI
FOR THE AWARD OF THE DEGREE OF

DOCTOR OF PHILOSOPHY

BY

PUSHKAR GOLE

UNDER THE SUPERVISION OF

SR. PROF. PUNAM BEDI

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF DELHI
DELHI - 110007
INDIA

SEPTEMBER 2024

DECLARATION

This is to declare that the work presented in this thesis titled "**Lightweight and Few-Shot Image-based Plant Disease Diagnosis and Remedy Recommender System**" has been carried out by me in the Department of Computer Science, University of Delhi, Delhi, India. The work in this thesis is original and has not been submitted in parts or full to this or any other University/Institute for any diploma or degree.

Pushkar Gole
(Candidate)
Department of Computer Science
University of Delhi
Delhi, India

CERTIFICATE

This is to certify that the work presented in this thesis titled "**Lightweight and Few-Shot Image-based Plant Disease Diagnosis and Remedy Recommender System**" being submitted by **Pushkar Gole** in the Department of Computer Science, University of Delhi, Delhi, India, for the award of the degree of Doctor of Philosophy is a record of original research work carried out by him under the supervision of **Sr. Prof. Punam Bedi**. The work in this thesis has not been submitted in parts or full to this or any other University/Institute for any diploma or degree.

Sr. Prof. Punam Bedi
(Supervisor)
Senior Professor
Department of Computer Science
University of Delhi
Delhi, India

Prof. Vivek Kumar Singh
(Head of Department)
Professor
Department of Computer Science
University of Delhi
Delhi, India

Department of Computer Science
University of Delhi
Delhi, India

Date:

CERTIFICATE OF ORIGINALITY

The research work presented in this thesis titled "**Lightweight and Few-Shot Image-based Plant Disease Diagnosis and Remedy Recommender System**" has been carried out by me at the Department of Computer Science, University of Delhi, Delhi, India. The manuscript has been subjected to plagiarism check by Drilbit software. The work submitted for consideration of award of Ph.D. is original.

Pushkar Gole
(Candidate)
Department of Computer Science
University of Delhi
Delhi, India

STUDENT APPROVAL FORM

Name of the Author	Pushkar Gole
Department	Computer Science
Degree	Ph.D.
University	University of Delhi
Supervisor	Sr. Prof. Punam Bedi
Co-Supervisor	Not Applicable
Thesis Title	Lightweight and Few-Shot Image-based Plant Disease Diagnosis and Remedy Recommender System
Year of Submission	2024

Agreement

1. I hereby certify that, if appropriate, I have obtained and attached hereto a written permission/statement from the owner(s) of each third party copy righted matter to be included in my thesis/dissertation, allowing distribution as specified below.

2. I hereby grant to the university and its agents the non-exclusive license to archive and make accessible, under the conditions specified below, my thesis/dissertation, in whole or in part in all forms of media, now or hereafter known. I retain all other ownership rights to the copyright of the thesis/dissertation. I also retain the right to use in future works (such as articles or books) all or part of this thesis/dissertation or project report.

Conditions:

1. Release the entire work for access worldwide	√

2. Release the entire work for 'My University' only for 1 Year 2 Years 3 Years and after this time release the work for access worldwide	
3. Release the entire work for 'My University' only while at the same time releasing the following parts of the work (e.g. because other parts relate to publications) for worldwide access a) Bibliographic details and Synopsis only. b) Bibliographic details, Synopsis and the following chapters only. c) Preview/Table of Contents/24 pages only	
4. View Only (No Downloads) (worldwide)	

Signature of the Candidate

Pushkar Gole

Signature and seal of Supervisor

Sr. Prof. Punam Bedi

(Senior Professor)

Place:

Date:

Abstract

Agricultural sector is the backbone of human civilization, as it is the prominent source of nutrition for human beings. Additionally, it provides fodder to livestock which gives milk, wool, and other necessary products for humans. The demand for food is increasing day by day as the world's population has been increasing exponentially over the past few decades. In order to fulfill such huge food demand, the growth of farming sector becomes essential. Early-stage plant disease diagnosis is a big challenge in the growth of farming sector, as it can minimize crop yield loss and maximize the farmer's profit.

Conventionally, farmers and plant pathologists manually examine the plants to detect probable diseases, which is quite a difficult and laborious task. Due to the technological advancements in computer vision, various researchers have utilized different Machine Learning (ML) or Deep Learning (DL) techniques in the literature for diagnosing plant diseases with the help of their digital leaf images. However, most of these research works have utilized large number of trainable weight parameters and large number of annotated leaf images for training their models.

Therefore, the aim of this thesis is to develop a lightweight DL model that requires a smaller number of annotated leaf images for training, as annotating leaf images is a laborious and time-consuming task. The reason for choosing the DL model over the ML model lies in its ability to automatically extract important features from raw data, which eliminates the requirement of a separate feature extraction module. This thesis first proposes a lightweight DL model named PlantGhostNet for diagnosing a plant disease from leaf images.

PlantGhostNet model utilizes the Ghost and Squeeze-and-Excitation modules to reduce the trainable weight parameters and improve the model's performance, respectively. Ghost Module generates the feature maps in two phases. First, it generates few feature maps via conventional convolution operation. After that, it applies the cheap linear operations on the feature maps generated in the first phase to obtain the desired number of feature maps. Hence, the number of trainable weight parameters utilized in the Ghost Module are significantly lesser than the conventional convolution operation. Squeeze-

and-Excitation Module adaptively prioritizes each channel of the input feature map by assigning them weights, resulting in better performance of the model. The effectiveness of the PlantGhostNet model is evaluated on the leaf images of peach plants extracted from the PlantVillage dataset. Experimental results demonstrate that the PlantGhostNet model achieved 99.51% accuracy in detecting bacterial spot disease of peach plants, and it utilizes roughly seventy-three thousand trainable weight parameters. Although the PlantGhostNet model requires minimum trainable weight parameters as compared to other counterparts, they are still high in number. Therefore, the next work of the thesis focuses on reducing trainable weight parameters further by a significant factor.

Next, a lightweight hybrid model based on Convolutional Autoencoder (CAE), and Convolutional Neural Network (CNN) has been proposed in this thesis for detecting a plant disease from their leaf images. This model reduces the spatial size of input leaf images via CAE before classifying it with CNN, which results in significantly lesser number of training parameters. The proposed model uses only 9,914 trainable weight parameters and achieved 98.35% accuracy in detecting bacterial spot disease of peach plants. Though the PlantGhostNet model and lightweight hybrid DL model can identify a plant disease very efficiently and effectively, these models cannot diagnose multiple types of plant diseases with high accuracy. Therefore, in order to deal with this issue, a lightweight and improved Vision Transformer (ViT) model named TrIncNet has been proposed in this thesis.

TrIncNet model can identify multiple types of plant diseases through their leaf images. This model encompasses of multiple linearly connected Trans-Inception blocks, which have been designed by replacing the Multi-Layer-Perceptron Module with the Inception Module in the encoder block of the ViT model. As a result of this replacement, the Trans-Inception block requires 32.67% lesser number of trainable weight parameters than the original encoder block of ViT. Unlike the ViT model, the TrIncNet model also employs skip connections around each Trans-Inception block to make the model more resistant towards the vanishing gradient problem. In order to showcase the applicability of the TrIncNet model in detecting plant diseases, it has been trained and tested on PlantVillage and Maize datasets. Experimental results showed that the TrIncNet model

outperformed existing state-of-the-art research works by achieving 99.93% and 96.93% accuracies on PlantVillage and Maize datasets, respectively.

Disease severity is required in order to take necessary steps for curing the identified disease as it can provide a quantitative assessment of the damage caused by the pathogen of the identified disease. Hence, this thesis proposes a lightweight and few-shot framework named PDSE-Lite for diagnosing plant diseases and estimating the severity of identified diseases. This framework is designed and developed in two stages. In the first stage, a lightweight CAE model is developed and trained to reconstruct leaf images from original leaf images with minimal reconstruction loss. In subsequent stage, pretrained layers of the CAE model built in the first stage are utilized to develop the image classification and segmentation models, which are then trained using Few-Shot-Learning (FSL). Experimental results on the Apple-Tree-Leaf-Disease-Segmentation (ATLDS) dataset demonstrate that the PDSE-Lite framework can detect four types of apple leaf diseases with 98.35% accuracy and segment infected areas from diseased leaf images with 94.54% MeanIoU value by utilizing only two leaf images per class for model training. Therefore, the PDSE-Lite framework reduces the reliance on large manually annotated datasets and minimizes the human efforts required to create such datasets. This framework can also estimate the severity of the identified disease by calculating the percentage of diseased pixels out of the total leaf pixels, i.e., sum of healthy and diseased pixels present in the segmented leaf image.

Lastly, an android mobile application named "PlantD^2R^2S-Lite" has been developed in this thesis for generating the advisory for curing the plant disease by identifying the plant disease and its severity from the captured leaf image. The application works in both English and Hindi language. PlantD^2R^2S-Lite application utilizes the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model for generating advisory to cure the diagnosed plant disease. The pre-trained BERT model is fine-tuned on the text of two research papers that have detailed descriptions of apple leaf diseases and their management for testing its advisory generation functionality. The developed mobile application can work in remote locations also with weak or no Internet connectivity, as the disease detection and segmentation models, along with the BERT model have been embedded into the PlantD^2R^2S-Lite application. Experimental

results on the ATLDS dataset revealed that the developed application can diagnose apple leaf disease and segment the diseased lesions with high accuracy. Additionally, this mobile application requires the least space in the mobile device as compared to other applications developed in the literature. Therefore, it can also work on low-computational powered smartphones.

Hence, the PlantD^2R^2S-Lite application designed and developed in this thesis can help farmers in diagnosing plant diseases, highlighting diseased areas of leaf image, estimating the severity of identified diseases, and generating advisory for taking timely actions to cure the identified disease even in remote areas where Internet connectivity may not be strong.

Acknowledgment

The journey of my Ph.D. is both challenging and rewarding. During this journey, I have faced various challenges that have strengthened my patience and never-give-up spirit. I am deeply grateful for the support and guidance of every individual who helped me directly or indirectly in my life.

असतो मा सद्गमय ।

तमसो मा ज्योतिर्गमय ।

मृत्योर्मा अमृतं गमय ।

"Oh! Almighty God, take me to truth from untruth.

Take me to light from the darkness.

Take me to immortality from death."

With this spiritual couplet, I take a bow to pay my gratitude towards Almighty God, the source of all my wisdom and strength. I thank him for constantly providing his divine blessings (patience and courage) to me in both the good and tough times of my Ph.D. journey. May God's blessings continue to light the way for me and everyone seeking knowledge.

गुरुर्ब्रह्मा ग्रुरुर्विष्णुः गुरुर्देवो महेश्वरः।

गुरुः साक्षात् परं ब्रह्म तस्मै श्री गुरवे नमः॥

"The Guru is a virtue of Brahma, Vishnu, and Shiva. He is truly a representative of the Supreme Almighty God. Salutations to such splendorous and Holy Guru."

I would like to sincerely express my gratitude to my supervisor, Senior Professor Punam Bedi. I consider myself blessed and honored to be supervised by her. Her immense pool of knowledge, invaluable guidance, constructive suggestions, and constant support at each step of my Ph.D. journey played a pivotal role in shaping this doctoral thesis. Under her supervision, I learned to adjust priorities and make a balance of all the things in my life. I am also thankful to her for providing various opportunities of classroom teaching, which deeply nourished my communication skills. She is truly

an inspiring person and the epitome of knowledge. I have always idolized her, and my gratitude towards her cannot be expressed in finite words.

I further extend my gratitude to the current Head of the Department of Computer Science, University of Delhi, Prof. Vivek Kumar Singh, and all former Heads of the Department for providing timely administrative help whenever needed. I would also like to thank all office and laboratory staff of the Department of Computer Science, University of Delhi, for providing necessary help whenever required.

I am extremely grateful to Prof. Naveen Kumar and Prof. Sudeep Marwaha, my doctoral advisory committee members, for constantly providing their critical and constructive feedback to significantly enhance the quality of my research work. I want to extend my heartfelt thanks to Prof. Sudeep Marwaha for providing me access to high computational power resources along with the real-in-field plant disease detection dataset. This dataset and high-computational powered system helped me a lot in accomplishing my Ph.D. research work. I am highly obliged to Prof. Sudeep Marwaha, as his support played an integral part in the completion of this work.

I want to express my heartfelt thanks towards my parents for providing me their unwavering support and encouragement, along with unconditional affection at each step of my Ph.D. journey. Their love and belief in my abilities motivated me to stay strong in tough times. I extend my deep appreciation to my best friends Ayush Malik, Eva Chaudhary, Neelam Gangwar, and Nehal Pandey for cheering me on my success and standing with me in difficult times.

I would also like to pay my profound gratitude to the University Grants Commission (UGC), India, for providing me the financial assistance via Junior Research Fellowship (JRF) with reference number 3607/(NET-JULY2018).

"With deep admiration, I dedicate this work to the Almighty God, my late grandfather, my parents, and my supervisor, Senior Professor Punam Bedi."

Pushkar Gole

Table of Contents

List of Figures

List of Tables

List of Abbreviations

AdaGrad	Adaptive Gradient
API	Application Programming Interface
ATLDS	Apple Tree Leaf Disease Segmentation
AWS	Amazon Web Services
BERT	Bidirectional Encoder Representations from Transformer
BLSB	Banded Leaf and Sheath Blight
CAE	Convolutional Auto Encoder
CBAM	Convolutional Block Attention Module
CNN	Convolutional Neural Network
DCDM	Deeplens Classification and Detection Model
DIP	Digital Image Processing
DL	Deep Learning
DNN	Deep Neural Network
FSL	Few Shot Learning
GA	Genetic Algorithm
GAN	Generative Adversarial Network
GDP	Gross Domestic Product
GELU	Gaussian Error Linear Units
GLCM	Gray Level Co-occurrence Matrix
GPUs	Graphic Processing Units
IDE	Integrated Development Environment
KNN	K-Nearest Neighbor
LIME	Local Interpretable Model-Agnostic Explanations
LR	Logistic Regression
MAML	Model Agnostic Meta-Learning
Meta-Adam	Meta-learned Adaptive Moments
MHA	Multi-Head Attention
ML	Machine Learning
MLB	Maydis Leaf Blight

MLP	Multi-Layer Perceptron
MSE	Mean Squared Error
NLP	Natural Language Processing
NRMSE	Normalized Root Mean Square Error
P2OP	Plant Pathology on Palms
PDSE-Lite	Lightweight Plant Disease Severity Estimation
PlantD^{2}R^{2}S-Lite	Lightweight Plant Disease Detection and Remedy Recommender System
ReLU	Rectified Linear Unit
RF	Random Forest
RNN	Recurrent Neural Network
RPFE	Residual Progressive Feature Extraction
SECNN	Squeeze-and-Excitation CNN
SGD	Stochastic Gradient Descent
SRGAN	Super Resolution Generative Adversarial Network
SVM	Support Vector Machine
TLB	Turcicum Leaf Blight
TrIncNet	Trans-Inception Network
VAE	Variational Auto Encoder
ViT	Vision Transformer
WGAN	Wasserstein Generative Adversarial Network

1. Introduction

Agriculture played a pivotal role in the development of human civilization. Majority of the world's population is directly or indirectly dependent on the farming sector to fulfill their food requirements. Moreover, the agricultural sector significantly contributes to the economic growth of many agrarian countries. In India, this sector contributes around 18.3% of the country's Gross Domestic Product (GDP), and half of the country's workforce is associated with agriculture and its allied fields (Tomar, 2023). The world's population has exponentially increased in the last few decades. Consequentially, the food demand is also proliferating day by day, and in order to fulfill such colossal food demand, the growth of agricultural sector is essential.

Disease infestation in crops is one of the prominent challenges in the growth of farming sector, as it hampers both food grain quality and quantity. Diagnosing plant diseases in their earliest possible stages can significantly conquer this challenge because it has the potential to minimize crop yield loss and maximize the farmer's profit, too. Conventionally, farmers or agricultural scientists manually examine the plant leaves to identify the probable type of disease. Thereafter, they evaluate the disease severity by utilizing their domain expertise and suggest the necessary actions to cure the plant disease.

Due to technological advancements in the computer vision domain, various researchers have utilized different Machine Learning (ML) or Deep Learning (DL) techniques in the literature for diagnosing plant disease with the help of their digital leaf images. However, most of these research works have utilized DL models having large number of trainable weight parameters, which can significantly increase the model's training as well as inference time. While there are numerous research works available on disease detection, the research works on plant disease severity estimation are relatively limited. Furthermore, disease severity is required in order to help farmers in prioritizing the actions to be taken based on the damage assessment.

Research works focused on plant disease severity estimation require large number of annotated leaf images to train their models, and annotating large number of leaf images

is laborious and time-consuming. Additionally, to the best of our knowledge, none of the existing research work can generate bilingual advisory to cure plant diseases by considering disease type and severity both. Therefore, this motivated us to develop an effective and efficient plant disease diagnosis and remedy recommender system that conquers all of the aforementioned challenges. The key contributions of this thesis have been summarized in the subsequent section.

1.1. Key Contributions

The major contributions of this thesis are listed below:

1. *Two lightweight plant disease detection models have been proposed in the thesis for identifying a plant disease from leaf images.*
    a. *The first model named PlantGhostNet utilizes Ghost and Squeeze-and-Excitation modules for detecting single type of plant disease.* The Ghost Module is used for trainable weight parameter reduction, and the Squeeze-and-Excitation Module is utilized for performance improvement.
    b. *In the second work the Convolutional Auto Encoder (CAE) and Convolutional Neural Network (CNN) have been combined to build a lightweight hybrid model.* In this model, the CAE has been utilized to reduce the trainable weight parameters further.

    The aforementioned models can identify a plant disease very efficiently and effectively. However, these models cannot diagnose multiple plant diseases with high accuracy. Therefore, in the subsequent work, the TrIncNet model has been proposed to deal with this issue.
2. *A lightweight and improved Vision Transformer (ViT) network named Trans-Inception Network (TrIncNet) model has been proposed next in this thesis to identify multiple plant diseases from leaf images.* This model comprises of multiple linearly concatenated Trans-Inception blocks. Each of these Trans-Inception blocks consists of Inception Module in place of Multi Layer Perceptron (MLP) Module. This replacement results in lesser number of trainable weight parameters. Moreover, skip connection around each Trans-Inception block has also been added in the proposed TrIncNet model to make it

more resistant to the vanishing gradient problem. Though the TrIncNet model can effectively and efficiently detect multiple plant diseases from leaf images, but estimation of disease severity is also crucial for taking timely action for curing the identified disease. Hence, in the next work, PDSE-Lite framework has been developed to handle this problem.

3. *A lightweight PDSE-Lite framework has been proposed in the thesis to estimate plant disease severity.* This framework has been designed and developed by using CAE and FSL. Since the PDSE-Lite framework utilizes FSL, it requires only few annotated leaf images for training. Hence, in this way, the PDSE-Lite framework reduces the dependence on large scale manually annotated datasets, and thereby minimizing the human efforts required to create such datasets. Although this framework can effectively and efficiently estimate plant disease severity, but a recommender system is needed to assist farmers by providing bilingual advisory to cure the identified plant disease. Hence, a recommender system named PlantD^2R^2S-Lite has been designed and developed next in the thesis to deal with this drawback.
4. *A recommender system named PlantD^2R^2S-Lite has been designed for providing various preventive and management practices to mitigate the effects of different diseases on plants.* This framework has been developed as an Android mobile application that can identify plant diseases and estimate their severity with the help of leaf images. This application also generates advisories for the farmers to cure the identified plant disease via pre-trained Bidirectional Encoder Representations from Transformer (BERT) model. The PlantD^2R^2S-Lite application works in both English and Hindi language to help non English speaking farmers.

The following section describes the chapter-wise organization of this thesis.

1.2. Organization of the Thesis

This thesis comprises of seven chapters, and each of these chapters has been summarized as follows:

- **Chapter 1** presents the significance of automatic plant disease detection and severity estimation for the growth of agricultural sector. Furthermore, it identifies several prominent research challenges in developing an effective and efficient recommender system for automatic plant disease diagnosis and providing advisory to cure the identified disease. This chapter also lists the key contributions of this research work along with the list of various journal and conference publications that are produced as the outcomes of this thesis.
- **Chapter 2** discusses the problem of image-based plant disease detection and severity estimation. It also describes various DL architectures, namely Deep Neural Network (DNN), CNN, CAE, and Vision Transformer (ViT). These DL architectures are used in subsequent chapters of this thesis. This chapter also describes various Few-Shot Learning (FSL) techniques based on distance metric learning, hallucination, and initialization. These techniques are utilized in chapter 5 of the thesis for designing the PDSE-Lite framework.
- **Chapter 3** evaluates the performances of various predefined CNN architectures in detecting single type of plant diseases from leaf images. This chapter also proposes the PlantGhostNet model and a lightweight hybrid model for identifying a plant disease from leaf images. PlantGhostNet model is designed by combining Ghost and Squeeze-and-Excitation modules which reduces the trainable weight parameters and enhances the model's performance, respectively. On the other hand, the lightweight hybrid model is based on CAE and CNN. This model first obtains the compressed domain representations of leaf images using the encoder network of CAE and then uses these compressed domain representations for classification via CNN. Due to the reduction of spatial dimensions using CAE, the number of features and, hence, the number of trainable weight parameters of the lightweight hybrid model reduced significantly as compared to the PlantGhostNet model and existing state-of-the-art models.
- **Chapter 4** presents a lightweight and improved ViT network named Trans-Inception Network (TrIncNet) for identifying multiple plant diseases from leaf images. The TrIncNet model encompasses of multiple modified encoder blocks, i.e., Trans-Inception blocks. Each Trans-Inception block comprises of Inception

Module in place of MLP Module for extracting various temporal and spatial features from leaf images. As a result of this replacement, the Trans-Inception block requires significantly lesser number of trainable weight parameters than the original encoder of ViT. Additionally, skip connections are added between each Trans-Inception block to make the proposed network more resistant towards the vanishing gradient problem. In this way, the TrIncNet model can effectively and efficiently identify multiple plant diseases from leaf images as compared to other counterparts.

- **Chapter 5** proposes a lightweight Plant Disease Severity Estimation (PDSE-Lite) framework to estimate the severity of identified diseases effectively and efficiently. This framework is designed and developed by utilizing CAE and FSL. As this framework is based on FSL, it requires only few annotated leaf images for training. In this way, the PDSE-Lite framework reduces the reliance on large scale manually annotated datasets, and thereby minimizing the human efforts required to create such datasets.
- **Chapter 6** presents a novel PlantD^2R^2S-Lite framework for generating bilingual advisory to cure plant diseases. The PlantD^2R^2S-Lite utilizes the PDSE-Lite framework for plant disease detection and severity estimation through digital leaf images of plants. Thereafter, it generates the advisory for the farmers to remediate the identified disease with the help of pretrained BERT model. This framework has been developed as an Android mobile application that can be used in either English or Hindi language. The models for plant disease diagnosis and remedy recommendation have been embedded into the application. Therefore, it can work even in remote locations where Internet connectivity may not be strong.
- **Chapter 7** summarizes all contributions of this research work and concludes the thesis. Moreover, it discusses several limitations of the research work presented in the thesis, along with directions for future research.

1.3. List of Publications

This section presents my co-authored papers that were published or communicated during my Ph.D. tenure.

Journal Publications:

1. Punam Bedi, Pushkar Gole, and Sudeep Marwaha (2024), "PDSE-Lite: Lightweight Framework for Plant Disease Severity Estimation based on Convolutional Autoencoder and Few-Shot Learning", *Frontiers in Plant Science*, 14, pp. 1319894, Frontiers Media S.A, DOI: 10.3389/fpls.2023.1319894, ISSN: 1664-462X. [SCIE, Impact Factor: 4.1]
2. Pushkar Gole, Punam Bedi, Sudeep Marwaha, Md. Ashraful Haque, and Chandan Kumar Deb (2023), "TrIncNet: a lightweight vision transformer network for identification of plant diseases", *Frontiers in Plant Science*, 14, pp. 1221557, Frontiers Media S.A, DOI: 10.3389/fpls.2023.1221557, ISSN: 1664-462X. [SCIE, Impact Factor: 4.1]
3. Punam Bedi and Pushkar Gole (2021), "Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network", *Artificial Intelligence in Agriculture*, 5, pp. 90-101, Elsevier, DOI: 10.1016/j.aiia.2021.05.002, ISSN: 2589-7217. [ESCI, Impact Factor: 8.2]

Conference Publications:

4. Pushkar Gole, Punam Bedi, and Sudeep Marwaha (2023), "Automatic Diagnosis of Plant Diseases via Triple Attention Embedded Vision Transformer Model" *In Hassanien, A.E., Castillo, O., Anand, S., Jaiswal, A. (Eds.), International Conference on Innovative Computing and Communications (ICICC-2023)*, 17-18 February 2023, pp: 879-889, Delhi, India, Springer, Singapore, DOI: 10.1007/978-981-99-4071-4_67. [Scopus Indexed]
5. Punam Bedi and Pushkar Gole (2021), "PlantGhostNet: An Efficient Novel Convolutional Neural Network Model to Identify Plant Diseases Automatically" *In 2021 9th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, 03-04 September 2021, pp. 1-6, Noida, India, IEEE, DOI: 10.1109/ICRITO51393.2021.9596543. [Scopus Indexed]

Book Chapter:

6. Punam Bedi, Pushkar Gole, and Sumit Kumar Agarwal (2021), "18 Using Deep Learning for image-based plant disease detection", Internet of Things and Machine Learning in Agriculture, pp. 369-402, De Gruyter, DOI: 10.1515/9783110691276-018, Online ISBN: 978-3-11-069122-1.

Research Paper Communicated to Journal:

1. Punam Bedi, Pushkar Gole, Sudeep Marwaha. "PlantD^2R^2S-Lite: Lightweight and Bilingual Plant Disease Diagnosis and Remedy Recommender System"

2. Background Concepts

This chapter briefly describes the problem of image-based plant disease detection and severity estimation. Further, this chapter explains the basic concepts of different DL and FSL techniques used in later chapters of this thesis. At the end this chapter concludes with a summary.

2.1 Image-based Plant Disease Detection and Severity Estimation

Disease infestation in the growing stages of crops can adversely affect their quality and quantity, which will later adversely affect the food supply chain. Diagnosing plant diseases at their earliest possible stages can be a viable strategy to conquer this, as it has the potential to minimize crop loss and maximize the profit of farmers. Traditionally, farmers and agricultural scientists diagnose plant diseases by manually examining the plant leaves for the occurrence of disease lesions. Then, they utilize their domain knowledge for disease severity estimation. This process of identifying plant diseases and estimating the severity of identified diseases is laborious and time-consuming, as it requires human intervention. Nowadays, plant disease diagnosis and severity estimation are performed by applying Image processing, ML, or DL algorithms on digital leaf images of plants. Subsequent sections of this chapter explain the basic concepts of different DL and FSL techniques that are utilized in this thesis.

2.2 Deep Learning Architectures

DL is a subfield of Artificial Intelligence inspired by the functioning of the human brain. The prominent advantage of DL techniques over ML techniques is that these techniques can automatically extract various features from raw data. Hence, there is no need for an extra feature extraction module. This section is further divided into four subsections in which various DL architectures, namely Deep Neural Network (DNN), Convolutional Neural Network (CNN), Convolutional Auto Encoder (CAE), and Vision Transformer (ViT), are explained.

2.2.1 Deep Neural Network (DNN)

DNN is a classical fully connected neural network in which neurons of adjacent layers are interconnected to each other. Each layer of this network has several neurons (also known as nodes) that utilize a non-linear activation function (like Sigmoid, TanH, ReLU, etc) for extracting various complex hidden features present in the input data. The first and last layers of DNN are known as input and output layers, respectively. The intermediate layers between the input and output layers are known as hidden layers. A six-layer DNN has been shown in Figure 2.1.

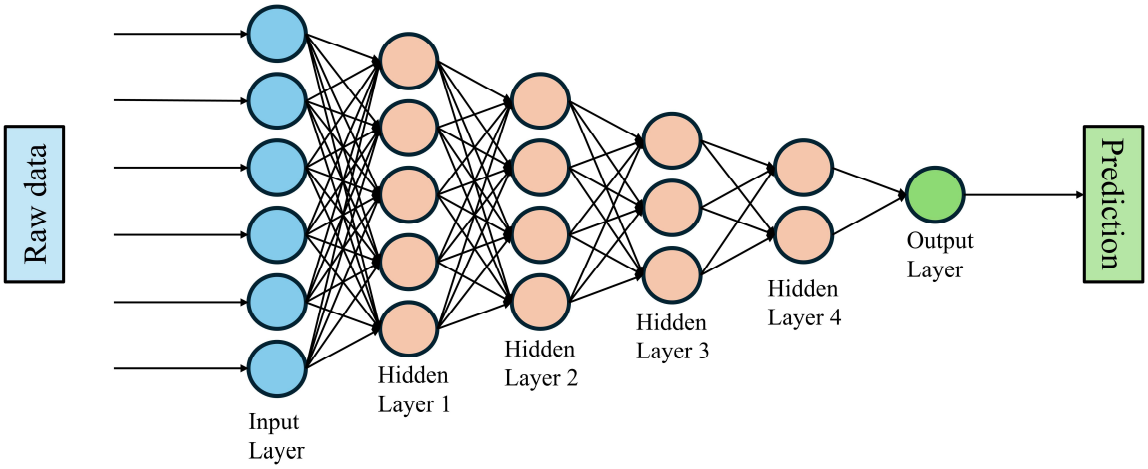


Figure 2.1: Block diagram of a six-layer DNN

In Figure 2.1, the circles represent neurons of the DNN, and edges connecting the neurons carry the weights, which are adjusted during the training process. Therefore, these weights are known as trainable weight parameters. In DNN, if i^{th} layer has N^i neurons and $(i+1)^{th}$ layer comprises of N^{i+1} neurons, then the number of trainable weight parameters between i^{th} and $(i+1)^{th}$ layer would be $N^i \times N^{i+1} + N^{i+1}$. Furthermore, the total number of trainable weight parameters of a DNN can be obtained by summing up the trainable weight parameters of all the layers present in the DNN.

The major drawback of DNN is that all adjacent layers of this network are fully connected to each other. Therefore, it requires large number of trainable weight parameters, which makes the DNN model computationally heavy. Moreover, this model cannot extract different spatial and temporal features from images efficiently, which can later enhance the performance of model in image classification. These

drawbacks of the DNN model have been conquered by the CNN model, which is described in the following subsection.

2.2.2 Convolutional Neural Network (CNN)

Convolutional Neural Network (CNN) is a type of DL architecture inspired by the human visual system. This network employs convolution operation instead of simple matrix multiplication in at least one of its layers (Lecun, 1989). Due to its excellent capability in extracting various spatial and temporal features from image data, it is widely used in different image processing tasks like image classification, image segmentation, object detection, etc. As this thesis focuses on plant disease detection and severity estimation using leaf images, CNN has been utilized in the research work. Hence, in order to describe CNN, its different building blocks are explained next in this section, and the block diagram of a typical CNN is given in Figure 2.2.

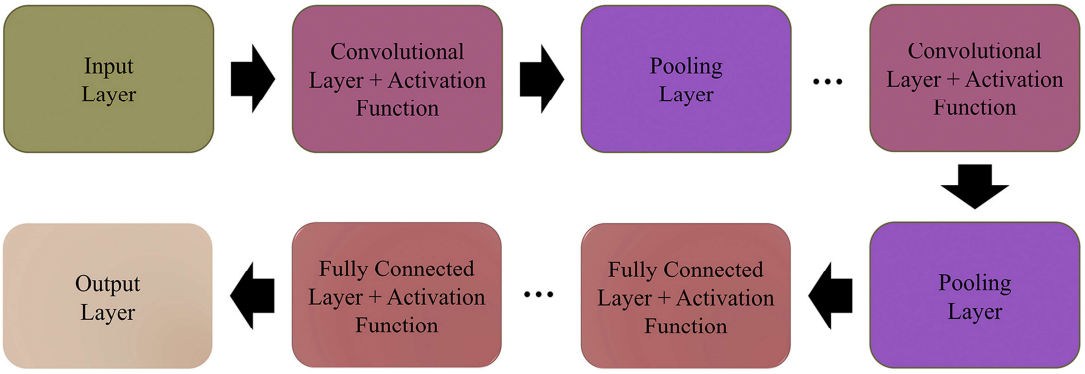


Figure 2.2: Block diagram of a typical CNN

A. Convolutional Layer

The convolutional layer of CNN performs the convolution operation on input images to extract different spatial and temporal features. A typical CNN can have more than one convolutional layer. The initial convolutional layer captures low-level features like edges, the orientation of gradients, color, etc. Convolutional layers that are placed at the end of a CNN extract high-level features that help the network to understand the input images. These extracted features are utilized in various computer vision tasks like image classification, image segmentation, object detection, etc. Mathematically, the convolution operation can be defined on two real-valued functions, say a and b. Its

output is also a function, say c, which expresses the effect of function a on b. The mathematical formula for convolution operation in continuous and discrete domain is given in equation 2.1 and equation 2.2, respectively.

$$c(x) = (a * b)(x) = \int_{-\infty}^{\infty} a(x) \cdot b(x - k) \cdot dk \tag{2.1}$$

$$c(x) = (a * b)(x) = \sum_{k=-\infty}^{\infty} a(x) \cdot b(x - k) \tag{2.2}$$

In CNN, $a(x)$, $b(x)$, and $c(x)$ are termed as input feature map, filter/kernel, and output feature map, respectively. The visualization of the convolution operation has been given in Figure 2.3. In the convolutional layer, multiple filters can be utilized, and these filters move row-wise along the image to get the output feature maps. The number of output feature maps is equal to the number of filters utilized in the convolutional layer.

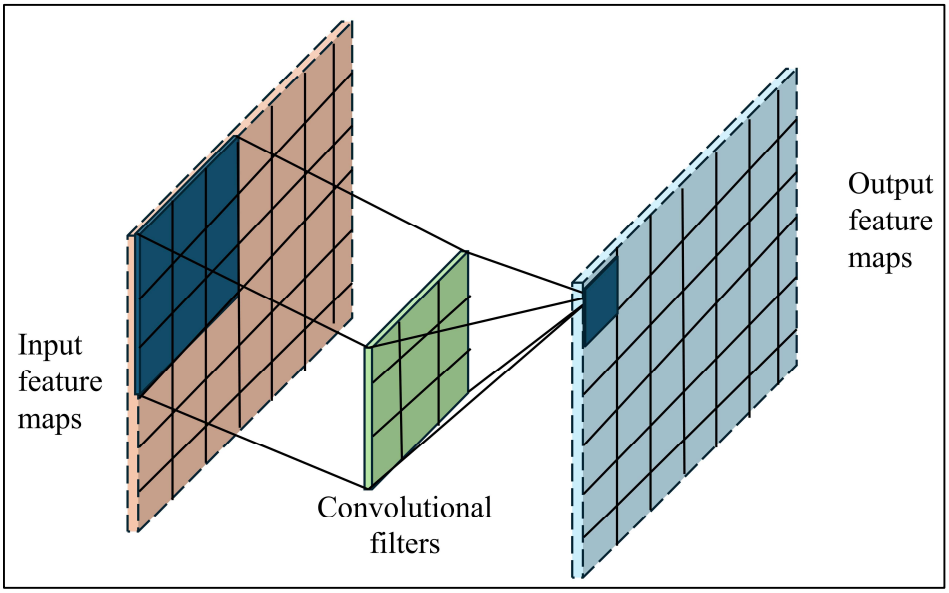


Figure 2.3: Visualization of convolution operation

It can be observed from the above figure that unlike DNN, the adjacent convolutional layers of CNN are not fully connected to each other. Instead of, they are connected by single or multiple kernel(s)/filter(s), which carry the weight parameters and move along the input feature map to extract different spatial and temporal features from input images. Hence, the number of edges connecting two adjacent convolutional layers of

CNN are significantly lesser than the edges present in DNN. Since the number of trainable weight parameters are based on the number of edges and hence the number of trainable weight parameters in convolutional layers are significantly lesser than the layers of DNN. If the dimension of the input feature map is $I_a \times I_a \times I_c$ (where I_a is the height and width of the input feature map and I_c represents the number of channels present in I), and there are n convolutional filters of size $I_b \times I_b$ are applied, then the number of trainable weight parameters used in this convolutional layer are $I_c \times I_b \times I_b \times n + n$.

It can be observed from the definition of convolution operation that if the size of the input matrix is $I_a \times I_a$ and the size of the filter is $I_b \times I_b$ (where $I_b \leq I_a$), then the size of the output feature map is $(I_a - I_b + 1) \times (I_a - I_b + 1)$. Hence, after each convolution operation, size of the output feature map is decreased, or in other words, the size of the input feature map reduces after each convolution operation and becomes zero after some convolutions. Thereby, it limits the CNN's depth by placing an upper bound on the number of convolutional layers present in a CNN. Furthermore, the elements present on the edges and corners are utilized less than the elements present in the center of the input matrix. In order to deal with these two issues, padding is used in the convolutional layers.

Padding is the process of appending zeroes to the borders of the input matrix. This increases the area of the input matrix on which the convolution operation has to be performed, and it ensures that the spatial dimensions of the input image do not reduce after performing a convolution operation. Moreover, performing padding multiple times ensures that the elements present on the edges and corners are utilized equally as of the elements present in the center of the input matrix. Figure 2.4 depicts padding on a 5×5 input matrix. Let us suppose p layers of zeroes are appended in an input matrix of size $m \times m$. Further, this matrix is convolved with a kernel of size $k \times k$. Then, the value of p for keeping the dimension of the input matrix same after the convolution operation can be calculated by equation 2.3.

$$m + 2p - k + 1 = m \Longrightarrow p = \frac{k - 1}{2} \tag{2.3}$$

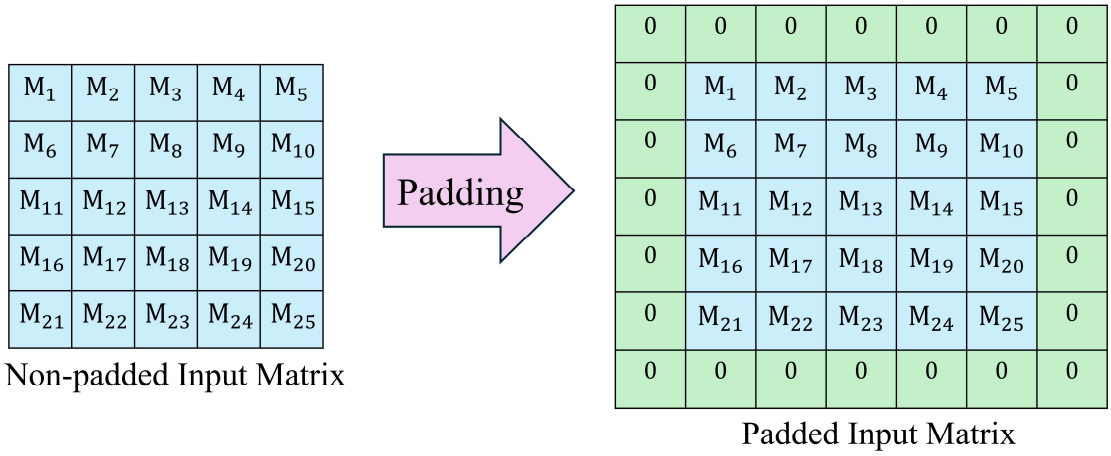


Figure 2.4: Padding operation on an input matrix of size 5×5

From equations 2.1 and 2.2, it can be observed that convolution is a linear operation. Thus, in order to extract the non-linear features from images, different non-linear activation functions, such as Sigmoid, Hyperbolic Tangent (TanH), ReLU, etc., are employed in convolutional layers.

B. Pooling Layer

Pooling layer is also an integral part of CNN, and it is employed in CNN for reducing the dimensions of the input feature map by selecting one value within a neighborhood based on different statistics. Some widely utilized statistics are average, L_2-norm, weighted average, and max-pooling. Max-pooling and average-pooling pick the maximum value and average value from its neighborhood, respectively. The pictorial representation of max-pooling and average-pooling is shown in Figure 2.5.

It can be seen from Figure 2.5 that the pooling layer can reduce the dimensions of the input feature map. Consequently, it reduces the number of trainable weight parameters, which results in less training and inference time.

C. Fully Connected Layer (Dense Layer)

The fully connected layer is a layer of DNN (also known as the dense layer), which comprises of multiple neurons with an activation function. A typical CNN encompasses of a set of fully connected layers, which utilizes the features extracted from previous convolutional and pooling layers for classifying the input image into their respective classes. The input to fully connected layers must be a one-dimensional vector, and the

output (feature maps) obtained from previous convolutional and pooling layers is in the form of multidimensional arrays, also known as tensors. Therefore, these output feature maps are flattened to one-dimensional vector. The last fully connected layer of a CNN acts as an output layer, which comprises of as many neurons as the number of classes available in the dataset. This layer gives the probability of belongingness of the input image to different classes available in the dataset. Next section of this chapter describes the CAE model.

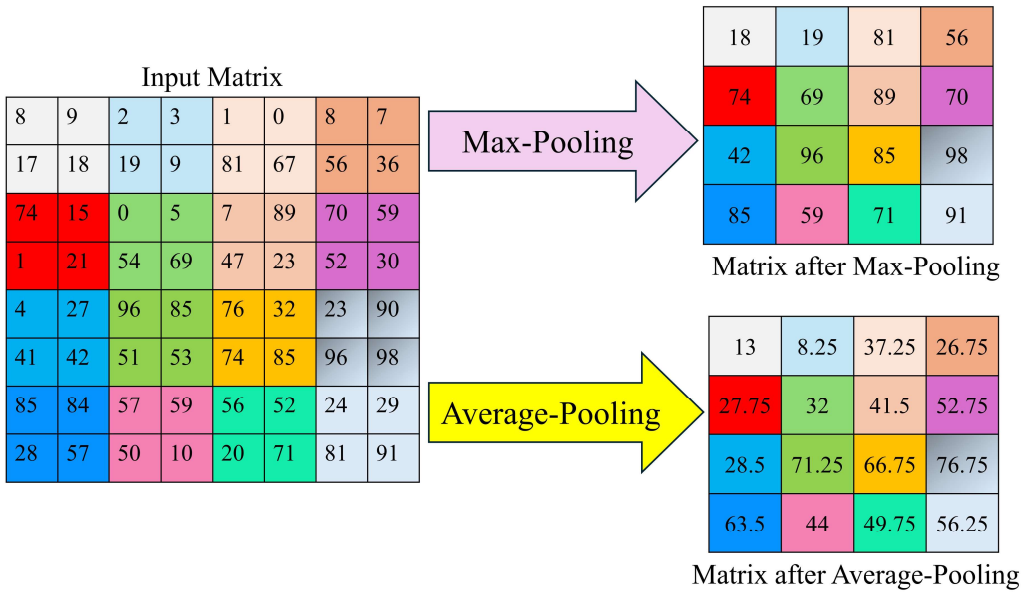


Figure 2.5: Max-pooling and Average pooling operations within 2×2 neighborhood

2.2.3 Convolutional Auto Encoder (CAE)

Convolutional Auto Encoder (CAE) is a type of Auto Encoder that utilizes a network of convolutional and pooling layers to encode the input images into some lower-dimensional space or compressed domain space. In order to do this, a bottleneck is introduced in the network, which compels the network to learn the compressed domain representation of the input images. The block diagram of a typical CAE having N layers is shown in Figure 2.6.

The CAE comprises of four components: Encoder Network, Bottleneck Layer, Decoder Network, and Reconstruction Loss. Encoder Network consists of a set of convolutional and downsampling layers (max-pooling layers). These layers extract various spatial and

temporal features from image data and then encode the extracted features into compressed domain representation. This compressed domain representation is stored in the Bottleneck Layer of CAE, and it encompasses of all essential features of images, which are further required by the Decoder network to reconstruct the images.

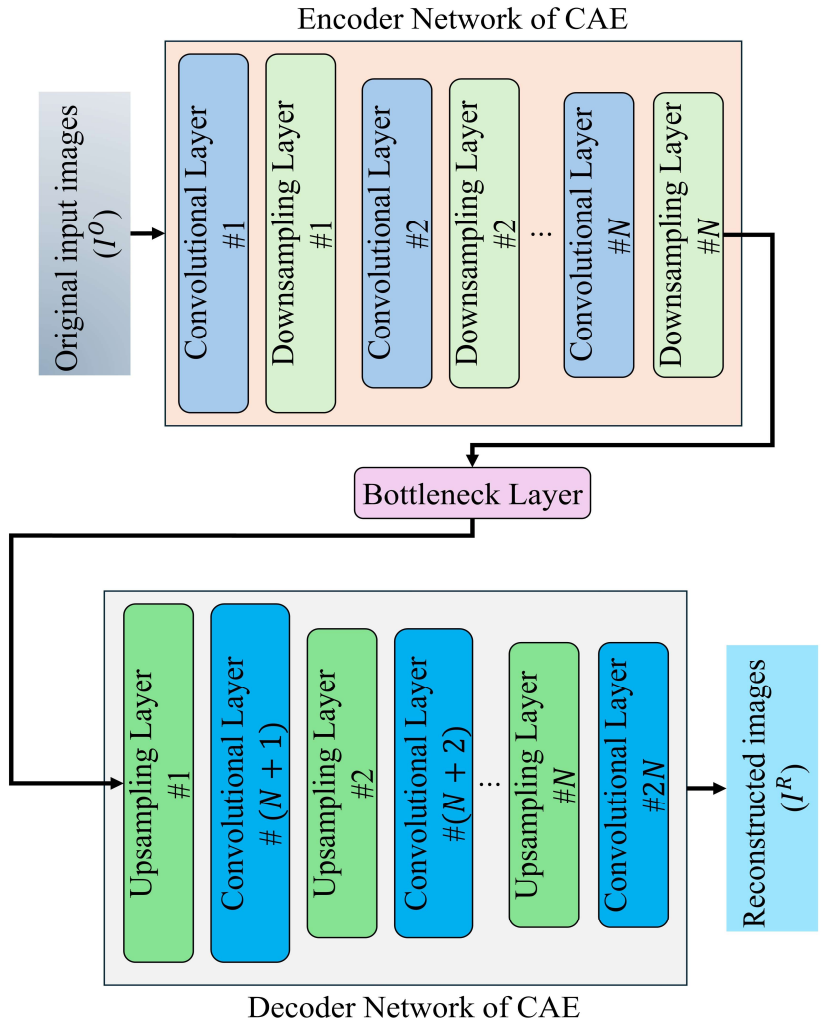


Figure 2.6: Block diagram of a typical CAE

Decoder Network of CAE consists of same number of layers as the Encoder Network but in reverse order. Moreover, in Decoder Network, upsampling layers are utilized instead of downsampling layers. The Decoder Network tries to reconstruct input images with minimum reconstruction loss. The reconstruction loss is a function that evaluates the difference between the original input images (say I^O) and reconstructed images (say I^R). In literature, majorly the Mean Squared Error (MSE) function is utilized to measure

the reconstruction loss. The mathematical formula for MSE is given in equation 2.4, where N_I denotes the number of images taken into consideration.

$$MSE(I^O, I^R) = \frac{1}{N_I}\sum_{j=1}^{N_I}(I_j^O - I_j^R) \tag{2.4}$$

In literature, there are different types of Auto Encoders, such as Stacked Auto Encoder, Denoising Auto Encoder, etc. However, the work presented in this thesis deals with plant leaf images. Therefore, CAE has been used in this research work. In the next section of this chapter, the ViT model has been described.

2.2.4 Vision Transformer (ViT)

The Vision Transformer (ViT) model is a Transformer (Vaswani, et al., 2017) based DL model designed and developed by (Dosovitskiy, et al., 2021) to perform various computer vision tasks such as image classification, image segmentation, etc. The block diagram of a typical ViT model is shown in Figure 2.7.

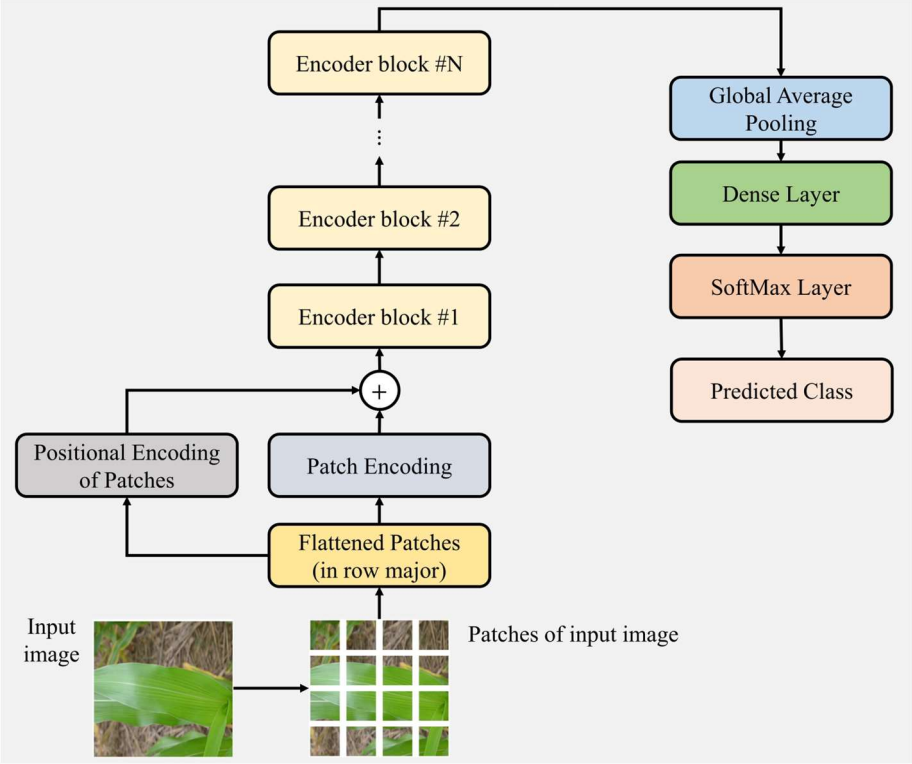


Figure 2.7: Block diagram of a typical ViT Model

In the ViT model, the input images of size $M \times M \times C$, are first divided into small patches of size $p \times p \times C$. Here, M denotes the height and width of input images, and p represents the height and width of patches. Moreover, C is the number of channels present in the input images. After splitting the input images into small patches, these patches are flattened and then transformed by a linear embedding, $E: \mathbb{R}^{p \times p \times C} \rightarrow \mathbb{R}^{d}$, also known as patch embedding. Here, d represents the embedding dimension of the ViT model. On the other hand, the positional embedding of the patches are computed by finding the index of a patch in row-major order. After that, the patch embedding and positional embedding of the patches are added together to form the input for the first encoder block of the ViT model. The ViT model comprises of multiple stacked encoder blocks, and each encoder block encompasses of three modules: Layer Normalization, Multi-Head Attention (MHA), and Multi Layer Perceptron (MLP). These modules are described in the following subsections, and the architectural diagram of the ViT model's encoder block is shown in Figure 2.8.

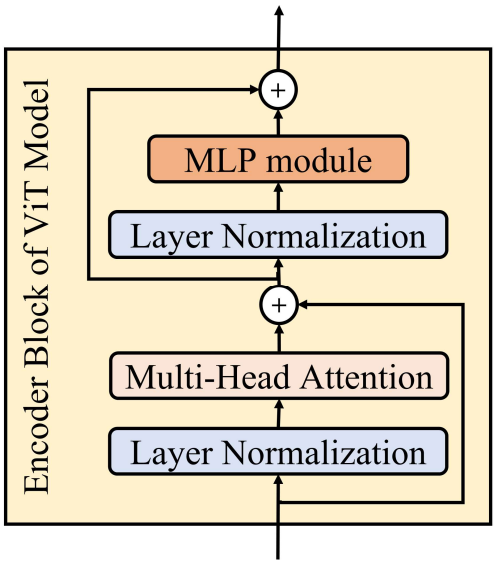


Figure 2.8: Architectural diagram of ViT model's encoder block

A. Layer Normalization Module

Layer Normalization has been proposed by (Ba, Kiros, & Hinton, 2016) to conquer the drawbacks of Batch Normalization, as its dependency on batch statistics can lead to instability with small batch sizes. On the other hand, Layer Normalization normalizes the activations in the feature direction instead of the batch direction. Therefore, it

overcomes the aforementioned shortcoming of Batch Normalization by removing the dependence on batches. Moreover, it normalizes every feature of the activations to unit variance and zero mean. In the Layer Normalization paradigm, first, means and variances are calculated for each channel of the feature map through equations 2.5 and 2.6, respectively. Second, the normalized feature maps are computed by equation 2.7, and at last, scaling and shifting are done with the help of two learnable parameters, i.e., γ and β, by equation 2.8.

$$\mu_{i,c} = \frac{1}{HW}\sum_{h=1}^{H}\sum_{w=1}^{W} x_{ihwc} \tag{2.5}$$

$$\sigma_{i,c}^2 = \sum_{h=1}^{H}\sum_{w=1}^{W}\left(x_{ihwc} - \mu_{i,c}\right)^2 \tag{2.6}$$

$$x_{i,hwc}^{norm} = \frac{x_{ihwc} - \mu_{i,c}}{\sqrt{\sigma_{i,c}^2 + \epsilon}} \tag{2.7}$$

$$y_i = \gamma\, x_i^{norm} + \beta \equiv LN_{\gamma,\beta}(x_i) \tag{2.8}$$

where m is the number of the feature maps, $1 \leq i \leq m$, H, W, and C are the height, weight, and channels of the feature map, respectively, and $1 \leq c \leq C$. In the ViT model, the Layer Normalization module normalizes the feature map received from its previous layer to improve the model's stability and performance.

B. Multi-Head Attention (MHA) Module

In the Multi-Head Attention (MHA) module, m self-attention operations are performed parallelly, where m is a hyperparameter representing the number of heads used in this module. The self-attention operation is inspired by the operation of a human eye that focuses only on the part of information while ignoring other things (Bahdanau, Cho, & Bengio, 2015). The aim of this operation is to gather the relationships among all entities. These entities can be words of a sentence in the Natural Language Processing (NLP) domain, while in the computer vision domain, these entities are the patches of an image. Let there be k such entities, i.e., $(e_1, e_2, e_3, \ldots, e_k)$ represented by $E \in \mathbb{R}^{k\times d}$, where d

is the embedding dimension in which the information of each entity has been embedded. In self-attention operation, three learnable weight matrices: Query ($W_Q \in \mathbb{R}^{d \times d_q}$), Key ($W_K \in \mathbb{R}^{d \times d_k}$), and Value ($W_V \in \mathbb{R}^{d \times d_v}$) are trained using the backpropagation algorithm, where $d_q, d_k,$ and d_v are the number of columns present in Query, Key, and Value weight matrices. In self-attention operation, first, the input sequence E is multiplied with these learnable matrices to get $Q = EW_Q, K = EW_K$, and $V = EW_V$ matrices. After obtaining the Q, K, and V matrices, the self-attention score (Z) matrix is calculated by equation 2.9 (Vaswani, et al., 2017).

$$Z = softmax\left(\frac{QK^t}{\sqrt{d_k}}\right) \cdot V \tag{2.9}$$

The outputs of all m heads are concatenated together and then multiplied by an output weight matrix ($W_O \in \mathbb{R}^{k \times md_v}$) according to equation 2.10, where Z_i is the self-attention score matrix of i^{th} head.

$$Z_{multihead} = concat(Z_1, Z_2, \dots, Z_i, \dots, Z_m)^t W_O \tag{2.10}$$

In the ViT model, the MHA module captures global dependencies between image patches by simultaneously performing multiple self-attention operations.

C. Multi Layer Perceptron (MLP) Module

The Multi Layer Perceptron (MLP) module present in the encoder block of the ViT model consists of two fully connected (dense) layers. The first and second layers of the MLP module have $2d$ and d number of neurons, respectively. Moreover, these layers employ Gaussian Error Linear Units (GELU) non-linear activation function to extract various non-linear features from image patches. The approximated GELU activation function (σ_{GELU}) and mathematical representation of the MLP module is given in equations 2.11 and 2.12, respectively.

$$\sigma_{GELU}(x) = \frac{x}{2} \cdot \left[1 + tanh\left\{\sqrt{\frac{2}{\pi}} \cdot (x + 0.044715x^3)\right\}\right] \tag{2.11}$$

$$\text{MLP}(X) = \sigma_{GELU}(W_2^T(\sigma_{GELU}(W_1^T X + b_1) + b_2) \quad (2.12)$$

where X is the d-dimensional input vector, W_1, b_1 and W_2, b_2 are the weights and biases of the first and second layers of the MLP module, respectively. In the ViT model, the MLP module extracts intricate correlations between the image patches, which further help the model to correctly classify the input images into their corresponding classes. The work presented in this thesis also utilizes concepts of Few-Shot Learning (FSL), which is described in the next section.

2.3 Few-Shot Learning (FSL)

Any ML or DL model necessitates a large number of annotated instances during training for better generalization. However, the human brain can learn anything effectively by utilizing only few samples (Weng Lilian, 2018). For example, a person who knows how to ride a bicycle can quickly learn to ride a motorcycle with few or even no demonstration. Therefore, researchers tried to explore various learning algorithms that have similar learning properties as those of the human brain. In this context, FSL based models give good results and emerged as possible solutions for the aforementioned problem (Song, Wang, Cai, Mondal, & Sahoo, 2023; Wang, Yao, Kwok, & Ni, 2020). These techniques significantly reduce the reliance on large-annotated datasets, which are utilized to train different ML or DL models. Moreover, data annotation is a time-consuming and laborious task; therefore, FSL based models significantly reduce the human effort required to annotate the dataset. In this thesis, FSL is employed for image classification and segmentation; thus, the mathematical formulation of FSL is defined based on image data.

In order to formulate the FSL problem mathematically, let us consider an image dataset $D = \{(x_i, y_i)\}_{i=1}^{N}$, where x_i and y_i denotes the i^{th} image and its corresponding label, respectively. Additionally, the dataset D comprises of C number of classes. In the context of FSL, the number of instances in the dataset, i.e., N should be significantly lesser as compared to traditional ML or DL techniques.

In FSL, the entire dataset D is divided into two non-overlapping subsets named "Support set" and "Query set." The Support set (denoted by S) is used to train a FSL-

based model, and it comprises of k instances from each class of the dataset. The mathematical representation of the Support set is given in equation 2.13, where x_{s_i} and y_{s_i} denotes the i^{th} image and its corresponding label in the Support set, respectively. In literature, if there are k instances of C classes present in the Query set Q, then it is widely known as C-way-k-shot FSL problem. If $k = 0$ then it is termed as "Zero-Shot-Learning" and if $k = 1$, then it is called as "One-Shot-Learning."

$$S: \left[\{(x_{s_i}, y_{s_i})\}_{i=1}^{k}\right]_{c=1}^{C} \tag{2.13}$$

On the other hand, the Query set (denoted by Q) is used to evaluate the performance of the FSL-based model, and its mathematical representation is shown in equation 2.14. The objective of FSL is to learn a mapping function f^* by using the Support set, which minimizes the loss function L over the Query set Q. This objective of FSL in terms of Support and Query set is mathematically shown in equation 2.15, where $f(x_{q_i}; S)$ represents the prediction made by the function f for the i^{th} image present in Query set Q based on the support set S. Furthermore, f^* is the optimal function, which has the minimum value for loss function L.

$$Q: D - S = \{(x_{q_i}, y_{q_i})\}_{i=1}^{m} \tag{2.14}$$

$$f^* = \arg\min_f \frac{1}{m} \sum_{i=1}^{m} L(f(x_{q_i}; S), y_{q_i}) \tag{2.15}$$

In literature, the FSL techniques are broadly categorized into three categories, namely, Techniques based on distance metric learning, Techniques based on Hallucination, and Techniques based on Initialization (Chen, Liu, Kira, Wang, & Huang, 2019). These categories are described in subsequent subsections.

2.3.1 Techniques based on Distance Metric Learning

This set of techniques tries to conquer the FSL problem by learning to compare any two instances of the dataset. Intuitively, if a model can effectively differentiate two images belonging to different classes of the dataset, then it can learn to identify the classes of

unseen images by utilizing only limited data. Hence, these techniques reduce the reliance on large datasets by learning the similarities between images instead of learning various image features for classification. In literature, three distance metric learning based techniques are widely used, namely, Siamese Network, Triplet Network, and Prototypical Network. These techniques are described below:

A. Siamese Network

Siamese Network was initially introduced by Bromley et al. (Bromley, Guyon, LeCun, Säckinger, & Shah, 1994) for signature verification. It comprises of two identical neural networks (CNNs in the case of image data) that share the same weights and other hyperparameters. These neural networks are trained to represent the input images into a feature vector such that images belonging to the same class have similar feature vectors and images corresponding to different classes have dissimilar feature vectors. This goal is achieved by training these neural networks using contrastive loss function (Hadsell, Chopra, & LeCun, 2006).

Each of these neural networks takes an image as input and provides the feature vector of the input image as output. After that, the feature vectors provided by both neural networks are fed into a distance function (say Euclidean distance function) to measure the similarity between them. The similarity between two vectors is inversely proportional to the distance between them. Thereby, images belonging to the same class have a smaller distance between their feature vectors, whereas images corresponding to different classes have a larger distance between their feature vectors. As both neural networks share the same weight and other hyperparameter, thus the output of the Siamese Network remains unaffected by interchanging the input to the neural networks.

B. Triplet Network

These networks try to learn the similarities and dissimilarities simultaneously between the dataset's images. This FSL model utilizes three images as input (thereby known as a triplet network) referred by anchor, positive, and negative. Anchor represents the image that needs to be classified, and positive denotes the image belonging to the same class as the anchor. On the other hand, the negative indicates the image that does not correspond to the anchor's class.

In the context of images, CNNs are utilized to extract various spatial and temporal features and represent these extracted features as image embedding. The objective of triplet networks is to learn a representation of inputs such that the distance between the embedding of anchor and positive images is minimized, and the distance between the anchor image's embedding and the embedding of negative image is maximized. In order to achieve this, triplet networks employ the triplet loss function (Schroff, Kalenichenko, & Philbin, 2015), which is mathematically represented in equation 2.16. Here, in this equation, I_A, I_P, and I_N denotes the anchor, positive, and negative images, respectively. Additionally, E represents image embedding function, and δ denotes the margin enforced between negative and positive pairs of images. The distance between any pair of images a and b is computed using Euclidean distance formula, i.e., $\|a - b\|_2$.

$$L_{tn}(I_A, I_P, I_N) = \max(\|E(I_A) - E(I_P)\|_2 - \|E(I_A) - E(I_N)\|_2 + \delta, 0) \tag{2.16}$$

After completion of the training process, the triplet network acts as a feature extractor. In order to perform image classification, it is usually integrated with the k-Nearest Neighbor (KNN) classifier.

C. Prototypical Network

The key objective of the prototypical network is to compute a prototype for each class present in the support set and compare it with the embedding of images present in the query set (Snell, Swersky, & Zemel, 2017). The mathematical formula to calculate the prototype is given in equation 2.17, where P_c is the prototype of class c, k is the number of images present in the Support set for each class. Additionally, $E(x_i^c)$ is the embedding of i^{th} image belonging to c^{th} class.

$$P_c = \frac{1}{k}\sum_{i=1}^{k} E(x_i) \tag{2.17}$$

After prototype computation, the sample image from the Query set, i.e., x_q, is compared with the prototypes of classes via Euclidean distance function. In contrast to triplet networks, prototypical networks can perform image classification (as per

equation 2.18) without utilizing another classifier like KNN. In equation 2.18, y is the label of the query image, C is the number of classes present in the dataset, and P_i is the prototype of i^{th} class.

$$probabilty(y = c|x_q) = \frac{\exp\left(\left\|E(x_q) - P_c\right\|_2\right)}{\sum_{\forall i \in C} \exp\left(\left\|E(x_q) - P_i\right\|_2\right)} \tag{2.18}$$

Though distance metric learning based methods can effectively deal with the FSL problem, but they have several drawbacks that can adversely affect their performance. Prototypical Network requires sufficiently large Support set for each class during model training, which contradicts the problem of FSL. Triplet Networks, which involve training with triplets (i.e., anchor, positive, negative images), can be computationally expensive. Additionally, the need to carefully select useful triplets and increased training complexity can make the Triplet Network more resource-intensive. The major drawback of Siamese Networks is their reliance on a fixed-size embedding space. The fixed-size representation may limit the network's ability to effectively capture the complexity of diverse and high-dimensional input data, potentially leading to low performance in tasks with intricate class structures or varying feature representations.

2.3.2 Techniques based on Hallucination

These techniques employ various data generation methods in order to generate artificial data. This artificially generated data is then combined with existing training data (few training instances) to improve the model's ability to classify unseen images. The artificially generated instances are commonly known as *hallucinated instances*. In literature, various data generation methods are widely utilized. Some of these methods are discussed below:

A. Interpolation based Data Generation

Interpolation is a mathematical technique through which new data points are generated between two existing data points. There are various interpolation techniques available in the literature, namely, linear interpolation, bilinear interpolation, geometric interpolation, spherical linear interpolation, etc. Interpolation can be done in the input

space or feature space both. By leveraging interpolation, synthetic data can be generated, which conquers the challenge of limited labeled data in the FSL problem. However, the effectiveness of this approach depends on the effectiveness of the interpolation technique.

B. Digital Image Processing based Data Augmentation

It is a technique that artificially enlarges the dataset of images with the help of different Digital Image Processing techniques such as image rotation, zoom in, zoom out, image flipping, etc. (Bedi, Gole, & Agarwal, 2021). These transformations simulate different viewpoints, lighting conditions, and deformations that the model might encounter in real-world scenarios. In the context of FSL, data augmentation can be a viable solution as it has the potential to artificially enlarge the dataset even if a limited number of images are available.

C. Deep Learning based Data Generation

The aforementioned data augmentation technique solely relies on various predefined Digital Image Processing techniques. However, nowadays, researchers are widely utilizing DL based generative models like Generative Adversarial Networks (GANs) or Variational Auto Encoders (VAEs) for generating much more realistic images as compared to previous data augmentation techniques.

GANs employ two neural networks (CNNs in case of image data) named Generator and Discriminator for image generation. The Generator and Discriminator models are trained in an adversarial fashion, i.e., the Discriminator is trained in such a way that it can differentiate between original and generated images. On the other hand, the objective of the Generator is to generate very realistic images so that the Discriminator cannot distinguish between generated and original images. At the end of training, the Generator can effectively generate very realistic images. On the other hand, VAEs generate images by utilizing their encoder-decoder architecture. Initially, VAE utilizes its encoder and decoder both to learn the compressed domain representation (latent space) of input images. After that, it randomly selects a vector in latent space and then feeds this randomly selected vector to its decoder for image generation.

Hallucination-based methods offer the advantage of generating artificial data when only limited training data is available. However, certain drawbacks are also associated with these techniques. The first drawback is that there is a high risk of model overfitting due to the artificially generated instances or hallucinated instances, especially if the hallucinated instances do not represent the true distribution of original data. Furthermore, if the original data is biased, then hallucination based techniques may amplify the bias through generated instances, which leads to biased predictions. In addition to this, generative models like GANs and VAEs are computationally expensive, which limits the scalability and practicality of these methods.

2.3.3 Techniques based on Initialization

These techniques primarily focus on providing a good starting point for model training. As a result, the model can learn new tasks effectively with limited training data. In this context, two widely utilized approaches are, namely, the Meta Learning-based approach and the Transfer Learning based approach. First approach aims to learn good optimizer, and the second approach focuses on learning good weight parameters. These two approaches are described in the following subsections.

A. Meta-Learning based Approach

DL models utilize various optimization algorithms (commonly known as optimizers) to adjust their weight parameters by computing the gradient of loss function with respect to weights (Goodfellow, Bengio, & Courville, 2016). However, these gradient-based optimization algorithms, like Stochastic Gradient Descent (SGD), Adaptive Gradient (AdaGrad), etc., are not designed in such a way that they can deal with the limited amount of data. Therefore, various researchers modified the algorithms of existing optimizers so that the aforementioned problem could be solved. These modified algorithms are widely known as Meta-Learning algorithms, and they majorly focus on learning to learn. In literature, some commonly employed Meta-Learning algorithms are Model Agnostic Meta Learner (MAML) (Finn, Abbeel, & Levine, 2017), Meta-AdaM (Sun & Gao, 2023), etc. Although Meta-Learning based optimization algorithms have conquered the problem of training a DL with limited

training data. However, these algorithms are complex to design. Therefore, in the following subsection, the Transfer Learning based approach is described.

B. Transfer Learning based Approach

Transfer Learning based approaches aim to initialize optimal weight parameters for any DL model so that it can learn new tasks by utilizing limited training data. In this approach, a DL model is trained on a source task with an ample amount of annotated data. After that, this trained model is fine-tuned on a target task (which lies in the same domain) by utilizing limited annotated instances per class. In the pre-training phase, the model learns to extract generic features from input data, while the fine-tuning phase focuses on learning specific features required for new task by using only a few training samples per class. As compared to Meta-Learning based approaches, Transfer Learning based techniques are easier to design and implement. Therefore, the Transfer Learning approach of FSL has been utilized to design and implement the PDSE-Lite framework, which is described in chapter 5 of this thesis.

2.4 Chapter Summary

This chapter described the basic concepts of different DL and FSL based techniques, which have been utilized in subsequent chapters of this thesis. The next chapter discusses two lightweight DL models that can effectively and efficiently diagnose a plant disease from leaf images.

3. Detecting a Plant Disease from Leaf Images using Lightweight CNN Models

This chapter presents two lightweight CNN models, namely, the PlantGhostNet model and the lightweight hybrid model, which can diagnose a plant disease effectively and efficiently from digital leaf images. As compared to other state-of-the-art techniques present in the literature, these models have achieved high accuracy in detecting Bacterial Spot disease of Peach plants despite of utilizing significantly lesser number of trainable weight parameters.

3.1. Introduction

Food is one of the basic needs of human life. The demand for food is increasing due to an exponential increase in the world's population. In order to overcome such a massive demand for food, agricultural scientists recommend the use of different insecticides and pesticides to increase crop yield. However, using these in large amounts can degrade the soil quality, which makes crops more prone to different diseases. These diseases can negatively affect the crop yield and reduce the farmer's profit. If the farmers can diagnose plant diseases in their initial stages, it is possible to take timely actions to cure the plant diseases. However, detecting diseases in a large field of crops with the naked eye is a challenging task. Thus, in order to simplify this process, an automatic framework is needed for identifying plant diseases.

Initially researchers have leveraged various ML techniques for plant disease identification (Shruthi, Nagaveni, & Raghavendra, 2019; Shrivastava & Pradhan, 2020). However, these techniques suffer from two prominent issues. Firstly, they are unable to automatically extract image classification features such as shape, texture, and color. Secondly, these techniques are not implemented in such a way that they can take advantage of Graphic Processing Units (GPUs) for performing faster computations. Thus, in order to conquer these shortcomings of ML techniques, nowadays, researchers are utilizing DL methods (specifically CNN) for image-based plant disease diagnosis (Mohanty, Hughes, & Salathé, 2016; Ferentinos, 2018).

Although different CNN models employed in various existing research works have identified plant diseases very effectively, these CNN models require huge number of trainable weight parameters. Thus, they are very computationally expensive. Furthermore, large number of trainable weight parameters also increases the size of the CNN model, which restricts its usage on lightweight systems like Raspberry Pi, Android mobiles, etc. Therefore, in order to conquer the aforementioned challenges, this chapter proposes two lightweight CNN models for detecting a plant disease from leaf images. The first proposed model is PlantGhostNet, in which the Ghost (Han, et al., 2020) and Squeeze-and-Excitation (Hu, Shen, & Sun, 2018) modules have been utilized. In the PlantGhostNet model, the Ghost Module minimizes the number of trainable weight parameters, whereas the Squeeze-and-Excitation Module enhances the model's performance. Second proposed model is the lightweight hybrid model in which CAE and CNN are combined to further reduce the number of trainable weight parameters by a significant factor.

In order to test the applicability of the proposed models, these models are trained to identify the Bacterial Spot disease of Peach plants caused by Xanthomonas campestris bacteria. However, these models can be used to detect disease in other plants as well. The leaf images of Peach plants are extracted from the PlantVillage dataset (Hughes & Salathe, 2015), which serves as a benchmark dataset for evaluating any ML or DL model designed for image based plant disease detection. Various evaluation metrics, namely accuracy, precision, recall, and f1-measure, are used to evaluate the performance of proposed models. Performance of the proposed models has been compared with five state-of-the-art CNN models namely, namely, LeNet-5 (Lecun, Bottou, Bengio, & Haffner, 1998), AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), VGG-16 (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy, et al., 2015), ResNet-50 (He, Zhang, Ren, & Sun, 2016).

The rest of the chapter is organized into four sections. Section 3.2 presents the related work, and section 3.3 describes the architectures of proposed models. Section 3.4 discusses the experiments and the obtained results, followed by the chapter summary in section 3.5.

3.2. Related Work

Various researchers have proposed several state-of-the-art methods to automate the disease diagnosis process in plants. In this section, several existing research works that employed either ML or DL techniques for automatic plant disease diagnosis have been discussed. Ahmed et al. (Ahmed, Shahidi, Alam, & Momen, 2019) developed a disease detection system for identifying three types of diseases, namely Brown Spot, Bacterial Blight, and Leaf Smut, in Rice plants. In their work, they applied five ML techniques named Logistic Regression (LR), Naïve Bayes, K-Nearest Neighbor (KNN), and Decision Tree classifiers for plant disease detection. Out of these techniques, the Decision Tree classifier outperformed other classification techniques with an accuracy of 97.91%. Although ML techniques can identify plant diseases effectively, but these techniques require hand-crafted manually extracted features for classifying the input leaf image into either healthy or diseased classes. Nevertheless, manually extracting features from leaf images requires lots of human effort and time. Therefore, various researchers have tried to identify plant diseases via DL methods, in which feature extraction is done automatically.

Mohanty et al. (Mohanty, Hughes, & Salathé, 2016) used two popular CNN architectures named AlexNet (Krizhevsky, Sutskever, & Hinton, 2012) and GoogLeNet (Szegedy, et al., 2015) on the PlantVillage dataset. This dataset contains 54,306 images of healthy and diseased plant leaf images, which are distributed among 38 different classes. They performed 60 experiments by varying different parameters like CNN architecture (AlexNet, GoogLeNet), training mechanism (transfer learning or training from scratch), type of images (color, grayscale, and segmented), and train-test set distribution. Out of these 60 experimental configurations, transfer learning in GoogLeNet architecture using colored image version of the dataset partitioned in an 80%-20% split of train and test set outperformed others with an accuracy of 99.34%. Sanga et al. (Sanga, Machuve, & Jomanga, 2020) developed a disease detection tool for banana plants. They compared the performances of five different CNN architectures, namely VGG-16, ResNet-152, ResNet-50, ResNet-18, and InceptionV3, in detecting diseases from banana plant leaf images. The authors of paper found that ResNet-152 outperformed others with an accuracy of 99.2%. The number of trainable

weight parameters utilized by their best performing model, i.e., ResNet-152, was 60 million, as mentioned in the ResNet paper (He, Zhang, Ren, & Sun, 2016).

Similar work was also done in the paper authored by Chohan et al. (Chohan, Khan, Chohan, Katpar, & Mahar, 2020). They employed VGG-19 and InceptionV3 CNN architectures for automatic plant disease detection using the PlantVillage dataset. In their research work, they also used data augmentation to enlarge the dataset artificially. The VGG-19 model outperformed the InceptionV3 model with 98% training accuracy and 95% testing accuracy, as claimed by them in their paper. The number of training parameters used by their best performing model, i.e., VGG-19, was 143 million, as claimed by (Simonyan & Zisserman, 2015) in their research work. Ferentinos (Ferentinos, 2018) employed five different modern CNN architectures named AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat, and VGG-16 for plant disease detection using the PlantVillage dataset. In his paper, he found that VGG-16 outperformed other CNN architectures with an accuracy of 99.5% using 138 million trainable weight parameters, as mentioned in the VGG-16 paper (Simonyan & Zisserman, 2015).

Kumar and Vani (Kumar & Vani, 2019) performed different experiments on disease detection in Tomato leaves using four predefined CNNs, namely LeNet-5 (Lecun, Bottou, Bengio, & Haffner, 1998), VGG-16 (Simonyan & Zisserman, 2015), ResNet (He, Zhang, Ren, & Sun, 2016), and Xception (Chollet F. , 2017). They used 14,903 images of Tomato leaves which are evenly distributed among ten classes. Out of the different CNN architectures, VGG-16 outperformed others with an accuracy of 99.25%. Zhong Y and Zhao M (Zhong & Zhao, 2020) proposed a novel approach based on DenseNet-121 (Huang, Liu, Van Der Maaten, & Weinberger, 2017) for disease identification in Apple plants. They gathered Apple leaf images from a publicly available dataset named AI-Challenger-Plant Disease Recognition. The variants of their proposed model achieved 93.51%, 93.71%, and 93.31%, respectively, on the test data. Chen et al. (Chen, Chen, Zhang, Sun, & Nanehkaran, 2020) designed a system to identify diseases in Rice and Maize plants with the help of VGG-19 CNN architecture. Instead of training the model from scratch, they used a pre-trained VGG-19 model trained on the ImageNet dataset, which contains images of different categories in a large amount. Their model achieved 91.83% test accuracy. Karthik et al. (Karthik, et al.,

2020) have proposed the novel Residual Progressive Feature Extraction (RPFE) block and used this block to build the novel attention embedded residual CNN for automatic tomato plant disease identification. According to their paper, their proposed model has achieved 98% accuracy by utilizing roughly 7.2 million trainable weight parameters.

Zhao et al. (Zhao, et al., 2021) have used a combination of two Generative Adversarial Networks (GANs): Wasserstein GAN (WGAN) (Arjovsky, Chintala, & Bottou, 2017) and Super-Resolution GAN (SRGAN) (Ledig, et al., 2017). This combination was named as DoubleGAN in their paper, and it was used to augment the PlantVillage dataset. After this, they have trained three predefined CNN architectures: VGG16, ResNet-50, and DenseNet-121. They have concluded that DoubleGAN outperformed other augmenting methods like flipping, rotation, etc., and out of three CNN architectures, DenseNet-121 outperformed others with an accuracy of 99.70%. Mohameth et al. (Mohameth, Bingcai, & Sada, 2020) used different modern CNN architectures and different classifiers for automatic plant disease detection on the PlantVillage dataset. They employed VGG-16, ResNet-50, and GoogLeNet CNN architectures for feature extraction, and for classification, they used KNN and SVM classifiers. They observed that SVM with ResNet-50 outperformed others with an accuracy of 98%. As mentioned in the ResNet paper, the number of training parameters used by ResNet-50 was approximately 25 million (He, Zhang, Ren, & Sun, 2016).

Similar work was also done by Tiwari et al. (Tiwari, et al., 2020). They proposed an automatic disease detection system for potato plants. This system used different CNN architectures such as VGG-19, VGG-16, and InceptionV3 for feature extraction and different classifiers such as LR, KNN classifier, SVM, and Neural Network for disease detection. They concluded that VGG-19 with LR outperformed others with an accuracy of 97.8%. The number of training parameters used by VGG-19 was approximately 143 million, as claimed by (Simonyan & Zisserman, 2015). Khamparia et al. (Khamparia, et al., 2020) proposed a Deep Convolutional Encoder Network system for disease identification in seasonal crops. They considered 900 leaf images of three crops, potato, tomato, and maize, distributed in six classes (i.e., five diseased and one healthy). They achieved 100% training accuracy, while the testing accuracy of their model was 86.78%. Since the training accuracy was much higher as compared to the testing

accuracy, there was a chance that the model overfitted on the training data. In their paper, they have also mentioned that their system used approximately 3.3 million training parameters, which is much higher than the number of trainable weight parameters utilized by the models proposed in this chapter. The summary of all above-discussed research works has been given in Table 3.1.

Table 3.1: Summary of various research works present in the literature on automatic plant disease detection.

Research Work	**Dataset/ Crop**	**Technique Used**	**Accuracy**	**Number of trainable weight parameters**
(Mohanty, Hughes, & Salathé, 2016)	PlantVillage dataset	GoogLeNet	99.34%	7 million
(Ferentinos, 2018)	PlantVillage	VGG-16	99.5%	138 million
(Kumar & Vani, 2019)	PlantVillage dataset (Tomato)	VGG-16	99.25%	138.4 million
(Zhong & Zhao, 2020)	Apple dataset	DenseNet-121	93.71%	8.1 million
(Chen, Chen, Zhang, Sun, & Nanehkaran, 2020)	Rice and Maize datasets	VGG-19	91.83%	143.7 million
(Karthik, et al., 2020)	PlantVillage dataset (Tomato)	Custom CNN with RPFE block	98%	600 thousand
(Chohan, Khan, Chohan, Katpar, & Mahar, 2020)	PlantVillage	VGG-19	98.3%	143 million
(Khamparia, et al., 2020)	PlantVillage (Maize, Potato, and Tomato)	CAE	86.78%	3.3 million
(Mohameth, Bingcai, & Sada, 2020)	PlantVillage	ResNet-50 + SVM	98%	25 million

Research Work	Dataset/ Crop	Technique Used	Accuracy	Number of trainable weight parameters
(Sanga, Machuve, & Jomanga, 2020)	Self-collected dataset of Banana plants	ResNet-152	99.2%	60 million
(Tiwari, et al., 2020)	PlantVillage (Potato)	VGG-19 + Logistic Regression	97.8%	143 million
(Zhao, et al., 2021)	PlantVillage dataset	DoubleGAN + DenseNet-121	99.7%	8.1 million

All of the research works summarized in Table 3.1 are either suffer from low classification accuracy, or they utilize huge number of trainable weight parameters. Hence, this chapter proposes two lightweight CNN models that can identify a single disease from plant leaf images with high accuracy and a lesser number of trainable weight parameters. Next section of this chapter describes the proposed models.

3.3. Proposed Lightweight CNN Models for Detecting a Plant Disease from Leaf Images

This section comprises of two subsections in which the architectures of both proposed models are described. Subsection 3.3.1 describes the architecture of the PlantGhostNet model, and the architecture of the lightweight hybrid model has been discussed in subsection 3.3.2.

3.3.1. PlantGhostNet Model

The proposed PlantGhostNet model has been designed and developed by combining Ghost and Squeeze-and-Excitation modules. To the best of our knowledge, no research work present in the literature has proposed this combination. In this model, the Ghost Module is used to reduce the number of trainable parameters significantly, and the Squeeze-and-Excitation Module is utilized to boost the model's performance by

extracting features related to channels. The Ghost and Squeeze-and-Excitation modules have been described below:

A. Ghost Module

Han et al. (Han, et al., 2020) observed many redundant feature maps (Ghost feature maps) present in the well-trained ResNet-50 CNN architecture, and these redundant feature maps are generated with the help of large number of trainable weight parameters. Therefore, Ghost Module conquered this drawback of existing CNN architectures by adopting a two-fold approach for generating similar feature maps as of existing CNNs. This approach is described below:

1. **First Phase:** In the first phase, few (say $m \leq z$, where z are total feature map needs to be generated) feature maps are generated using the traditional convolution operation with the help of $k \times k$ convolutional filters. This operation is mathematically represented in equation 3.1, where $B' \in \mathbb{R}^{x \times y \times m}$ represents a set of m feature maps generated in the first phase, $f' \in \mathbb{R}^{c \times k \times k \times m}$ denotes the set of convolutional filters utilized to generate B', and g' is the bias term.

$$B' = A * f' + g' \tag{3.1}$$

   Feature maps generated in the first phase are known as intrinsic feature maps and later utilized in the second phase to generate Ghost feature maps. The m is the hyper-parameter, and its value can be tuned in fine-tuning process.

2. **Second Phase:** In the second phase, Ghost feature maps are generated by applying a series of cheap[1] linear operations (denoted by ϕ) on the intrinsic feature maps generated in the first phase. Each intrinsic feature map present in the set of m feature maps generate s new feature maps as per equation 3.2. In this equation, B'_i is the i^{th} intrinsic feature of B', $\phi_{i,j}$ is the j^{th} linear operation

[1] Han et al., (2020) referred the Depthwise convolutional operations as cheap linear operations because these operations require significantly lesser number of trainable weight parameters than conventional convolutional operation.

for generating j^{th} Ghost feature map from $B_i{}'$ and the output after applying $\phi_{i,j}$ operation is represented by B_{ij}.

$$B_{ij} = \phi_{i,j}(B_i'), \ \ \forall i = 1,2,\dots,m, \ \ j = 1,2,\dots,s \tag{3.2}$$

Hence, by following the aforementioned two phases, the desired $z = m \cdot s$ feature maps are generated with the help of the Ghost Module.

The number of trainable weight parameters utilized in Ghost Module can be computed in two parts, as it encompasses of two phases for generating feature maps. The weight parameters used in the first phase and second phase of Ghost Module have been mathematically calculated in equations 3.3 and 3.4, respectively. In these equations, N_{Ghos}^1 and N_{Ghos}^2 denotes the number of trainable weight parameters utilized in the first and second phases of the Ghost Module, respectively. Moreover, d represents the average kernel size of each linear operation. In a practical scenario, k and d have similar magnitude and $s \ll c$. Therefore, the total number of trainable weight parameters utilized in a Ghost Module can be expressed by equation 3.5.

$$N_{Ghost}^1 = \frac{z}{s} \cdot c \cdot k \cdot k \tag{3.3}$$

$$N_{Ghost}^2 = \frac{z}{s} \cdot (s-1) \cdot d \cdot d \tag{3.4}$$

$$N_{Ghost} = N_{Ghost}^1 + N_{Ghost}^2 = \frac{z}{s} \cdot k \cdot k \cdot (c + s - 1) \tag{3.5}$$

The speed-up ratio (r_s) by using Ghost Module in place of the traditional convolutional layer can computed as per equation 3.6.

$$r_s = \frac{N_{conv}}{N_{Ghost}} = \frac{c \cdot k \cdot k \cdot z}{{}^{z}/_{s} \cdot k \cdot k \cdot (c + s - 1)} = \frac{s \cdot c}{c + s - 1} \approx s \tag{3.6}$$

It can be concluded from equation 3.6 that the Ghost Module can reduce trainable weight parameters by a factor of s. The other building block of the proposed PlantGhostNet model is the Squeeze-and-Excitation Module, which is described in the following subsection.

B. Squeeze-and-Excitation Module

The convolutional filters used in convolutional layers of CNN extract hierarchical information from the images. The initial convolutional layers extract low-level features like edges or corners, whereas layers present at the end of CNN extract high-level features like faces, curves, etc. The convolutional layers perform the information extraction by using spatial and channel information of the images. The major shortcoming of the convolutional layer is that it gives equal weightage to all channels of input feature maps while creating the output feature maps.

Thus, to overcome this shortcoming of convolutional layers, Hu et al. (Hu, Shen, & Sun, 2018) designed the Squeeze-and-Excitation Module that adaptively gives weight to each channel of the feature map. The Squeeze-and-Excitation module comprises of two phases: Squeeze Phase (Global Information Embedding) and Excitation Phase (Adaptive Recalibration). These two phases are illustrated below:

1. **Squeeze Phase (Global Information Embedding):** In this phase, a vector (say z_c) of size $1 \times 1 \times P_C$, is formed by applying the Global Average Pooling operation to the input feature map (say P) of size $P_D \times P_D \times P_C$, where P_D is the height and width of P, and P_C is the number of channels in P. The mathematical expression for the Global Average Pooling operation (denoted by s_f), is shown in equation 3.7.

$$s_f = \frac{1}{P_D \times P_D} \sum_{a=1}^{P_D} \sum_{b=1}^{P_D} P[a, b] \tag{3.7}$$

2. **Excitation Phase (Adaptive Recalibration):** In this phase, the output of the previous phase is passed to a simple gating mechanism with a sigmoid activation function in the output layer. Mathematical expression for the gating mechanism is shown in equation 3.8,

$$e_f = \sigma\big(g(z_c, w)\big) = \sigma\big(w_2^T \gamma(w_1^T z_c)\big) \tag{3.8}$$

In equation 3.10, γ is the ReLU function, σ denotes the sigmoid activation function, $w_1 \in \mathbb{R}^{\frac{C}{R} \times C}$, and $w_2 \in \mathbb{R}^{C \times \frac{C}{R}}$ are weight matrices of the first and second layers, respectively. Moreover, R denotes the reduction ratio. Finally, the output of the Squeeze-and-Excitation Module is obtained by performing channel wise multiplication operation between the output of the Excitation Phase, i.e., e_f, and the input feature map, i.e., P.

The flow diagram for both phases of the Squeeze-and-Excitation Module has been shown in Figure 3.1.

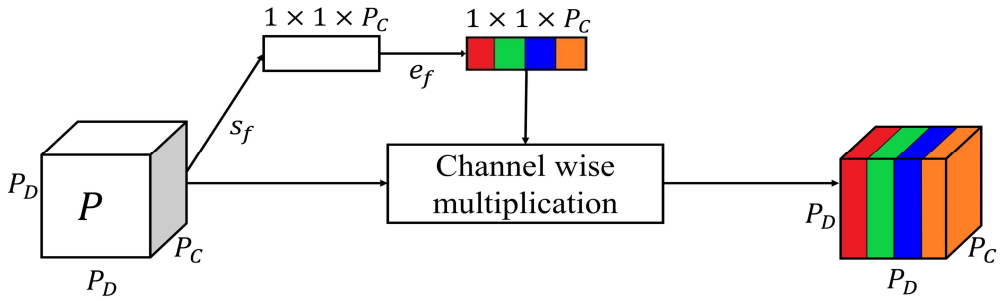


Figure 3.1: Flow diagram of Squeeze-and-Excitation Module

In order to analyze the efficiency of the PlantGhostNet model in terms of the number of trainable parameters, a baseline CNN model has been designed and implemented without using Ghost and Squeeze-and-Excitation modules. The layer-wise implementation details of baseline and PlantGhostNet models are tabulated in Tables 3.2 and 3.3, respectively.

Table 3.2: Layer-wise implementation details of baseline CNN architecture

Layer	Layer details	Input shape	Output shape	Number of trainable weight parameters
Input layer	-	-	$256 \times 256 \times 3$	0
Convolutional layer #1	64 filter of size 3×3	$256 \times 256 \times 3$	$256 \times 256 \times 64$	1,792

Layer	Layer details	Input shape	Output shape	Number of trainable weight parameters
Max-Pooling layer #1	Pool window=2 × 2	256 × 256 × 64	128 × 128 × 64	0
Convolutional layer #2	128 filter of size 3 × 3	128 × 128 × 64	128 × 128 × 128	73,856
Max-Pooling layer #2	Pool window=2 × 2	128 × 128 × 128	64 × 64 × 128	0
Convolutional layer #3	256 filter of size 3 × 3	64 × 64 × 128	64 × 64 × 256	295,168
Max-Pooling layer #3	Pool window=2 × 2	64 × 64 × 256	32 × 32 × 256	0
Convolutional layer #4	128 filter of size 3 × 3	32 × 32 × 256	32 × 32 × 128	295,040
Max-Pooling layer #4	Pool window=2 × 2	32 × 32 × 128	16 × 16 × 128	0
Convolutional layer #5	64 filter of size 3 × 3	16 × 16 × 128	16 × 16 × 64	73,792
Max-Pooling layer #5	Pool window=2 × 2	16 × 16 × 64	8 × 8 × 64	0

Layer	Layer details	Input shape	Output shape	Number of trainable weight parameters
Convolutional layer #6	32 filter of size 3×3	$8 \times 8 \times 64$	$8 \times 8 \times 32$	18,464
Max-Pooling layer #6	Pool window=2×2	$8 \times 8 \times 32$	$4 \times 4 \times 32$	0
Convolutional layer #7	16 filter of size 3×3	$4 \times 4 \times 32$	$4 \times 4 \times 16$	4,624
Max-Pooling layer #7	Pool window=2×2	$4 \times 4 \times 16$	$2 \times 2 \times 16$	0
Flattening layer	-	$2 \times 2 \times 16$	64	0
Dense layer #1	32 neurons activated by ReLU function	64	32	2,080
Dense layer #2 (Output layer)	1 neuron activated by sigmoid function	32	1	33
Total trainable parameters				**764,849**

Table 3.3: Layer-wise implementation details of the proposed PlantGhostNet model

Layer	Layer details	Input shape	Output shape	Number of trainable weight parameters
Input layer	-	-	$256 \times 256 \times 3$	0
Ghost module #1	$z = 64$, $k, d = 3 \times 3$ $s = 16$	$256 \times 256 \times 3$	$256 \times 256 \times 64$	648
Squeeze-and-Excitation module #1	$R = 8$	$256 \times 256 \times 64$	$256 \times 256 \times 64$	1024
Max-Pooling layer #1	Pool window=2×2	$256 \times 256 \times 64$	$128 \times 128 \times 64$	0
Ghost module #2	$z = 128$, $k, d = 3 \times 3$ $s = 16$	$128 \times 128 \times 64$	$128 \times 128 \times 128$	5,688
Squeeze-and-Excitation module #2	$R = 16$	$128 \times 128 \times 128$	$128 \times 128 \times 128$	2184
Max-Pooling layer #2	Pool window=2×2	$128 \times 128 \times 128$	$64 \times 64 \times 128$	0
Ghost module #3	$z = 256$, $k, d = 3 \times 3$ $s = 16$	$64 \times 64 \times 128$	$64 \times 64 \times 256$	20,592
Squeeze-and-Excitation module #3	$R = 16$	$64 \times 64 \times 256$	$64 \times 64 \times 256$	8,464
Max-Pooling layer #3	Pool window=2×2	$64 \times 64 \times 256$	$32 \times 32 \times 256$	0

Layer	Layer details	Input shape	Output shape	Number of trainable weight parameters
Ghost module #4	$z = 128$, $k, d = 3 \times 3$ $s = 16$	$32 \times 32 \times 256$	$32 \times 32 \times 128$	19,512
Squeeze-and-Excitation module #4	$R = 16$	$32 \times 32 \times 128$	$32 \times 32 \times 128$	2,184
Max-Pooling layer #4	Pool window=2×2	$32 \times 32 \times 128$	$16 \times 16 \times 128$	0
Ghost module #5	$z = 64$, $k, d = 3 \times 3$ $s = 16$	$16 \times 16 \times 128$	$16 \times 16 \times 64$	5,148
Squeeze-and-Excitation module #5	$R = 8$	$16 \times 16 \times 64$	$16 \times 16 \times 64$	1,096
Max-Pooling layer #5	Pool window=2×2	$16 \times 16 \times 64$	$8 \times 8 \times 64$	0
Ghost module #6	$z = 32, k, d = 3 \times 3$ $s = 8$	$8 \times 8 \times 64$	$8 \times 8 \times 32$	2,556
Squeeze-and-Excitation module #6	$R = 4$	$8 \times 8 \times 32$	$8 \times 8 \times 32$	552
Max-Pooling layer #6	Pool window=2×2	$8 \times 8 \times 32$	$4 \times 4 \times 32$	0

Layer	**Layer details**	**Input shape**	**Output shape**	**Number of trainable weight parameters**
Ghost module #7	$z = 16, k, d = 3 \times 3$ $s = 4$	$4 \times 4 \times 32$	$4 \times 4 \times 16$	1,260
Squeeze-and-Excitation module #7	$R = 2$	$4 \times 4 \times 16$	$4 \times 4 \times 16$	280
Max-Pooling layer #7	Pool window=2×2	$4 \times 4 \times 16$	$2 \times 2 \times 16$	0
Flattening layer	-	$2 \times 2 \times 16$	64	0
Dense layer #1	32 neurons activated by ReLU function	64	32	2,080
Dense layer #2 (Output layer)	1 neuron activated by sigmoid function	32	1	33
			Total trainable parameters	**73,301**

It can be observed from Tables 3.2 and 3.3 that the PlantGhostNet model utilizes around ten times fewer trainable weight parameters than the baseline CNN model created without using Ghost and Squeeze-and-Excitation modules. However, the trainable weight parameters of the PlantGhostNet model are around seventy-three thousand, which is still high. Therefore, a lightweight hybrid model based on CAE and CNN has also been proposed in this chapter, which reduces the trainable weight parameters further by a significant factor. This model has been described in the next subsection.

3.3.2. Lightweight Hybrid Model based on CAE and CNN

The proposed lightweight hybrid model utilizes two DL techniques, namely CAE and CNN, which have been described in the previous chapter. In this model, firstly the CAE

network has been trained to reduce the spatial dimensions of leaf images. The reduction of spatial dimensions of the leaf images has been done such that the important features of the leaf images are preserved. This has been ensured by applying the upper limit on the Reconstruction Loss of CAE. After reducing the spatial dimensions of leaf images, the output of the Encoder Network of CAE (i.e., compressed domain representations of leaf images) is used as input to the CNN. With the help of CNN, the input leaf images have been classified into either diseased or healthy classes.

The process of designing the proposed lightweight hybrid model comprises of two steps. In the first step, a CAE network has been designed and developed to reduce the spatial dimensions of the input leaf images from $256 \times 256 \times 3$ to $32 \times 32 \times 8$. Architecture of the CAE network is shown in Figure 3.2, and its layer-wise implementation details are tabulated in Table 3.4.

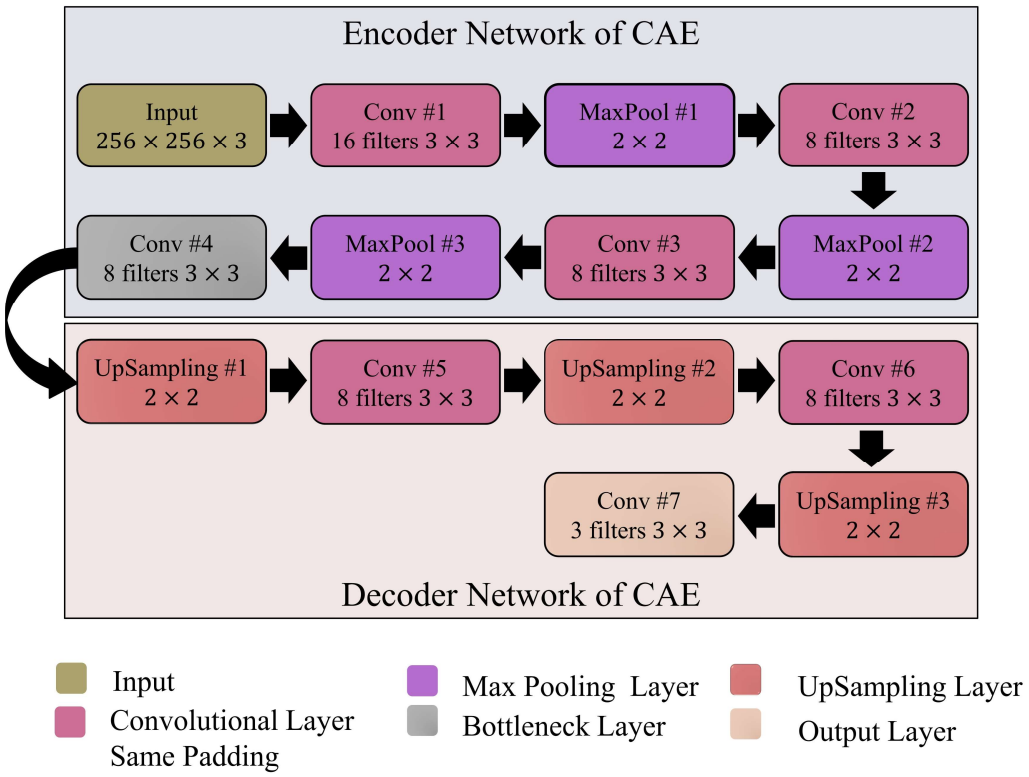


Figure 3.2: CAE network architecture for the proposed lightweight hybrid model

The CAE network also contains a Decoder Network that is used to reconstruct the original leaf images from the encoded data. Training of the CAE network is done such that the Reconstruction Loss is minimized. This ensures that the CAE network reduces

the spatial dimensions of the leaf images without losing their important features. In this chapter, the Normalized Root Mean Squared Error (NRMSE) (Feng, Feng, Ozer, & Fukuda, 2015) loss function is used to compute the Reconstruction Loss between the original leaf images and the reconstructed leaf images. The formula to compute NRMSE is shown in equation 3.9. In this equation, X^O is the set of original leaf images, X^R is the set of leaf images reconstructed by the CAE network, N is the total number of leaf images taken into consideration. Moreover, $P_{maximum}$ denotes the maximum pixel intensity in the input leaf images, i.e., 255 and $P_{minimum}$ is the minimum pixel intensity in the input leaf images, i.e., 0.

$$NRMSE\ (X^O, X^R) = \frac{\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X^O - X^R)^2}}{P_{maximum} - P_{minimum}} \tag{3.9}$$

Table 3.4: Layer-wise implementation details of CAE network architecture for the proposed lightweight hybrid model

Layer number	Layer name	Input shape	Output shape	Number of trainable weight parameters
1	Input Layer	256 × 256 × 3	256 × 256 × 3	0
2	Conv #1	256 × 256 × 3	256 × 256 × 16	448
3	MaxPool #1	256 × 256 × 16	128 × 128 × 16	0
4	Conv #2	128 × 128 × 16	128 × 128 × 8	1,160
5	MaxPool #2	128 × 128 × 8	64 × 64 × 8	0
6	Conv #3	64 × 64 × 8	64 × 64 × 8	584
7	MaxPool #3	64 × 64 × 8	32 × 32 × 8	0
8	Conv #4 (Bottleneck Layer)	32 × 32 × 8	32 × 32 × 8	584
9	UpSampling Layer #1	32 × 32 × 8	64 × 64 × 8	0
10	Conv #5	64 × 64 × 8	64 × 64 × 8	584

Layer number	Layer name	Input shape	Output shape	Number of trainable weight parameters
11	UpSampling Layer #2	64 × 64 × 8	128 × 128 × 8	0
12	Conv #6	128 × 128 × 8	128 × 128 × 8	584
13	UpSampling Layer #3	128 × 128 × 8	256 × 256 × 8	0
14	Conv #7 (Output Layer)	256 × 256 × 8	256 × 256 × 3	219
			Total number of trainable parameters	**4,163**

After reducing the spatial dimensions of input leaf images, CNN has been applied to classify the input leaf images into either diseased or healthy classes. The output of the Bottleneck Layer of the CAE is taken as the input for the CNN. Figure 3.3 depicts the architecture of CNN that is used to classify leaf images. The proposed lightweight hybrid model has been designed by concatenating the layers of the Encoder network of CAE and the layers of the CNN. The architecture of the proposed lightweight hybrid model is shown in Figure 3.4, and details of its layer-wise implementation are shown in Table 3.5.

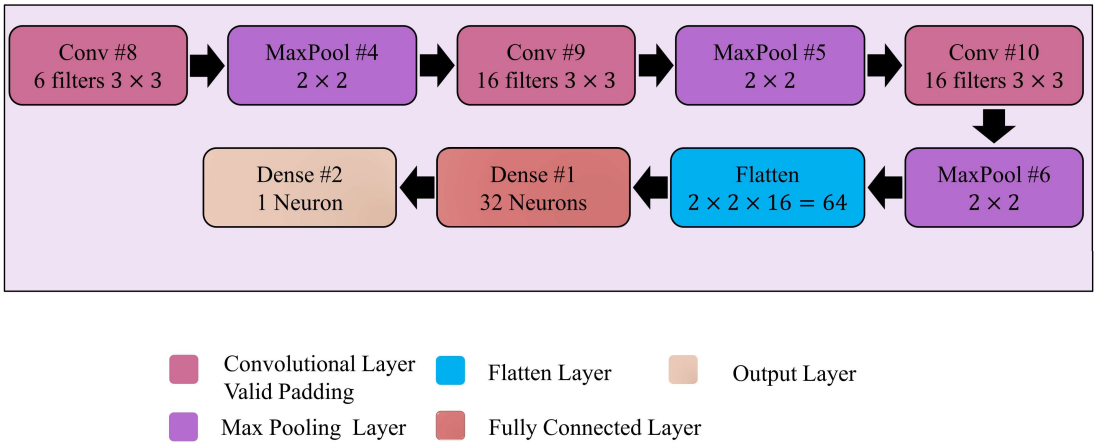


Figure 3.3: CNN architecture for the proposed lightweight hybrid model

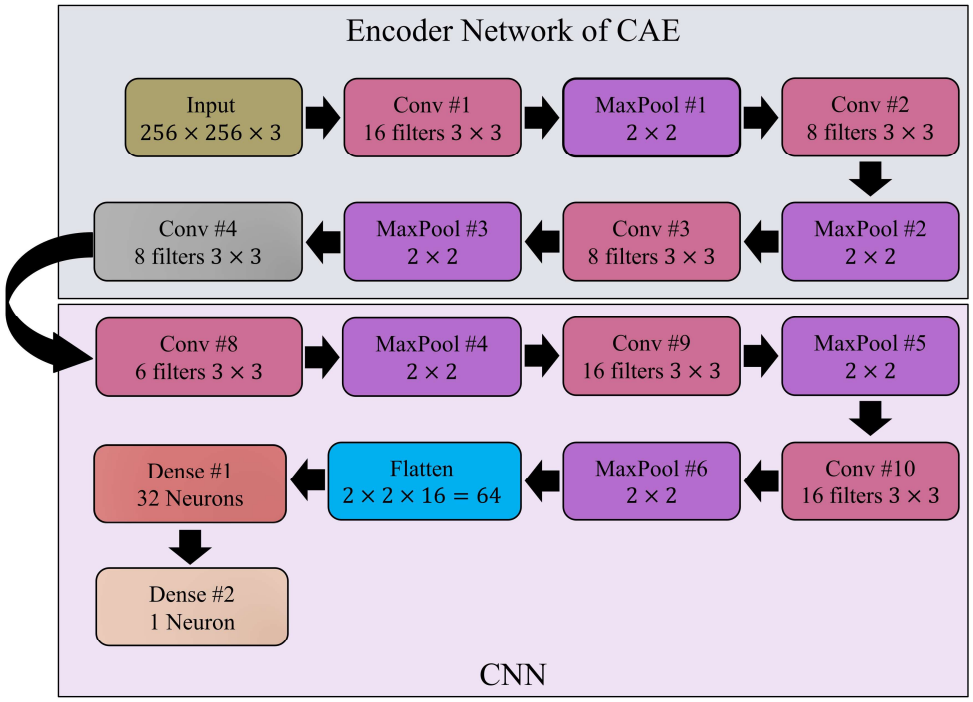


Figure 3.4: Architecture of the proposed lightweight hybrid model

Table 3.5: Layer-wise implementation details of the proposed lightweight hybrid model

Layer number	**Layer name**	**Input shape**	**Output shape**	**Number of trainable weight parameters**
1	Input Layer	256 × 256 × 3	256 × 256 × 3	0
2	Conv #1	256 × 256 × 3	256 × 256 × 16	448
3	MaxPool #1	256 × 256 × 16	128 × 128 × 16	0
4	Conv #2	128 × 128 × 16	128 × 128 × 8	1,160
5	MaxPool #2	128 × 128 × 8	64 × 64 × 8	0
6	Conv #3	64 × 64 × 8	64 × 64 × 8	584
7	MaxPool #3	64 × 64 × 8	32 × 32 × 8	0
8	Conv #4 (Bottleneck Layer)	32 × 32 × 8	32 × 32 × 8	584
9	Conv #8	32 × 32 × 8	30 × 30 × 6	438

Layer number	Layer name	Input shape	Output shape	Number of trainable weight parameters
10	MaxPool #4	$30 \times 30 \times 6$	$15 \times 15 \times 6$	0
11	Conv #9	$15 \times 15 \times 6$	$13 \times 13 \times 16$	880
12	MaxPool # 5	$13 \times 13 \times 16$	$6 \times 6 \times 16$	0
13	Conv #10	$6 \times 6 \times 16$	$4 \times 4 \times 16$	2,320
14	MaxPool # 6	$4 \times 4 \times 16$	$2 \times 2 \times 16$	0
15	Flatten Layer	$2 \times 2 \times 16$	64	0
16	Dense #1	64	32	2,080
17	Dense #2 (Sigmoid)	32	1	33
Total trainable weight parameters				**5,751**
Total non-trainable parameters				**2,776**
Total weight parameters				**8,527**

The layers of the Encoder network of CAE (i.e., layer 1 to layer 8) and layers of CNN (i.e., layer 8 to layer 17) have been used to create the proposed lightweight hybrid model. Since the layers imported from the encoder network of CAE are pre-trained, the trainable weight parameters of these layers are not included while computing the trainable weight parameters of the hybrid model. Hence, it can be observed from Table 3.5 that the lightweight hybrid model uses only 5,751 trainable weight parameters and 2,776 non-trainable weight parameters. Moreover, it can also be observed from Table 3.4 that the CAE network uses 4,163 trainable weight parameters. Thus, the total trainable weight parameters used in the proposed lightweight hybrid model can be expressed by equation 3.10. Here, TT_{wp}^{LHM} represents the total number of trainable weight parameters used in the lightweight hybrid model, and TWP^{LHM} denotes the total number of weight parameters utilized in the lightweight hybrid model. Furthermore,

T_{wp}^{CAE} represents the trainable weight parameters of the CAE model, and NT_{wp}^{LHM} denotes the number of non-trainable weight parameters of the lightweight hybrid model.

$$TT_{wp}^{LHM} = TWP^{LHM} + T_{wp}^{CAE} - NT_{wp}^{LHM} = 8527 + 4163 - 2776 = 9914 \qquad (3.10)$$

Next section of this chapter presents the experimental study performed to analyze the performances of proposed PlantGhostNet and lightweight hybrid models. Moreover, the experimental results obtained from different experiments are also discussed in the next section.

3.4. Experimental Study and Results

The practical implementation of the proposed models has been done on a Windows system with an Intel® i7-10750H processor, 32GB RAM, and Nvidia GeForce GTX 1650 graphic card. Moreover, PyCharm Integrated Development Environment (IDE) has been used to write the Python script to perform the experiments. However, these experiments can also be performed using other programming languages like R, MATLAB, etc. The proposed models are developed and trained using Keras Application Programming Interface (API) (Chollet F. , 2015) with Tensorflow backend.

The performance of proposed PlantGhostNet and lightweight hybrid models has been compared with the baseline CNN model along with five state-of-the-art CNN architectures, namely, LeNet-5 (Lecun, Bottou, Bengio, & Haffner, 1998), AlexNet (Krizhevsky, Sutskever, & Hinton, 2012), VGG-16 (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy, et al., 2015), ResNet-50 (He, Zhang, Ren, & Sun, 2016). This section is divided into three subsections. Section 3.4.1 provides the details of the dataset that has been utilized for training and evaluating the PlantGhostNet model, lightweight hybrid model, baseline CNN model, and five state-of-the-art CNN architectures. Section 3.4.2 provides the best values of different hyperparameters, followed by section 3.4.3, in which the experimental results are provided and discussed.

3.4.1. Dataset Description

The PlantGhostNet and lightweight hybrid models have been trained on Peach plant leaf images collected from the PlantVillage dataset (Hughes & Salathe, 2015). There

are 2,160 healthy leaf images and 2,297 leaf images suffered from Bacterial Spot disease, which is caused by the Xanthomonas Campestris bacterium. Few healthy and diseased leaf images of Peach plants are shown in Figure 3.5.

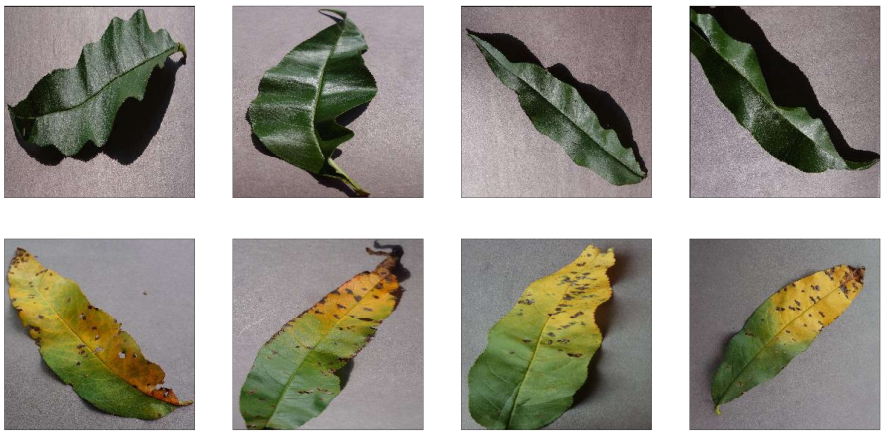


Figure 3.5: Few healthy and diseased leaf images of Peach plants

The *train_test_split* function of the scikit-learn API (Pedregosa, Varoquaux, Gramfort, & V., 2011) is used to divide these images in a 70:15:15 ratio to create the training, validation, and testing subsets. Hence, the training subset has 3,120 leaf images, the validation subset has 668 leaf images, and the test subset has 669 leaf images. The leaf images present in the training subset are utilized to train the proposed models, and the validation subset's leaf images are used to tune different hyperparameters of the proposed models. On the other hand, leaf images available in test subsets are utilized to evaluate the performance of the proposed models on unseen data.

Next section provides the best values of different hyperparameters for proposed PlantGhostNet and lightweight hybrid models.

3.4.2. Hyperparameter Selection

The proposed models comprise of various hyperparameters, and the values of each and every hyperparameter can significantly affect the model's performance. Hence, in order to get the best values of hyperparameters, the PlantGhostNet and lightweight hybrid models have been trained and tested on different values of hyperparameters. After extensive experimentation, the best values of these hyperparameters for PlantGhostNet and lightweight hybrid models have been obtained and tabulated in Table 3.6 and Table 3.7, respectively.

Table 3.6: Best values of different hyperparameters used to implement the PlantGhostNet model

Hyperparameter	Number of Ghost and Squeeze-and-Excitation modules	Activation function	Loss function	Optimizer	Learning rate	Epochs	Batch size
Value	7	ReLU (for hidden layers) Sigmoid (for output layer)	MSE	Adam (Kingma & Ba, 2014)	0.001	100	16

Table 3.7: Best values of different hyperparameters used to develop the lightweight hybrid model

Hyperparameter	Loss function	Optimizer	Learning rate	Epochs	Batch size
Value	NRMSE (for CAE) MSE (for lightweight hybrid model)	Adam (Kingma & Ba, 2014)	0.001	200 (for CAE) 100 (for lightweight hybrid model)	32

Early stopping has been used to prevent the proposed models from overfitting. The patience value for early stopping is 5 (i.e., if the testing loss does not improve over five consecutive epochs, then the training will stop). Next subsection provides and discusses the results obtained from experimentation.

3.4.3. Results and Discussion

Performance of the proposed models has been evaluated and compared with the baseline CNN model and five state-of-the-art CNN architectures with the help of four evaluation metrics, namely accuracy, precision, recall, and f1-measure. These metrics are expressed in terms of True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN).

In the domain of plant disease detection, TP signifies the number of diseased leaf images that are classified as diseased. On the other hand, TN connotates the number of

healthy leaf images that are classified as healthy by the model. The FN and FP represent the number of healthy and diseased leaf images that are classified as diseased and healthy by the model, respectively. The mathematical formulas of these metrics are given in equations 3.11, 3.12, 3.13, and 3.14, respectively (Zaki & Wagner, 2020).

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.11}$$

$$precision = \frac{TP}{TP + FP} \tag{3.12}$$

$$recall = \frac{TP}{TP + FN} \tag{3.13}$$

$$f1\text{-}measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = \frac{2 \times precision \times recall}{precision + recall} \tag{3.14}$$

In this section, the results of PlantGhostNet model are given first, followed by the results of the lightweight hybrid model.

A. Results of PlantGhostNet Model

Performance of PlantGhostNet and baseline CNN models, along with five other state-of-the-art CNN architectures, have been evaluated on validation subset using accuracy and MSE loss metric. The trend of validation accuracy (accuracy on validation subset) and validation loss (MSE loss on validation subset) for the aforementioned models has been represented by a line chart shown in Figures 3.6 and 3.7, respectively.

It can be observed from Figure 3.6 that the proposed PlantGhostNet model has achieved maximum validation accuracy, i.e., 99.75%. Furthermore, VGG-16 achieved minimum accuracy, i.e., 93.83%, among other models. The performances of AlexNet and ResNet-50 CNN architectures are comparable. Additionally, LeNet-5 and GoogLeNet achieved 95.53% and 98.02% validation accuracies, respectively. The same trend can also be observed for the validation loss, as shown in Figure 3.7.

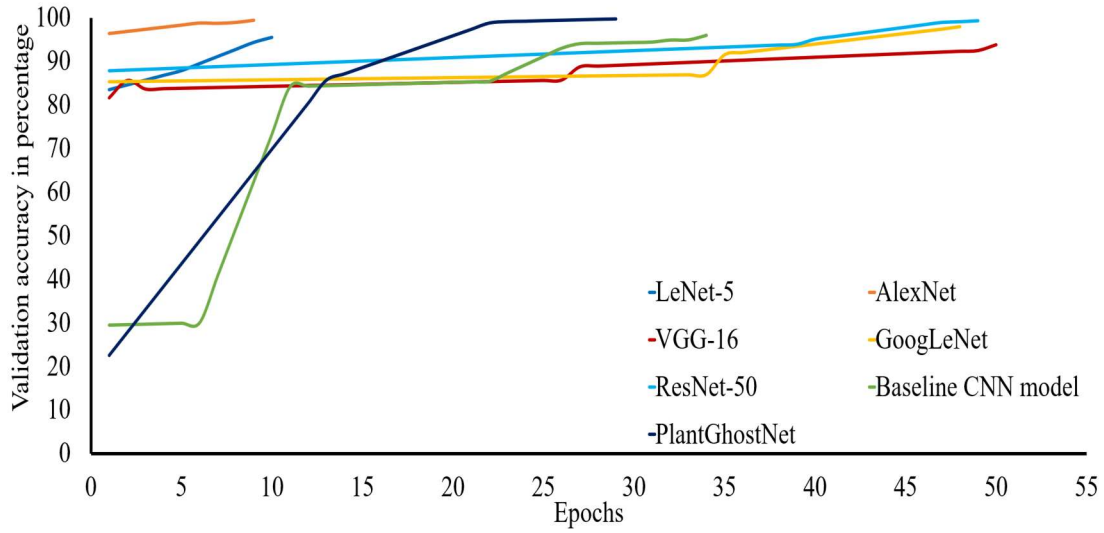


Figure 3.6: Trend of validation accuracy with respect to the number of epochs for proposed PlantGhostNet and baseline CNN model along with five state-of-the-art CNN architectures

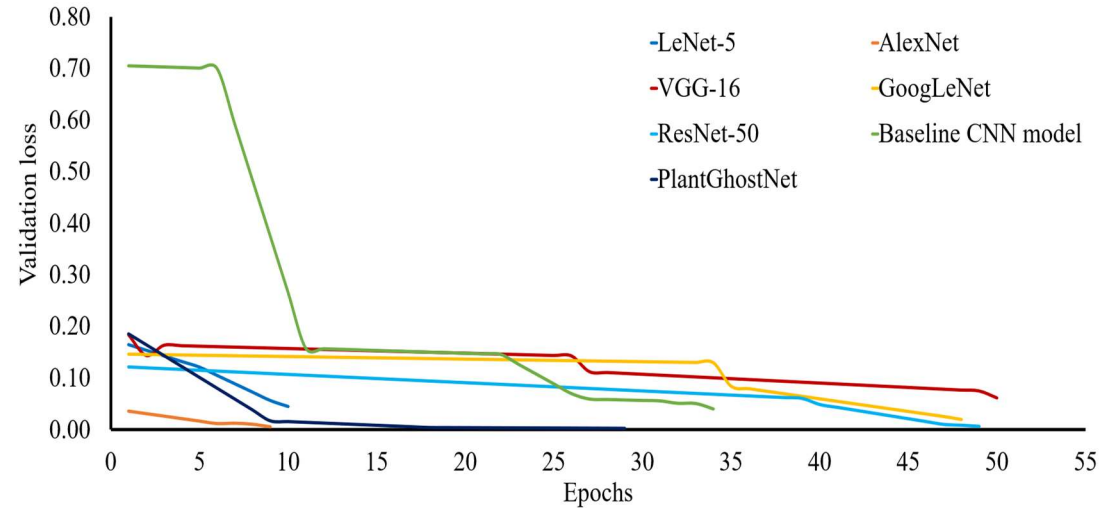
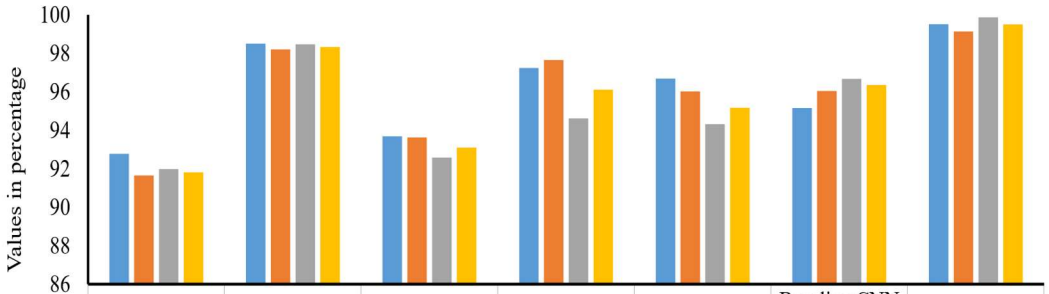


Figure 3.7: Trend of validation loss with respect to the number of epochs for proposed PlantGhostNet and baseline CNN model along with five state-of-the-art CNN architectures

The proposed PlantGhostNet model's performance is also evaluated and compared with baseline and five other state-of-the-art CNN architectures on testing subset via accuracy, precision, recall, and f1-measure metrics. The scores of these metrics for all models have been plotted with the help of a bar graph in Figure 3.8.



	LeNet-5	AlexNet	VGG-16	GoogLeNet	ResNet-50	Baseline CNN model	PlantGhostNet
Accuracy	92.78	98.5	93.68	97.24	96.69	95.15	99.51
Precision	91.65	98.2	93.63	97.65	96.02	96.04	99.13
Recall	91.98	98.47	92.58	94.62	94.32	96.67	99.87
F1-measure	91.81	98.33	93.10	96.11	95.16	96.35	99.50

Figure 3.8: Accuracy, precision, recall, and f1-measure of proposed PlantGhostNet and baseline CNN model along with five state-of-the-art CNN architectures

It can be perceived from this figure that the PlantGhostNet model outperformed other architectures with 99.51% testing accuracy and 99.5% f1-measure. In addition, GoogLeNet, ResNet-50, and the baseline CNN model have achieved comparable performance. On the other hand, LeNet-5 and VGG-16 have attained minimum values for the aforementioned metrics. Next subsection provides the results of proposed lightweight hybrid model.

B. Results of Lightweight Hybrid Model

In this section, first, the results of the CAE network have been shown. After that, the results of the proposed lightweight hybrid model are presented.

The NRMSE loss has been used to evaluate the performance of the CAE network. It is computed with the help of original leaf images and reconstructed leaf images using equation 3.9. The change in NRMSE loss with respect to the number of epochs on the training and validation subsets of the dataset has been shown in Figure 3.9. Moreover, few original leaf images with their corresponding reconstructed leaf images have been depicted in Figure 3.10.

It can be observed from Figure 3.9 that the NRMSE loss is reducing as the number of epochs increases, and after the 22^{nd} epoch, the loss does not change significantly. During experimentation, it is found that after completion of the training process, the

values of NRMSE loss on training, validation, and test subsets are 0.0597, 0.0607, and 0.0612, respectively.

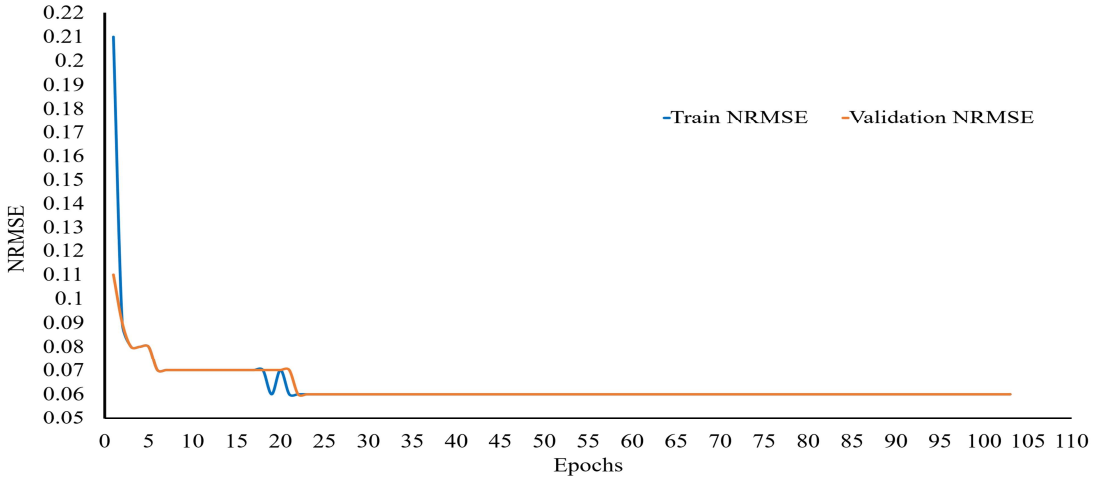


Figure 3.9: Change in NRMSE loss with respect to the number of epochs on the training and validation subsets of the dataset

Figure 3.10: Few original and reconstructed leaf images of peach plants. (Top row) original leaf images. (Bottom row) reconstructed leaf images using the CAE network of the proposed lightweight hybrid model.

The low values of NRMSE loss on training, validation, and test subsets can also be qualitatively verified by observing Figure 3.10, in which the original and reconstructed leaf images have been shown. It can be visualized through Figure 3.10 that reconstructed leaf images are looking very similar to the original leaf images. Hence, it justifies the low value obtained for reconstruction loss (NRMSE loss).

Performance of the proposed lightweight hybrid model on the validation subset of the dataset has been compared with the PlantGhostNet model and five other state-of-the-art CNN architectures with line charts in Figures 3.11 and 3.12.

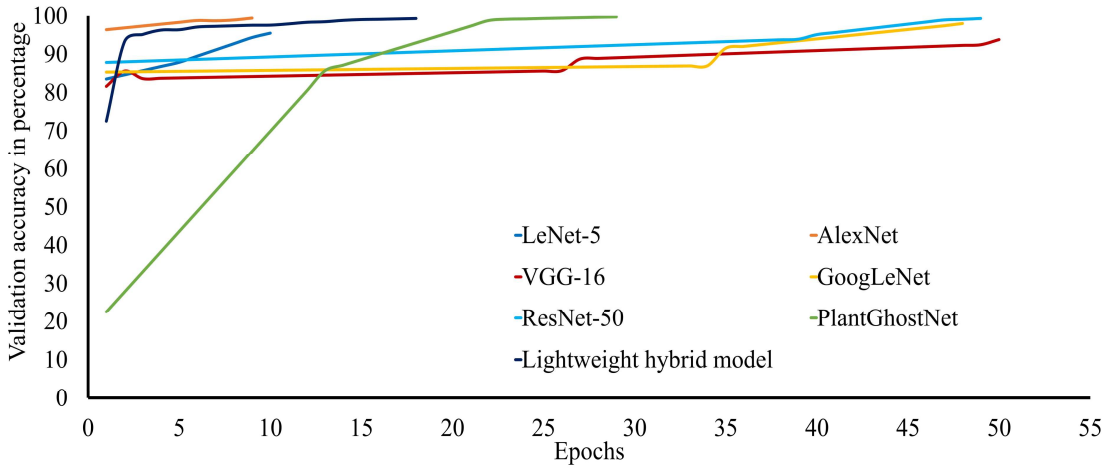


Figure 3.11: Variation of validation accuracy with respect to the number of epochs for the proposed lightweight hybrid model and PlantGhostNet model, along with five state-of-the-art CNN architectures

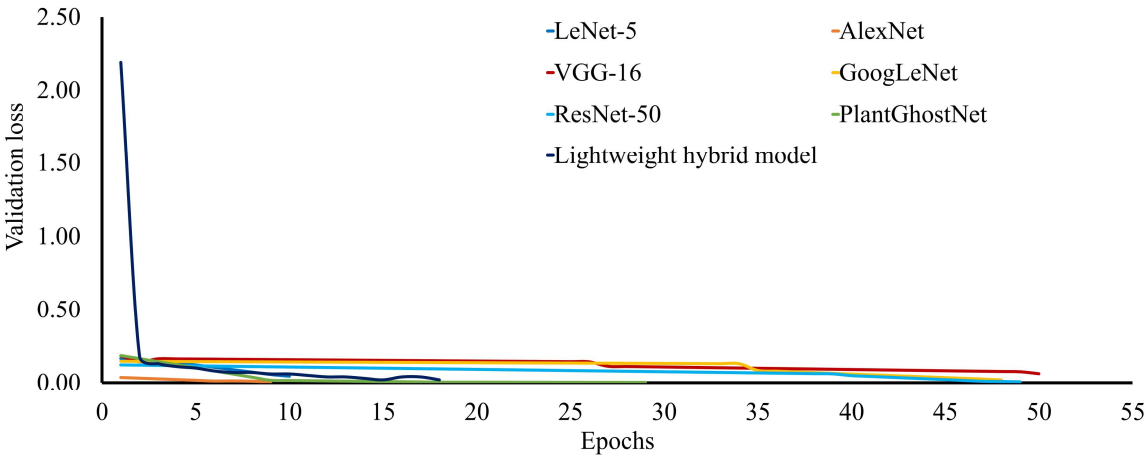
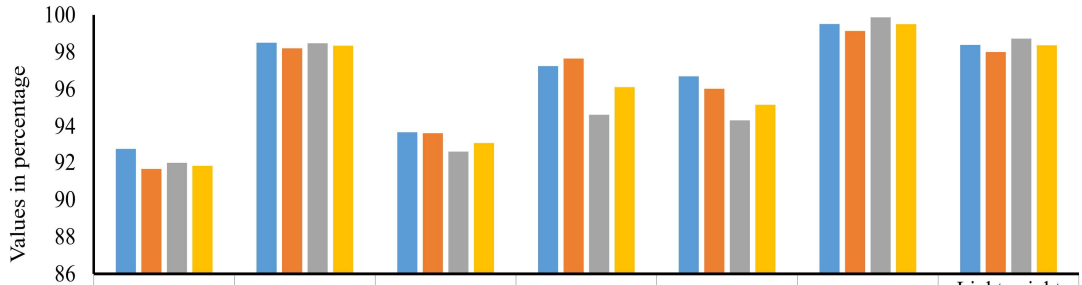


Figure 3.12: Variation of validation loss with respect to the number of epochs for the proposed lightweight hybrid model and PlantGhostNet model, along with five state-of-the-art CNN architectures

It can be observed from Figure 3.11 that the validation accuracy of the VGG-16 model is minimum, i.e., 93.83%, among all other models. The VGG-16 model is a very deep CNN architecture. Thereby, it may suffer from the vanishing gradient problem. Hence, this may be the possible reason behind the VGG-16 model's low performance. The validation accuracy of LeNet-5 and GoogLeNet is 95.53% and 98.02%, respectively. It can also be seen from Figure 3.11 that the accuracies of PlantGhostNet, AlexNet, and ResNet-50 models on the validation subset are slightly higher than the accuracy of the proposed lightweight hybrid model. However, the number of trainable weight parameters utilized by the proposed model is significantly less than other models.

Hence, it can be considered that the proposed lightweight hybrid model can effectively and efficiently identify a plant disease from their digital leaf images. The same fact can also be argued by analyzing the trend of validation loss incurred during the training process of the proposed model and its other counterparts, as shown in Figure 3.12.

In order to investigate the proposed lightweight hybrid model more thoroughly, its performance is evaluated on the test subset and compared with the performances of the PlantGhostNet model and five other state-of-the-art CNN architectures. This performance comparison is shown in Figure 3.11 with the help of four evaluation metrics, namely, accuracy, precision, recall, and f1-measure (Zaki & Wagner, 2020).



	LeNet-5	AlexNet	VGG-16	GoogLeNet	ResNet-50	PlantGhostNet	Lightweight hybrid model
Accuracy	92.78	98.5	93.68	97.24	96.69	99.51	98.38
Precision	91.65	98.2	93.63	97.65	96.02	99.13	98.00
Recall	91.98	98.47	92.58	94.62	94.32	99.87	98.72
F1-measure	91.81	98.33	93.10	96.11	95.16	99.50	98.36

Figure 3.13: Accuracy, precision, recall, and f1-measure of the proposed lightweight hybrid model and PlantGhostNet model, along with five other state-of-the-art CNN architectures.

It can be perceived from Figure 3.11 that LeNet-5 and VGG-16 models have achieved 91.81% and 93.10% f1-measure on the test subset of the dataset, which is the lowest among all other models. Moreover, the performance of ResNet-50, GoogLeNet, and AlexNet in terms of f1-measure are 95.16%, 96.11%, and 98.33%, respectively. It can also be seen from this figure that the f1-measure of the proposed lightweight hybrid model is approximately 1% less than the PlantGhostNet model. However, this model is much lighter than the PlantGhostNet model with respect to the number of trainable weight parameters. As it requires only 9,914 trainable weight parameters, whereas the PlantGhostNet model utilizes roughly 73,301 weight parameters, which is approximately seven times higher than the proposed lightweight hybrid model.

In order to examine the lightweight nature of the proposed models, the number of trainable weight parameters employed in these models are compared with the baseline CNN model along with five other state-of-the-art CNN architectures in Table 3.8. It can be perceived from this table that the proposed lightweight hybrid model requires the least number of trainable weight parameters, i.e., 9,914, among other models. Further, the PlantGhostNet model utilizes roughly 73 thousand trainable weight parameters, and LeNet-5 CNN architecture requires 61 thousand trainable weight parameters. The VGG-16 model uses 138 million trainable weight parameters, which is the maximum among other models. It can also be observed from Table 3.8 that AlexNet and baseline CNN architectures utilized comparable trainable weight parameters. The research work presented in this chapter is also compared in Table 3.9 with existing state-of-the-art research works available in the literature.

Table 3.8: Number of trainable weight parameters used by proposed PlantGhostNet and baseline CNN model along with five state-of-the-art CNN architectures

Model	**Number of trainable weight parameters (approximately)**
LeNet-5	61 thousand
AlexNet	650 thousand
VGG-16	138 million
GoogLeNet	7 million
ResNet-50	25.6 million
Baseline CNN model	765 thousand
PlantGhostNet	**73 thousand**
Proposed lightweight hybrid model	**9,914**

It can be seen from this table that the proposed lightweight hybrid model has outperformed all research works given in Table 3.9 despite of using the least number of trainable weight parameters. Hence, it can be concluded that the proposed models

have several advantages over existing state-of-the-art research works present in the literature.

Table 3.9: Comparison of proposed models with state-of-the-art research works present in the literature

Research Work	**Technique**	**Accuracy**	**Number of trainable weight parameters (approximately)**
(Mohanty, Hughes, & Salathé, 2016)	GoogLeNet	99.34%	7 million
(Zhao, et al., 2021)	DoubleGAN + DenseNet-121	99.7%	8.1 million
Proposed work	**PlantGhostNet**	**99.51%**	**73,301**
	Lightweight hybrid model based on CAE and CNN	**98.38%**	**9,914**

The proposed models have two prominent use cases. First, it can be trained and used for automatic plant disease detection on low-computational powered systems with less training time and prediction time. Second, the proposed models can also be trained and used on smartphones. Running a DL model in mobile applications instead of sending leaf images of plants to the cloud/server reduces the latency and provides data privacy to farmers.

3.5. Chapter Summary

In this chapter, two lightweight CNN models, namely PlantGhostNet and lightweight hybrid model, were proposed to diagnose a plant disease effectively and efficiently using leaf images. The PlantGhostNet model utilized the Ghost Module to reduce the number of trainable parameters, and the Squeeze-and-Excitation Module was used to further enhance the performance of the PlantGhostNet model. The lightweight hybrid model was designed and developed by combining the CAE and CNN models. This model first obtained the compressed domain representations of leaf images using the Encoder network of CAE and then used these compressed domain representations for classification using CNN. Due to the reduction in spatial dimensions of leaf images

using CAE, the number of features, and hence the number of training parameters reduced significantly.

Though the models proposed in this chapter can effectively and efficiently identify a plant disease from leaf images. However, in the real world, different types of diseases can infect a crop, and designing an individual DL model for each disease is costly. Therefore, next chapter of this thesis tries to mitigate the aforementioned problem by designing a novel effective and efficient DL model to identify different diseases of a crop from their digital leaf images.

4. Lightweight and Improved Vision Transformer Model to Detect Multiple Plant Diseases from Leaf Images

This chapter proposes a lightweight and improved ViT model named Trans-Inception Network (TrIncNet) for diagnosing multiple plant diseases with the help of their digital leaf images. The proposed TrIncNet model can diagnose multiple plant diseases very effectively and efficiently as compared to other state-of-the-art models present in the literature. Additionally, in order to enhance the confidence of farmers and agricultural scientists in predictions of the proposed model, human understandable visual explanations are also provided along with the predictions.

4.1. Introduction

The previous chapter of this thesis primarily focuses on identifying a single type of plant disease from leaf images. However, in real-world scenarios, crops can be infected by several types of diseases, and developing an individual DL model for each crop-disease pair is very costly and time-consuming. Therefore, an effective and efficient DL model is required to identify multiple plant diseases. Moreover, human interpretable visual explanations are also required to enhance the trust of farmers and agricultural scientists in the predictions of the model.

In literature, various researchers have utilized either state-of-the-art CNN architectures or customized CNN architectures for diagnosing plant diseases via their symptomatic leaf images (Atila, Uçar, Akyol, & Uçar, 2021; Dhaka, et al., 2021; Tiwari, Joshi, & Dutta, 2021). Some researchers have also applied the ViT model to detect plant diseases automatically (Borhani, Khoramdel, & Najafi, 2022). Despite of high performance of the ViT model, this model suffers from a major drawback that it contains an MLP module in its encoder block, which is computationally expensive as well as inefficient in extracting various temporal and spatial features from leaf images.

Therefore, this chapter proposes a comparatively less computationally expensive ViT model named “TrIncNet” for diagnosing multiple plant diseases. The TrIncNet model comprises of multiple modified encoder blocks named “Trans-Inception blocks,” which

comprise of the Inception module in place of the MLP module for extracting various temporal and spatial features from leaf images. Additionally, skip connections are also added around each Trans-Inception block to make the proposed model more resistant towards the vanishing gradient problem. The proposed TrIncNet model has been trained and tested on two plant disease datasets viz: PlantVillage dataset (Hughes, Salathé, & Mohanty, 2015) and in-field Maize dataset (Haque, et al., 2022) for showcasing the applicability of the proposed model in the real-world scenario. Moreover, the comparative performance analysis of the proposed model has also been done with the existing state-of-the-art models, namely, VGG-19, GoogLeNet, ResNet-50, Xception, InceptionV3, MobileNet, and ViT on both datasets.

Due to the complex and deep nested structure of DL models, these models are considered as black-box. Thus, in order to provide human understandable visual explanations for the predictions of these models, different researchers have proposed various eXplainable Artificial Intelligence (XAI) algorithms. Local Interpretable Model Agnostic (LIME) is an XAI algorithm that can efficiently provide visual explanations for the predictions of any ML or DL model (Ribeiro, Singh, & Guestrin, 2016). In order to enhance the trust of farmers in the predictions of the proposed model, human interpretable visual explanations are also provided with the help of LIME framework.

Rest of the chapter is organized into four sections. Section 4.2 explores and discusses various research works present in the literature on plant disease detection. Section 4.3 describes the proposed TrIncNet model. Section 4.4 discusses the experiments and the obtained results, followed by the chapter summary in section 4.5.

4.2. Related Work

Many research efforts are made in literature to automatically identify plant diseases via their digital leaf images. Earlier, researchers applied different ML techniques (Trivedi, Shamnani, & Gajjar, 2020; Varshney, Babukhanwala, Khan, Saxena, & Singh, 2021) for automatic plant disease identification. Nowadays, researchers are utilizing DL methods, particularly CNNs, to identify plant diseases automatically, as these techniques can extract various spatial and temporal features from images automatically. Haque et al. (Haque, et al., 2022) investigated the effect of the dense layer, global

average pooling layer, and flatten layer on the performance of the InceptionV3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016) model in detecting three types of diseases in maize plants. After experimentation, they concluded that the InceptionV3 model with a global average pooling layer outperformed others by achieving 95.71% accuracy. Nigam et al. (Nigam, et al., 2023) experimented with eight EfficientNet-based CNN architectures to identify Stem Rust, Stripe Rust, and Leaf Rust diseases in wheat plants. They found that EfficientNet-B4 CNN architecture outperformed other architectures with 99.35% testing accuracy.

Some researchers tried to build a lightweight DL model for plant disease diagnosis. Xiang et al. (Xiang, Liang, Sun, Zhang, & Wang, 2021) developed a lightweight network to identify plant diseases. They designed a lightweight CNN model with the help of multiple-size convolutional filters and channel shuffle operation. Their best model achieved 90.6% accuracy and 84.3% f1-measure on the PlantVillage dataset. Haque et al. (Haque, Marwaha, Deb, Nigam, & Arora, 2023) proposed a custom lightweight CNN model for detecting four diseases in maize crops using leaf images obtained from the PlantVillage dataset. Their proposed network worked quite well on the test dataset and obtained 99.1% classification accuracy. Sharma et al. (Sharma, Tripathi, & Mittal, 2023) designed a lightweight DLMC-Net model by using novel collective blocks and passage layers. Moreover, they used depth-wise separable convolution operation to reduce the number of trainable weight parameters. Their proposed DLMC-Net model achieved 93.56%, 92.34%, 99.50%, and 96.56% accuracy in detecting twenty diseases from the leaf images of citrus, cucumber, grapes, and tomato plants, respectively.

In some recent studies, the attention mechanism has also been utilized to enhance the efficacy of different DL frameworks. Karthik et al. (Karthik, et al., 2020) designed a novel Attention based Residual CNN architecture for disease diagnosis in tomato plants and achieved 98% accuracy in detecting ten tomato plant diseases. Chen et al. (Chen, Wang, Zhang, Zeb, & Nanehkaran, 2021) embedded channel and spatial attention modules in the DenseNet CNN architecture and used the depth-wise separable convolution operation in place of the standard convolution operation. They evaluated the applicability of their approach in identifying four diseases of maize plants on their

own collected dataset and PlantVillage dataset. The authors reported in the paper that their model attained 95.86% and 98.5% accuracy on their collected and PlantVillage datasets, respectively. Zhao et al. (Zhao, Sun, Xu, & Chen, 2022) designed the RIC-NET model using Residual and Inception blocks. They used the Convolutional Block Attention Module (CBAM) to enhance the RIC-NET model's performance. Their model identified seventeen types of diseases in potato, corn, and tomato plants with 99.55% accuracy, as claimed by the authors. Li et al. (Li, et al., 2023) designed a novel Muti-Dilated-CBAM-DenseNet (MDCDenseNet) architecture to identify maize plant diseases from the farmlands. Their proposed model attained 98.84% testing accuracy on the maize plant leaf images collected from the agricultural fields of Northeastern Agricultural University, China. Naik et al. (Naik, Malmathanraj, & Palanisamy, 2022) used the Squeeze-and-Excitation CNN (SECNN) to detect five diseases (down curl of a leaf, geminivirus, cercospora leaf spot, yellow leaf disease, and up curl) in Chili plant's leaf images. Their proposed model attained 98.63% and 99.12% accuracy without data augmentation and with data augmentation, respectively. Moreover, they assessed the model's performance on the PlantVillage dataset and found that the SECNN model achieved 99.28% accuracy. Pandey and Jain (Pandey & Jain, 2022) proposed a novel attention-based learning paradigm to improve the CNN model's performance in diagnosing plant diseases from leaf images. Their proposed model achieved 99.93% accuracy on the PlantVillage dataset. Kaya and Gürsoy (Kaya & Gürsoy, 2023) used the MHA operation in the DenseNet-121 CNN architecture to identify plant diseases and achieved 98.17% accuracy on the PlantVillage dataset.

Due to the powerful capabilities of the ViT model in image classification, Thai et al. (Thai, Tran-Van, & Le, 2021) applied the ViT model to identify four types of diseases in the cassava field. They observed that the ViT model outperformed other standard CNN architectures like EfficientNet and ResNet-50 by giving 1% higher accuracy. Another work that utilized the ViT model for plant disease detection was done by Wu et al. (Wu, Sun, & Huang, 2021). They used the ViT model and a novel multi-granularity feature extraction module to identify ten types of tomato plant diseases. As per their paper, the proposed approach outperformed others by achieving 2% higher accuracy. In research work done by Lu et al. (Lu, et al., 2022), a novel ghost-convolutional Transformer model was proposed to detect diseases in grape plants, and

this model attained 98.14% accuracy in identifying eleven diseases of grape plants. Some recent studies have combined the ViT and CNN models to solve various computer vision problems. Si et al. (Si, et al., 2022) designed an Inception-Transformer model for image classification and segmentation tasks. They evaluated their model on the ImageNet and COCO datasets and found that it surpassed other DL models.

Similarly, Bana et al. (Bana, Loya, & Kulkarni, 2022) designed a GAN that utilized the ViT model and Inception module for image colorization. Another research work done by Zhang et al. (Zhang, Wa, Zhang, & Lv, 2022) combined the goodness of ViT and CNN models to design a novel Tranvolution model to diagnose plant diseases automatically. They evaluated their model on the PlantDoc dataset and found that the Tranvolution model outperformed other research works present in the literature by achieving a 50.3% mean average precision score. Although the research works (Bana, Loya, & Kulkarni, 2022; Si, et al., 2022; Zhang, Wa, Zhang, & Lv, 2022) have combined the Inception module with the ViT model, the computationally expensive MLP module present in the ViT model's encoder block has not been removed in any of these research works. Hence, in this chapter, the MLP module is replaced with the Inception module in the encoder block of the ViT model, as it is computationally expensive and inefficient in extracting features from images. The modified encoder block named "Trans-Inception block" is then utilized to design and develop a lightweight and improved ViT model named TrIncNet model for identifying multiple plant diseases from their digital leaf images. Next section of this chapter describes the proposed TrIncNet model.

4.3. Proposed TrIncNet Model for Detecting Multiple Plant Diseases from Leaf Images

The ViT model is a Transformer (Vaswani, et al., 2017) based DL model designed by (Dosovitskiy, et al., 2021) to perform image classification and segmentation tasks. This model comprises of multiple stacked encoder blocks, and each encoder block of the ViT model contains three modules: MHA, Layer Normalization, and MLP modules. The MHA module performs multiple self-attention operations parallelly, through which the model can capture global dependencies between image patches. The Layer-

Normalization module normalizes its previous layer's activations to improve the model's stability and performance. The MLP module comprises of two densely connected layers that extract various features from image patches, which are further utilized for classifying the given images into their respective classes.

As all layers of the MLP module are densely connected to each other, therefore, it requires a huge number of weight parameters to be trained, which makes the ViT model computationally expensive. Moreover, the MLP module is unable to capture the temporal and spatial features of images efficiently and effectively. Hence, a novel TrIncNet model has been designed and developed in this chapter, which conquers these drawbacks of the ViT model.

The TrIncNet model encompasses of multiple linearly connected modified encoder blocks, also known as Trans-Inception blocks, in which the MLP module has been replaced with the Inception module. The reason for using the Inception module in place of the MLP module is that the Inception module performs convolution and max-pooling operations parallelly. Thus, it uses significantly fewer trainable weight parameters than the MLP module. Moreover, it can also extract various spatial and temporal features of images more effectively and efficiently than the MLP module, which can enhance the performance of the model in performing image classification task. Furthermore, each Trans-Inception block of the TrIncNet model is also surrounded by a skip connection, which makes the model much more resistant to the vanishing gradient problem. The architectural design of the TrIncNet model and its Trans-Inception block has been shown in Figure 4.1.

Each Trans-Inception block of the TrIncNet model comprises of three modules: MHA, Layer Normalization, and Inception modules. Out of these three modules, two modules, namely MHA and Layer Normalization modules, are taken from the ViT model's encoder block, and the Inception module is added to the Trans-Inception block in this research work. Since the MHA and Layer Normalization modules are already described in section 2.2.4, therefore in this chapter only the Inception module is described.

The Inception module performs three convolutional operations with 1×1, 3×3, and 5×5 filters and a 3×3 max-pooling operation simultaneously. Therefore, it can extract various spatial and temporal features of leaf images simultaneously with

different receptive fields. The block diagram of the Inception module has been depicted in Figure 4.2. As the Inception module performs convolution and max-pooling operations. Thus, it has various advantages over the MLP module, which are listed below:

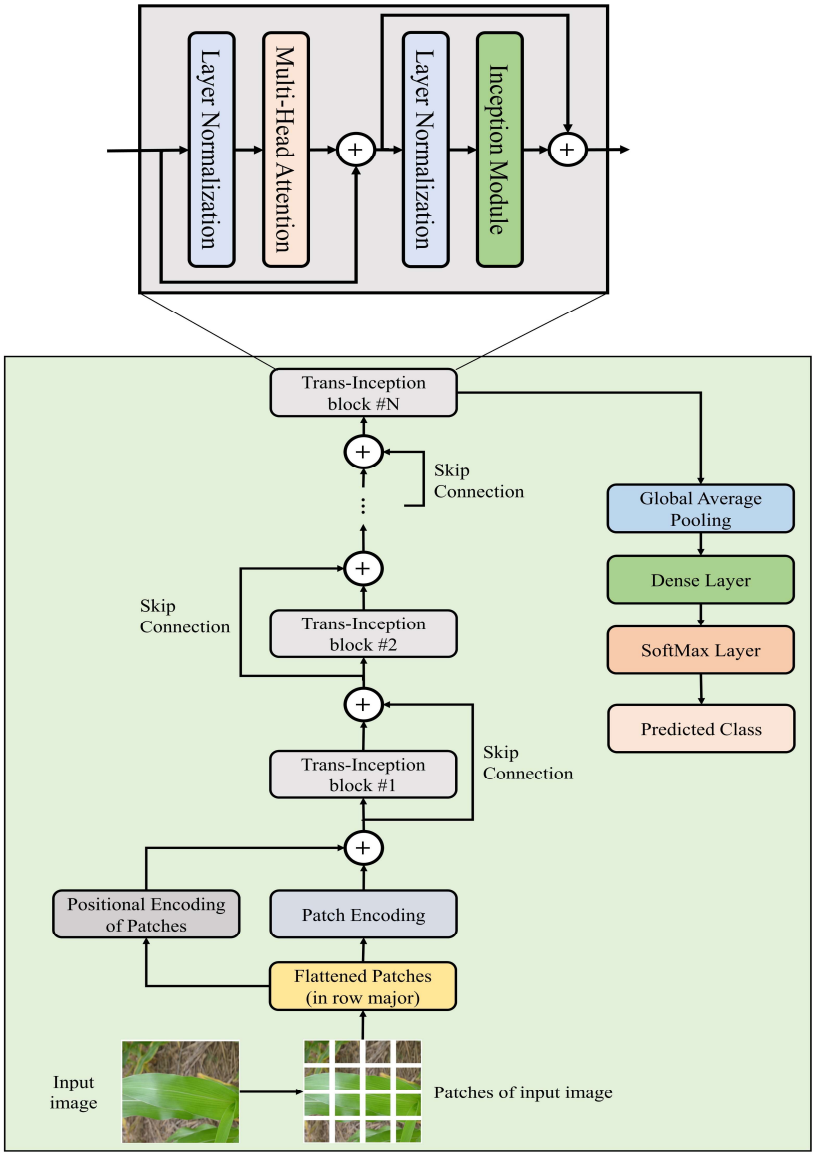


Figure 4.1: Architectural design of the proposed TrIncNet model

- **Spatial invariance (Shift invariance):** It refers to the property of the convolution operation, which makes it able to recognize the object in the image irrespective of its position. The convolution operation holds this property because the convolutional filters move over the entire image.

- **Local translation invariance:** Through this property, the Inception module can recognize the rotated or tilted object in the image. Pooling operation of the Inception module helps to achieve this property.
- **Parameter sharing:** In convolution operation, weight parameters are shared with the help of convolutional filters, and the size of these filters is much lesser than the image size. Hence, the total trainable parameters present in the Inception module are much less than those in the MLP.

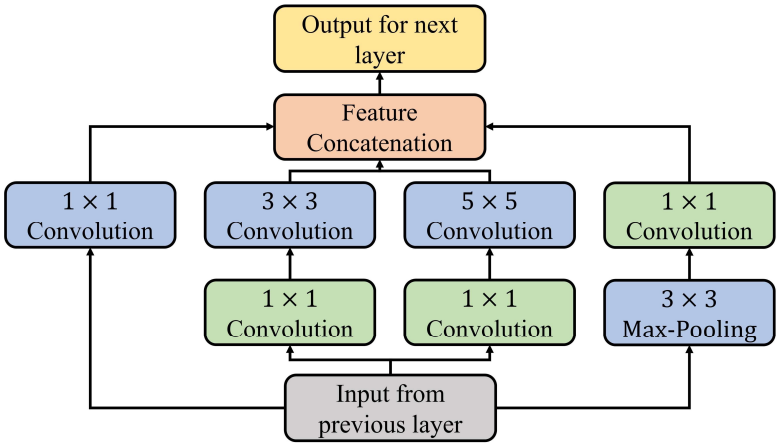


Figure 4.2: Block diagram of the Inception module

In order to analyze the efficiency of the novel Trans-Inception block over the original encoder of the ViT model, asymptomatic analysis has been done on the number of weight parameters used by these blocks. Since, in the Trans-Inception block, the MLP module has been replaced by the Inception module, therefore, the asymptomatic analysis is done only between these modules. Let $I \in \mathbb{R}^{M \times N}$ is the input to the MLP module of the ViT, where M is the number of patches in one leaf image, and N is the size of one embedded patch. As mentioned in (Dosovitskiy et al., 2021), the MLP module present in the encoder block of the ViT model contains two fully connected layers having output sizes $2N$ and N, respectively. Hence, the total number of weight parameters used by the MLP module for one patch of leaf image is $\mathcal{O}(2N \times N + N \times 2N) \Rightarrow \mathcal{O}(N^2)$, asymptomatically. Similarly, for M number patches, total $\mathcal{O}(MN^2)$ weight parameters are used by the MLP module. On the other hand, if F is the maximum number of filters used by any convolution operation of the Inception module, then it requires $\mathcal{O}(\max(M^2, F^2))$ weight parameters asymptomatically

(calculated in Appendix A). The ratio r_{MI} between the number of trainable weight parameters used by MLP and Inception modules can be computed as per equation 4.1, where $X = \max(M, F, N)$.

$$r_{MI} = \frac{\mathcal{O}(MN^2)}{\mathcal{O}(\max(M^2, F^2))} \leq \frac{\mathcal{O}(X^3)}{\mathcal{O}(X^2)} = \mathcal{O}(X) \tag{4.1}$$

The above analysis shows that the proposed Trans-Inception block requires $\mathcal{O}(X)$ times fewer weight parameters to train than the ViT model's encoder block. Furthermore, lesser weight parameters used by any model imply that it would require less training time and inference time. Hence, the TrIncNet model needs a smaller amount of training time and inference time than the ViT model. Next subsection discusses the details of various experiments conducted to evaluate the effectiveness of the proposed TrIncNet model in automatic plant disease detection.

4.4. Experimental Study and Results

The Nvidia DGX Server, which has an Intel® Xeon® CPU with 528 GB RAM and an Nvidia Tesla V100-SXM2 32 GB Graphic Card, is used to conduct the experiments of research work presented in this chapter. Python programming language is used to write the scripts for the experiments; however, any programming language can be used for experimentation.

The TrIncNet model's performance is compared with the ViT model (Dosovitskiy, et al., 2021) and six state-of-the-art CNN architectures: VGG-19 (Simonyan & Zisserman, 2015), GoogLeNet (Szegedy, et al., 2015), ResNet-50 (He, Zhang, Ren, & Sun, 2016), Xception (Chollet F. , 2017), InceptionV3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016), MobileNet (Howard, et al., 2017). The Keras Python library (Chollet F. , 2015) embedded in Tensorflow 2.6.0 has been utilized to develop the TrIncNet and ViT models along with six state-of-the-art CNN architectures. Furthermore, the LIME API version 0.2.0.1 has been used to generate human interpretable visual explanations for the predictions of the proposed TrIncNet model.

This section is divided into three subsections. Subsection 4.4.1 provides the details of datasets that have been utilized for training and evaluating the proposed model and

other counterparts. Subsection 4.4.2 provides the best values of different hyperparameters of the proposed TrIncNet model, and subsection 4.4.3 discusses various experimental results obtained from experimentation.

4.4.1 Dataset Description

Experimentation of the research work presented in this chapter has been conducted on two plant disease detection datasets. First dataset encompasses of leaf images of Maize plants captured from agricultural fields having complex backgrounds. Second dataset is the PlantVillage dataset, which is used as a benchmark dataset for plant disease detection. These datasets are described below:

1. **Maize dataset:** The Maize dataset contains 13,971 leaf images, which were captured from multiple agricultural fields of the Indian Institute of Maize Research, Ludhiana, India. The images are captured non-invasively by maintaining a 25-40 cm distance from the camera device to the affected part of the plant and focused on the top/front view of symptomatic parts of the plant. In this dataset, leaf images of three diseases, i.e., MLB, Banded Leaf and Sheath Blight (BLSB), and Turcicum Leaf Blight (TLB), are present along with the healthy leaf images. Few representative leaf images from each class of the dataset are shown in Figure 4.3.

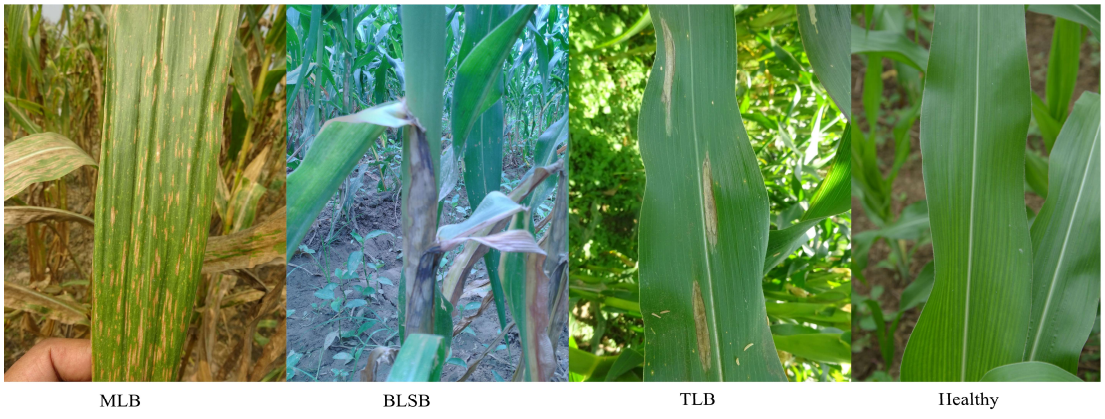


Figure 4.3: Leaf images from each class of the Maize dataset

2. **PlantVillage Dataset:** It is a benchmark dataset used to measure the performance of any ML or DL model for automatically recognizing diseases in plants (Hughes, Salathé, & Mohanty, 2015). This dataset contains 54,503 leaf images of fourteen

plant species categorized into 38 classes. Few representative leaf images from each class of the dataset are shown in Figure 4.4.

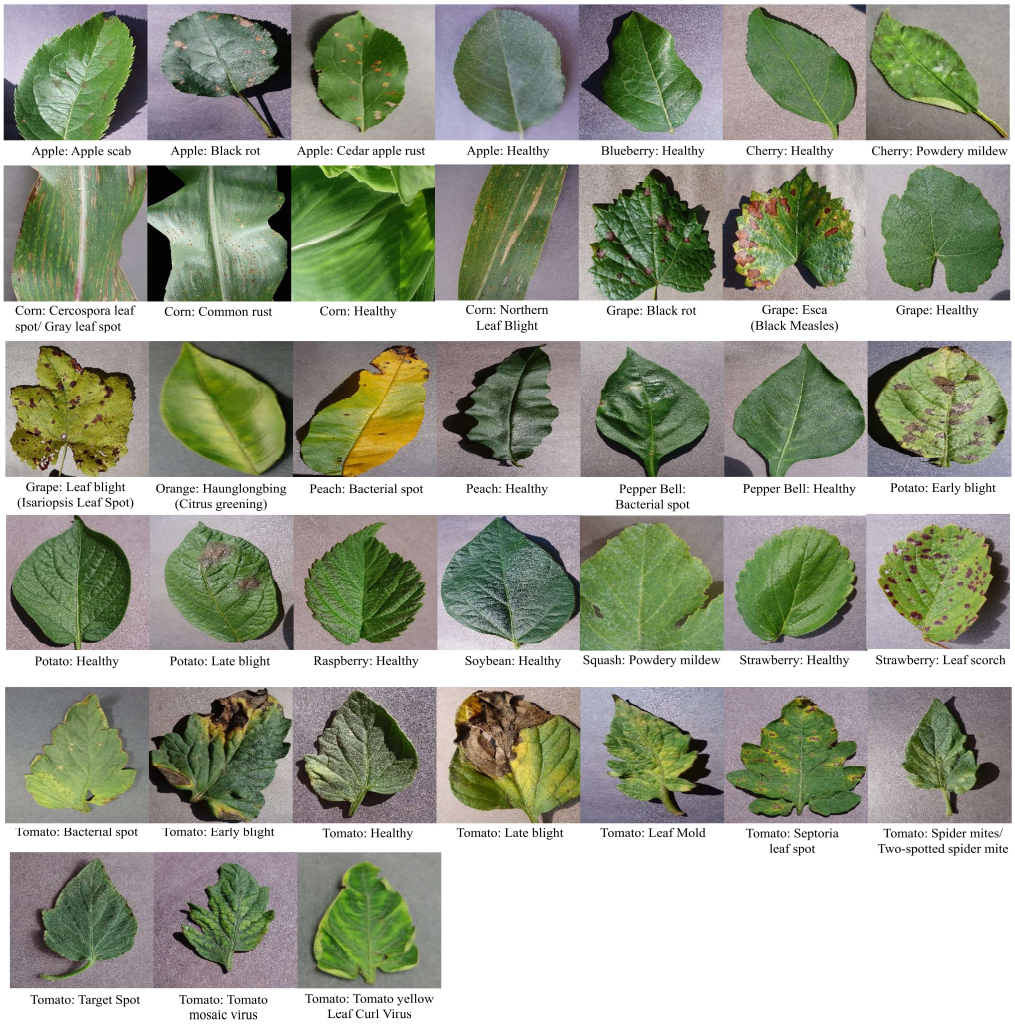


Figure 4.4: Leaf images from each class of the PlantVillage dataset

Since the leaf images present in the Maize dataset are fewer in number and during model training, it can cause model overfitting. Therefore, in order to tackle this problem, the size of the Maize dataset is artificially increased via data augmentation. Data augmentation is a process that increases the dataset's size by applying various image processing techniques like rotation, flipping, etc., (Bedi, Gole, & Agarwal, 2021). After augmentation, the Maize dataset has 100000 leaf images.

The leaf images of both datasets are randomly split into the training, validation, and test subsets as per the 70:15:15 ratio using the scikit-learn library of Python (Pedregosa, Varoquaux, Gramfort, & V., 2011). The training subset's leaf images are utilized to

train the models, and the validation subset is used to adjust the values of hyperparameters so that the best-performing model can be achieved. Finally, the test subset is utilized to measure the TrIncNet model's effectiveness on unseen leaf images.

4.4.2 Hyperparameter Selection

As already discussed, that the performance of the proposed model has been compared with ViT and six other state-of-the-art CNN models. These models are trained for 500 epochs and 32 batch size using the Adam optimizer (Kingma & Ba, 2014) to minimize the categorical cross-entropy loss between the logits and actual labels of leaf images. Early stopping with patience value 20 is used to prevent model overfitting., i.e., if validation accuracy is not improved for twenty consecutive iterations, then model training would stop.

As the TrIncNet model is designed by replacing the MLP model with the Inception module in the encoder block of the ViT model. Therefore, in order to examine the effect of this replacement on the number of weight parameters and performance, ViT and TrIncNet models are implemented using the hyperparameters given in Table 4.1 and Table 4.3, respectively. These values for different hyperparameters of the ViT and TrIncNet models have been derived via extensive experimentation.

Table 4.1: Values of hyperparameters for the ViT model's implementation

Hyperparameter	**Value**
Image size	256×256
Patch size $(p \times p)$	16×16
Size of Embedded Patch (N)	256
Number of Encoder blocks	2
Number of Heads (m)	12
Activation function	SoftMax (Output Layer)
	ReLu (Hidden Layers)
Layer_normalization_rate (epsilon)	10^{-6}

The layer-wise implementation details of ViT and TrIncNet models have been tabulated in Table 4.2 and Table 4.4, respectively.

Table 4.2: Layer-wise implementation details of the ViT model

Layer No.	**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
1	Input layer	256 × 256 × 3	-	256 × 256 × 3	0
2	Patches	256 × 256 × 3	Input layer	256 × 768	0
3	Patch Encoder	256 × 768	Patches	256 × 256	262400
4	Layer Normalization #1	256 × 256	Patch Encoder	256 × 256	512
5	Multi-Head Attention #1	256 × 256	Layer Normalization #1	256 × 256	3155200
6	Add #1	256 × 256, 256 × 256	Multi-Head attention #1 Patch Encoder	256 × 256	0
7	Layer Normalization #2	256 × 256	Add #1	256 × 256	512
8 **MLP module #1**	**Dense #1**	**256 × 256**	**Layer Normalization #2**	**256 × 512**	**131584**
	Dense #2	**256 × 512**	**Dense #1**	**256 × 256**	**131328**
9	Add #2	256 × 256, 256 × 256	Dense #2, Add #1	256 × 256	0
10	Add #3	256 × 256, 256 × 256	Add #2, Patch Encoder	256 × 256	0

Layer No.	**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
11	Layer Normalization #3	256 × 256	Add #3	256 × 256	512
12	Multi-Head attention #2	256 × 256	Layer Normalization #3	256 × 256	3155200
13	Add #4	256 × 256 256 × 256	Multi-Head attention #2 Add #3	256 × 256	0
14	Layer Normalization #4	256 × 256	Add #4	256 × 256	512
15 **MLP module #2**	**Dense #1**	**256 × 256**	**Layer Normalization #4**	**256 × 512**	**131584**
	Dense #2	**256 × 512**	**Dense #3**	**256 × 256**	**131328**
16	Add #5	256 × 256 256 × 256	Dense #4 Add #4	256 × 256	0
17	Global Average Pooling2D #1	256 × 256	Add #5	256	0
18	Dense #5	256	Global Average Pooling2D #1	64	16448
19	Dense #6 (Output layer)	64	Dense #5	4 (for Maize dataset) 38 (for Plant-Village dataset)	262 2470
Total weight parameters				**7117382 (for Maize dataset)** **7119590 (for PlantVillage dataset)**	

Table 4.3: Values of hyperparameters for the proposed TrIncNet model's implementation

Hyperparameter	**Value**
Image size	256×256
Patch size ($p \times p$)	16×16
Size of Embedded Patch (N)	256
Number of Encoder blocks	2
Number of Heads (m)	12
Activation function	SoftMax (Output Layer)
	ReLu (Hidden Layers)
Layer_normalization_rate (epsilon)	10^{-6}

Table 4.4: Layer-wise implementation details of the proposed TrIncNet model

Layer No.	**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
1	Input layer	$256 \times 256 \times 3$	-	$256 \times 256 \times 3$	0
2	Patches	$256 \times 256 \times 3$	Input layer	256×768	0
3	Patch Encoder	256×768	Patches	256×256	262400
4	Layer Normalization #1	256×256	Patch Encoder	256×256	512
5	Multi-Head Attention #1	256×256	Layer Normalization #1	256×256	3155200
6	Add #1	256×256, 256×256	Multi-Head attention #1 Patch Encoder	256×256	0

Layer No.		**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
7		Layer Normalization #2	256 × 256	Add #1	256 × 256	512
8	**Inception Module #1**	**Reshape #1**	**256 × 256**	**Layer Normalization #2**	**16 × 16 × 256**	**0**
		Conv2D #1	**16 × 16 × 256**	**Reshape #1**	**16 × 16 × 96**	**24672**
		Conv2D #2	**16 × 16 × 256**	**Reshape #1**	**16 × 16 × 16**	**4112**
		Conv2D #3	**16 × 16 × 256**	**Reshape #1**	**16 × 16 × 64**	**16448**
		MaxPooling2D #1	**16 × 16 × 256**	**Reshape #1**	**16 × 16 × 256**	**0**
		Conv2D #4	**16 × 16 × 96**	**Conv2D #1**	**16 × 16 × 128**	**110720**
		Conv2D #5	**16 × 16 × 16**	**Conv2D #2**	**16 × 16 × 32**	**12832**
		Conv2D #6	**16 × 16 × 256**	**MaxPooling2D #1**	**16 × 16 × 32**	**8224**
		Concatenate #1	**16 × 16 × 64** **16 × 16 × 128** **16 × 16 × 32** **16 × 16 × 32**	**Conv2D #3** **Conv2D #4** **Conv2D #5** **Conv2D #6**	**16 × 16 × 256**	**0**
		Reshape #2	**16 × 16 × 256**	**Concatenate #1**	**256 × 256**	**0**
9		Add #2	256 × 256, 256 × 256	Reshape #2, Add #1	256 × 256	0
10		Add #3	256 × 256, 256 × 256	Add #2, Patch Encoder	256 × 256	0
11		Layer Normalization #3	256 × 256	Add #3	256 × 256	512

Layer No.		**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
12		Multi-Head attention #2	256 × 256	Layer Normalization #3	256 × 256	3155200
13		Add #4	256 × 256 256 × 256	Multi-Head attention #2 Add #3	256 × 256	0
14		Layer Normalization #4	256 × 256	Add #4	256 × 256	512
15	**Inception Module #2**	**Reshape #3**	**256 × 256**	**Layer Normalization #4**	**16 × 16 × 256**	**0**
		Conv2D #7	**16 × 16 × 256**	**Reshape #3**	**16 × 16 × 96**	**24672**
		Conv2D #8	**16 × 16 × 256**	**Reshape #3**	**16 × 16 × 16**	**4112**
		Conv2D #9	**16 × 16 × 256**	**Reshape #3**	**16 × 16 × 64**	**16448**
		MaxPooling2D #2	**16 × 16 × 256**	**Reshape #3**	**16 × 16 × 256**	**0**
		Conv2D #10	**16 × 16 × 96**	**Conv2D #7**	**16 × 16 × 128**	**110720**
		Conv2D #11	**16 × 16 × 16**	**Conv2D #8**	**16 × 16 × 32**	**12832**
		Conv2D #12	**16 × 16 × 256**	**MaxPooling2D #2**	**16 × 16 × 32**	**8224**
		Concatenate #2	**16 × 16 × 64** **16 × 16 × 128** **16 × 16 × 32** **16 × 16 × 32**	**Conv2D #9** **Conv2D #10** **Conv2D #11** **Conv2D #12**	**16 × 16 × 256**	**0**
		Reshape #4	**16 × 16 × 256**	**Concatenate #2**	**256 × 256**	**0**
16		Add #5	256 × 256 256 × 256	Dense #4 Add #4	256 × 256	0

Layer No.	**Layer Name**	**Input shape**	**Connected to**	**Output shape**	**Parameters**
17	Global Average Pooling2D #1	256 × 256	Add #5	256	0
18	Dense #5	256	Global Average Pooling2D #1	64	16448
19	Dense #6 (Output layer)	64	Dense #5	4 (for Maize dataset) 38 (for Plant-Village dataset)	262 2470
Total weight parameters				**6945574 (for Maize dataset)** **6947782 (for PlantVillage dataset)**	

It can be perceived from Table 4.2 and Table 4.4 that the Inception module present in the TrIncNet model has used 32.67% fewer weight parameters compared to the MLP module present in the ViT model. This results in a 2.41% overall decrement in weight parameters from the ViT model to the TrIncNet model for both datasets. The results obtained during the experimentation of this research work are given in the next section.

4.4.3. Results and Discussion

The proposed TrIncNet model was trained and evaluated on two plant disease detection datasets, viz: Maize and PlantVillage datasets. Performance of the proposed model was evaluated on the validation and test subsets of both the datasets, and comparative analysis was done with the existing ViT model and six state-of-the-art CNN architectures: VGG-19, GoogLeNet, ResNet-50, Xception, InceptionV3, and MobileNet. In this section, first, the results obtained on the Maize dataset are discussed, followed by discussion on the results obtained from the PlantVillage dataset.

A. Results obtained on Maize Dataset

The plot of validation accuracy and validation loss of the TrIncNet model, along with the ViT model and six state-of-the-art CNN architectures for the Maize dataset, has

been depicted in Figure 4.5. and 4.6, respectively. It can be observed from these figures that the proposed TrIncNet model attained the maximum validation accuracy, i.e., 97.0%. Among the other six DL models, the GoogLeNet model attained the second finest results for both validation accuracy and validation loss. Moreover, Xception and InceptionV3 models have achieved comparable accuracies, i.e., 90.38% and 90.23%, respectively. Other DL models that are used for comparison have attained validation accuracies in the range of 73.18% to 91.78%. Same trend can also be observed from Figure 4.6 for the validation loss.

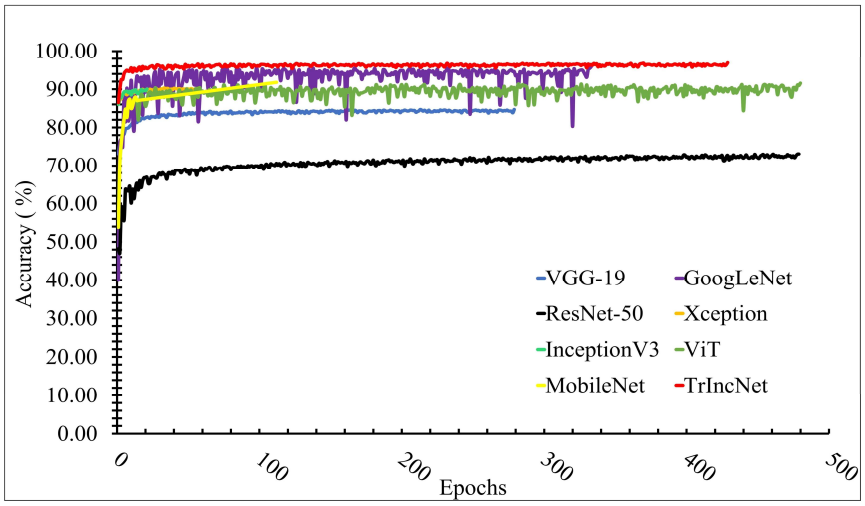


Figure 4.5: Plot of validation accuracies of the TrIncNet model along with the ViT model and the six other state-of-the-art CNN architectures for the Maize dataset

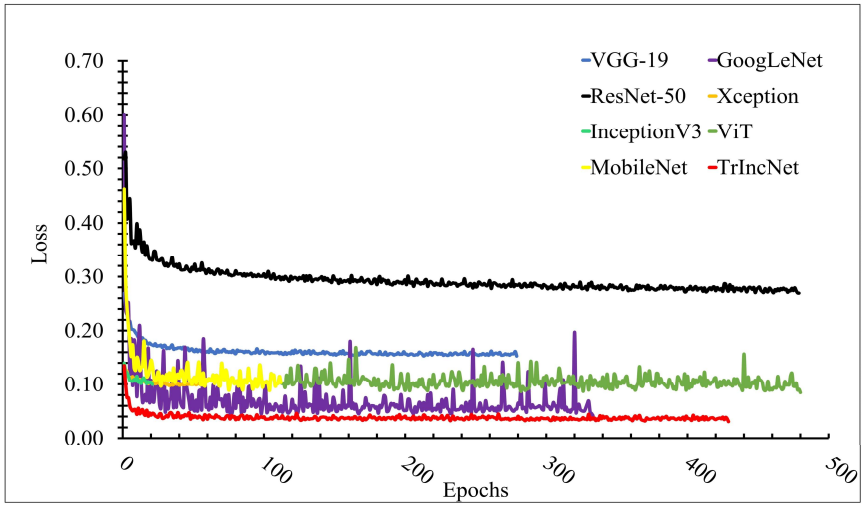
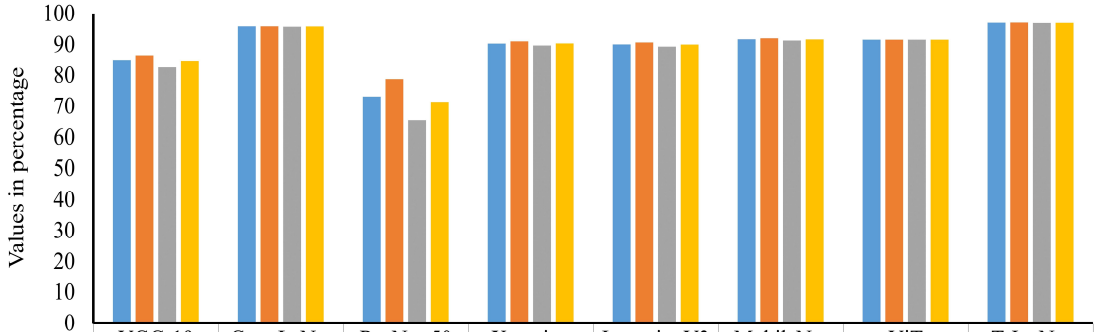


Figure 4.6: Plot of validation losses of the TrIncNet model along with the ViT model and the six other state-of-the-art CNN architectures for the Maize dataset

To study the efficacy of the TrIncNet model more thoroughly, accuracy, precision, recall, and f1-measure are also computed for the TrIncNet model along with the ViT model and six state-of-the-art CNN architectures on the Maize dataset's test subset. These results are given in Figure 4.7. It can be observed from Figure 4.7 that the TrIncNet model achieved the best results for each of the above-mentioned metrics, i.e., 96.93% accuracy, 96.98% precision, 96.83% recall, and 96.9% f1-measure on the Maize dataset. The Xception and InceptionV3 models have attained comparable results, and ResNet-50 has minimum values for the aforementioned metrics. Moreover, GoogLeNet, ViT, MobileNet, and VGG-19 models achieved 95.72%, 91.55%, 91.64%, and 84.46% f1-measure, respectively.



	VGG-19	GoogLeNet	ResNet-50	Xception	InceptionV3	MobileNet	ViT	TrIncNet
Accuracy	84.72	95.8	73.08	90.28	90.03	91.65	91.55	96.93
Precision	86.48	95.79	78.66	90.99	90.65	92.00	91.55	96.98
Recall	82.53	95.65	65.37	89.66	89.28	91.29	91.55	96.83
F1-measure	84.46	95.72	71.40	90.32	89.96	91.64	91.55	96.90

Figure 4.7: Comparison of accuracy, precision, recall, and f1-measure attained by the proposed TrIncNet model along with the ViT model and six state-of-the-art CNN architectures for the Maize dataset

The number of trainable weight parameters utilized by the TrIncNet model, along with the ViT model and six state-of-the-art CNN architectures trained on the Maize dataset, have been compared in Figure 4.8. It is observed from Figure 4.8 that Xception, VGG-19, and InceptionV3 models require a comparable number of trainable parameters, i.e., 20.03 million, 20.87 million, and 21.81 million. Meanwhile, ResNet-50 uses 23.60 million, and the GoogLeNet model uses 8.21 million trainable weight parameters. It can also be observed from Figure 4.8 that the TrIncNet model requires 6.95 million trainable weight parameters, which is 2.41% less than the ViT model, which requires 7.12 million weight parameters. Although the MobileNet model has minimum trainable

weight parameters, i.e., 3.23 million, but it did not perform well as compared to the proposed TrIncNet model.

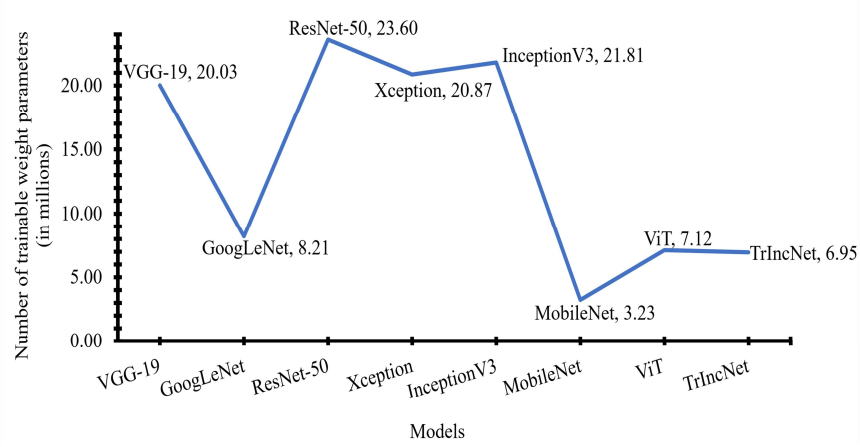


Figure 4.8: Comparison of the number of trainable weight parameters used by the TrIncNet along with the ViT model and six state-of-the-art CNN architectures trained on the Maize dataset

The TrIncNet model's performance on the Maize dataset has been compared in Table 4.5 with the research work done by (Haque, et al., 2022). The reason for comparing it with only this research work is that the TrIncNet model is trained on the same Maize dataset, which was used by (Haque, et al., 2022) in their research work.

Table 4.5: Comparison of the TrIncNet model's performance with a recent research work present in the literature for the identification of Maize plant diseases

Research Work	**Techniques used**	**Number of classes available in the dataset**	**Accuracy**	**Number of trainable weight parameters (In millions)**
(Haque, et al., 2022)	InceptionV3 with Global Average Pooling	4	95.99%	21.78
Proposed Work	**TrIncNet model**	**4**	**96.93%**	**6.95**

It can be perceived from Table 4.5 that the TrIncNet model achieved approximately one percent higher testing accuracy than the research work done by (Haque, et al., 2022) in detecting three diseases (MLB, TLB, and BLSB) of Maize plants under real-field conditions. Moreover, the TrIncNet model requires approximately 68.1% lesser

trainable weight parameters than the research work done by (Haque, et al., 2022). In the following subsection, results obtained on the PlantVillage dataset are discussed.

B. Results obtained on PlantVillage Dataset

The trend of validation loss and validation accuracy with respect to the number of epochs has been analyzed to evaluate the performance of the TrIncNet model with the ViT model and six state-of-the-art CNN architectures on the PlantVillage dataset. This trend of validation accuracy and validation loss have been depicted in Figures 4.9 and 4.10, respectively.

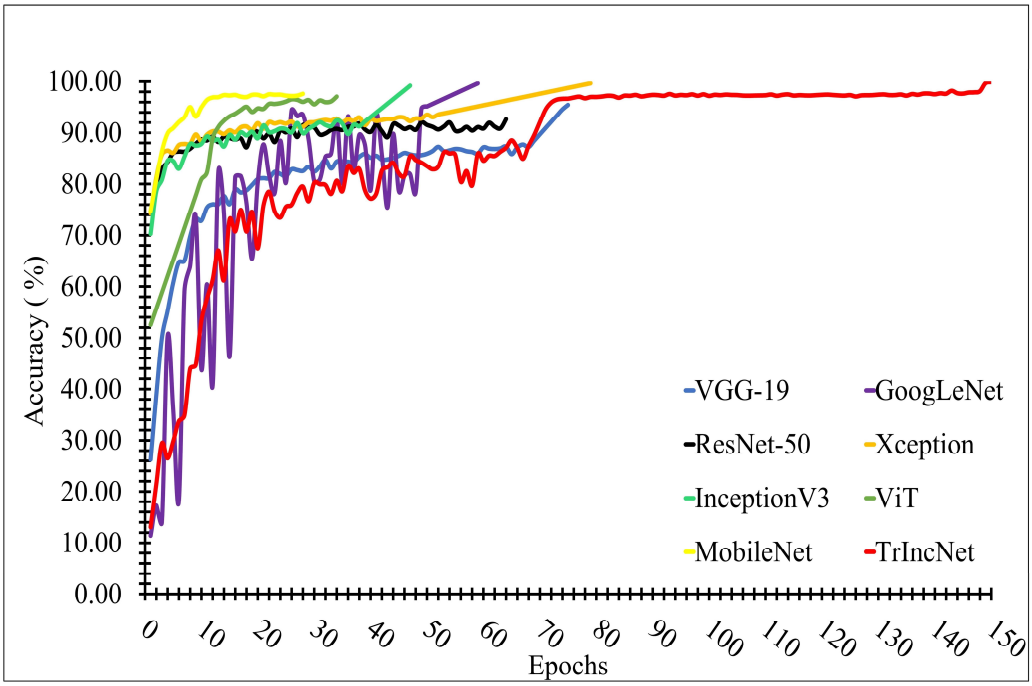


Figure 4.9: Plot of validation accuracies of the TrIncNet model along with the ViT model and the six other state-of-the-art CNN architectures for the PlantVillage dataset

It has been observed by analyzing the plots given in these figures that the Xception, GoogLeNet, and InceptionV3 models have attained comparable accuracies, i.e., 99.76%, 99.78%, and 99.28%. Furthermore, other DL models used for comparison have attained validation accuracies in the range of 92% to 97%. It can also be observed from these figures that the proposed TrIncNet model achieved the highest validation accuracy and lowest validation loss, i.e., 99.95%. Same trend can also be followed in Figure 4.10 for the validation losses of the proposed TrIncNet model and other counterparts.

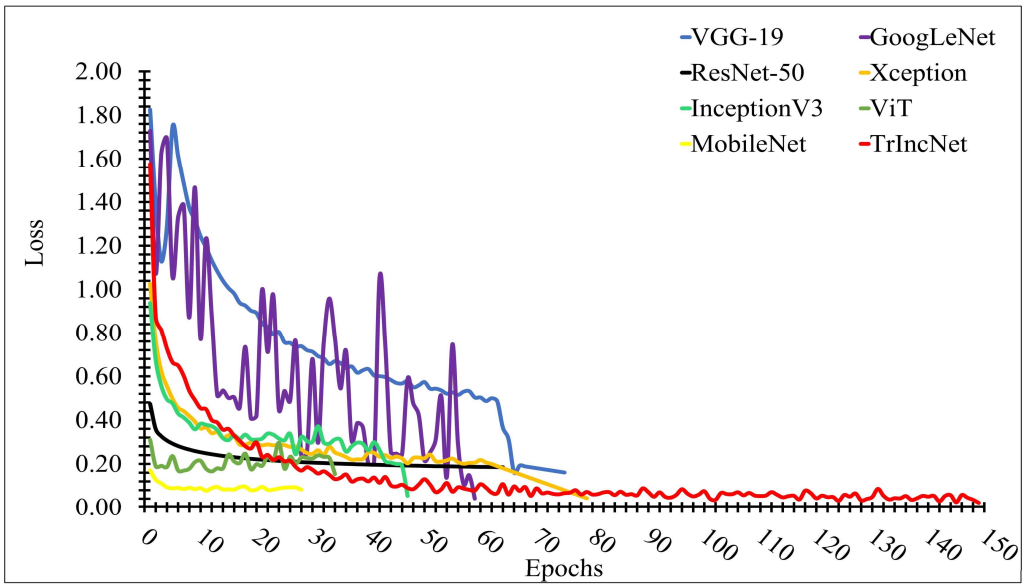
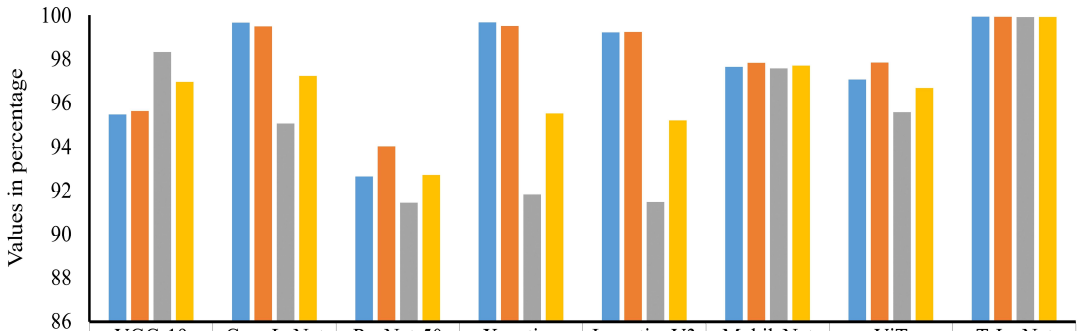


Figure 4.10: Plot of validation losses of the TrIncNet model along with the ViT model and the six other state-of-the-art CNN architectures for the PlantVillage dataset



	VGG-19	GoogLeNet	ResNet-50	Xception	InceptionV3	MobileNet	ViT	TrIncNet
Accuracy	95.46	99.66	92.66	99.67	99.22	97.63	97.06	99.93
Precision	95.61	99.49	94.05	99.51	99.24	97.81	97.82	99.92
Recall	98.34	95.06	91.44	91.81	91.47	97.56	95.56	99.91
F1-measure	96.96	97.22	92.73	95.51	95.20	97.68	96.68	99.91

Figure 4.11: Comparison of accuracy, precision, recall, and f1-measure attained by the proposed TrIncNet model along with the ViT model and six state-of-the-art CNN architectures for the PlantVillage dataset

The performance of TrIncNet model, along with the ViT model and six state-of-the-art CNN architectures, has been analyzed more thoroughly by computing accuracy, precision, recall, and f1-measure on the test subset of the PlantVillage dataset for all models. These results have been compared in Figure 4.11, and it can be observed from this figure that the proposed TrIncNet model has achieved 99.93% accuracy, 99.92% precision, 99.91% recall, and 99.91% f1-measure. Whereas GoogLeNet, VGG-19, ViT,

and MobileNet models achieved 97.22%, 96.96%, 96.68%, and 97.68% f1-measure, respectively.

In order to compare the lightweight nature of the TrIncNet model and ViT model along with six state-of-the-art CNN architectures, the trainable weight parameters of these models have been compared in Figure 4.12 for the PlantVillage dataset. It can be seen by analyzing the line chart given in Figure 4.12 that the ResNet-50 and GoogLeNet models use 8.24 million and 23.67 million trainable parameters, respectively. Meanwhile, the VGG-19, Xception, and InceptionV3 models require comparable weight parameters. Although it can also be observed from Figure 4.12 that the MobileNet model requires minimum trainable weight parameters, i.e., 3.27 million, it did not perform well as compared to the proposed TrIncNet model.

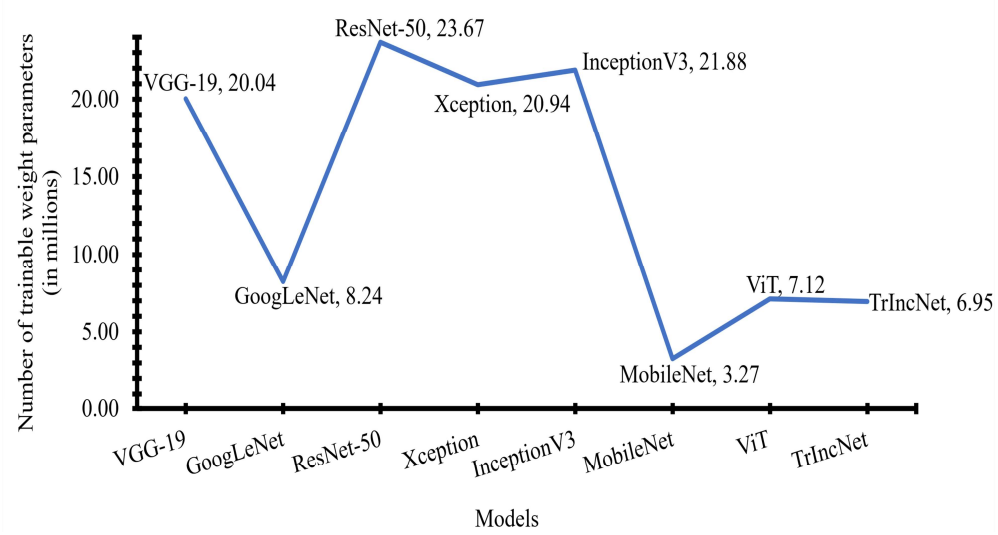


Figure 4.12 Comparison of the number of trainable weight parameters used by the TrIncNet along with the ViT model and six state-of-the-art CNN architectures trained on the PlantVillage dataset

The performance of the TrIncNet model on the PlantVillage dataset has also been compared in Table 4.6 with several recent research works present in the literature in which the PlantVillage dataset is used for model training. It can be observed from Table 4.6 that the proposed model has attained state-of-the-art results by using a significantly lesser number of trainable weight parameters on the PlantVillage dataset as compared to other studies present in the literature.

Table 4.6: Comparison of the TrIncNet model's performance with several recent studies present in the literature on the PlantVillage dataset

Research Work	**Techniques used**	**Number of classes available in the dataset**	**Accuracy**	**Number of trainable weight parameters (In millions)**
(Kaya & Gürsoy, 2023)	Fused-DenseNet-121	38	98.17%	8.13
(Ahmad, Gamal, & Saraswat, 2023)	DenseNet-169	38	99.5%	12.70
(Atila, Uçar, Akyol, & Uçar, 2021)	EfficientNet-B5	38	99.91%	30.56
Proposed Work	**TrIncNet model**	**38**	**99.93%**	**6.95**

Experimental results revealed that the TrIncNet model attained higher testing accuracy than the ViT model. This trend of the results can be argued on the fact that in the Trans-Inception block of the proposed TrIncNet model, the MLP module is replaced with the Inception module, which can effectively and efficiently extract various spatial and temporal features from leaf images. This replacement also reduced the number of trainable weight parameters used by the proposed TrIncNet model, as the Inception module performs convolution and max-pooling operations, which require fewer trainable weight parameters than the fully connected layers present in the MLP module. It can be concluded from the results obtained on both data datasets that the proposed model has achieved higher testing accuracy with a significantly lesser number of trainable weight parameters than the research work done by (Atila, Uçar, Akyol, & Uçar, 2021; Haque, et al., 2022; Ahmad, Gamal, & Saraswat, 2023; Kaya & Gürsoy, 2023). In order to visualize the feature extraction abilities of the MLP module of the ViT model's encoder block and the Inception module of the Trans-Inception block, their extracted features are plotted in Figure 4.13 and Figure 4.14, respectively.

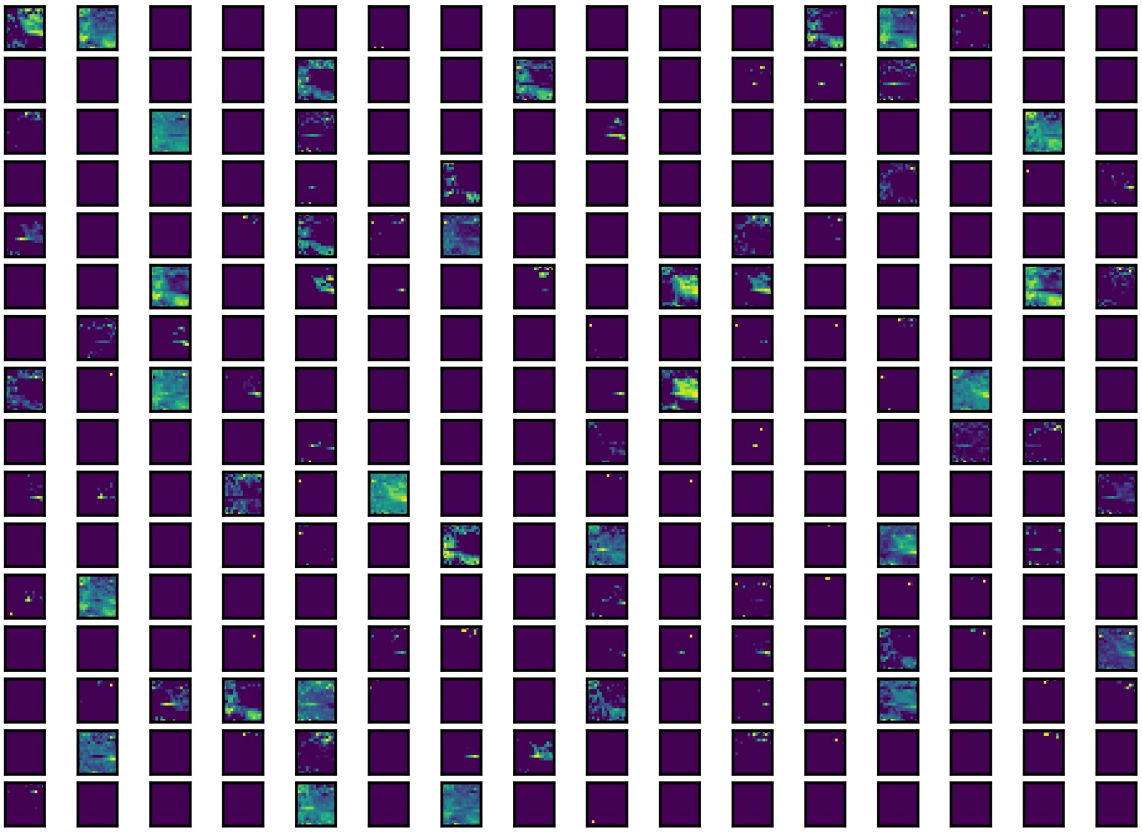


Figure 4.13: Visual representation of features extracted by MLP module present in the ViT model's encoder block

It can be seen from Figure 4.13 that the MLP module present in the ViT model's encoder block is able to capture various features of leaf images. However, these features are very limited (as many feature maps shown in Figure 4.13 are empty) and not very rich in quality because the MLP module has not effectively captured the various spatial and temporal features of leaf images.

On the other hand, the Inception module performs three convolution operations with 1×1, 3×3, and 5×5 filters and a 3×3 max-pooling operation simultaneously. The features extracted by individual operations of the Inception module are represented in Figure 4.14 (a-d), and the concatenation of features extracted by all four operations of the Inception module is shown in Figure 4.14 (e). It can be observed from Figure 4.14 (e) that the features captured by the Inception module are much richer in quality as compared to the MLP module. Moreover, the Inception module has captured more number of features than the MLP module.

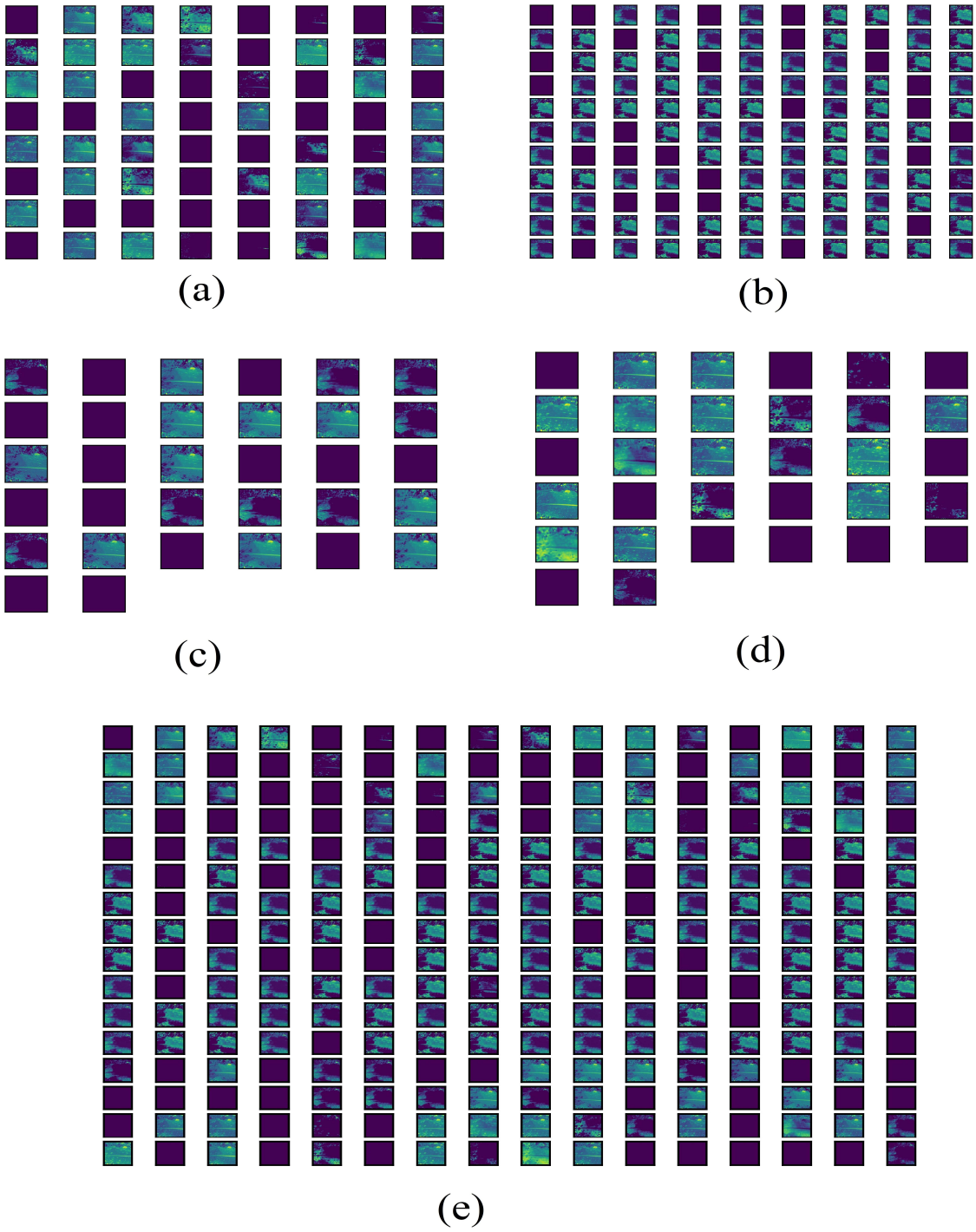


Figure 4.14: Visual representation of features extracted by the Inception module present in the Trans-Inception block. (a) Features extracted by 1×1 convolution operation. (b) Features extracted by 3×3 convolution operation. (c) Features extracted by 5×5 convolution operation. (d) Features extracted by 3×3 max-pooling operation. (e) Concatenation of all features extracted by 1×1, 3×3, 5×5 convolution operations and 3×3 max-pooling operation

Due to the complex and deep nested structure of DL models, these models are considered as black-box. Hence, in order to enhance end-users' trust in the predictions of the proposed TrIncNet model, human interpretable visual explanations are also generated with the help of LIME framework (Ribeiro, Singh, & Guestrin, 2016). This framework provides sample-wise (local) explanations which are human-understandable (interpretable). Since the LIME framework can provide explanations for the predictions of any ML or DL model, therefore this framework is considered as model-agnostic (does not dependent on the model) in nature. It highlights the top n super pixels (collection of neighboring pixels), which favored the predicted class. The value of n can be changed as per user-choice. These visual explanations are easy to understand and provide direct insight into the decision-making process of any ML or DL model. The explanations for the leaf images of Maize and PlantVillage datasets obtained from the LIME framework are shown in Figure 4.15.

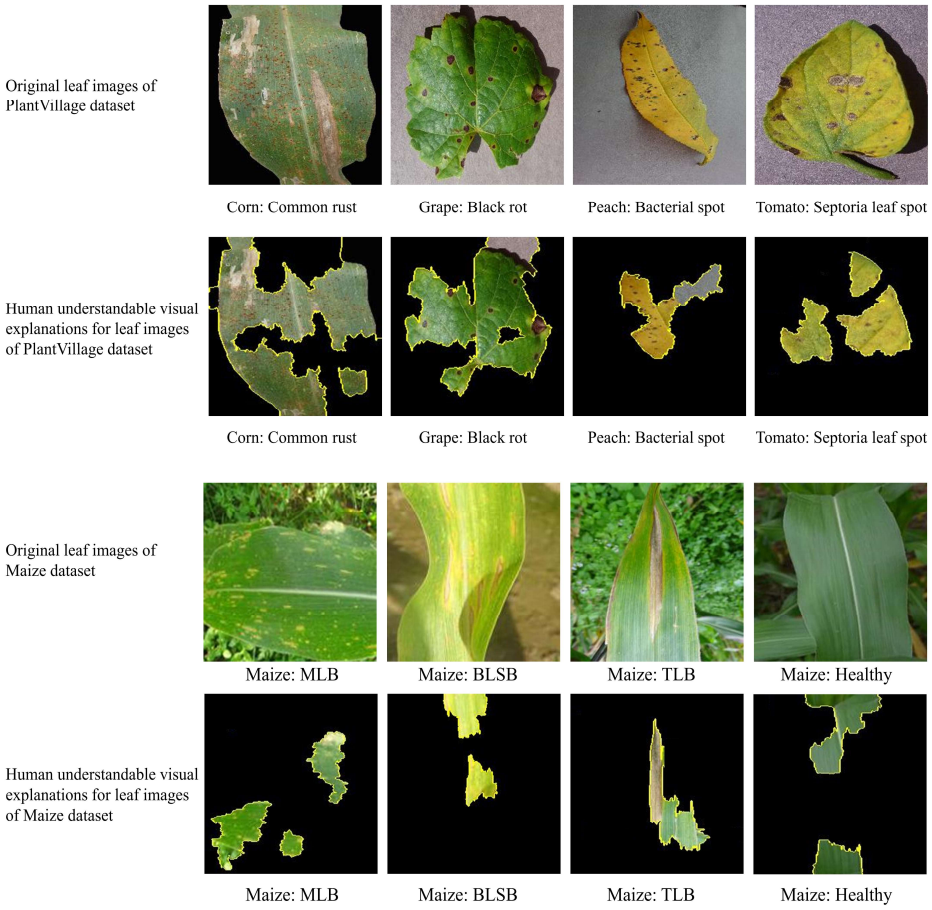


Figure 4.15: Human interpretable visual explanations generated using the LIME framework for the leaf images of Maize and PlantVillage datasets

It can be seen from Figure 4.15 that for diseased leaf images of both the Maize and PlantVillage datasets, the LIME framework has highlighted the diseased lesions of leaf images, and thereby the prediction of the TrIncNet model is validated. On the other hand, the LIME framework highlights the green (healthy) areas of the leaf image when the model predicts the leaf image as healthy.

Conclusively, it can be said that the TrIncNet model proposed in this chapter has the potential to identify multiple plant diseases efficiently and effectively via their digital leaf images captured either from the lab or agricultural fields with high accuracy. Furthermore, the visual explanations of the proposed TrIncNet model obtained from the LIME framework clearly highlight the diseased areas of leaf images, which enhances the trust of farmers and agricultural scientists in the predictions of the proposed model. Hence, due to the high performance and lightweight nature of the proposed TrIncNet model, it can be integrated with different IoT devices to assist farmers in identifying plant diseases at the earliest possible stage.

4.5. Chapter Summary

In this chapter, a lightweight and improved ViT model named TrIncNet has been proposed for identifying multiple plant diseases from their digital leaf images. This model comprised of multiple stacked Trans-Inception blocks, which were designed by replacing the MLP module with the Inception module in the original ViT model's encoder block. Moreover, in the TrIncNet model, each Trans-Inception block was surrounded by a skip connection, which made the proposed model much more resistant to the vanishing gradient problem. During the experimental study, it was found that the TrIncNet model achieved state-of-the-art results as compared to other counterparts despite of utilizing minimum trainable weight parameters.

Although the proposed TrIncNet model can diagnose multiple plant diseases very effectively and efficiently from leaf images. However, in order to recommend the remedy for diagnosed disease, disease severity estimation is also necessary. Hence, the next chapter of this thesis deals with designing and developing an effective and efficient DL model to precisely estimate the severity of the identified disease.

5. Estimating Plant Disease Severity using Convolutional Auto Encoder and Few-Shot Learning

This chapter presents a lightweight Plant Disease Severity Estimation framework named "PDSE-Lite" based on CAE and FSL for estimating the disease severity in plants. By leveraging FSL, the proposed framework requires only a few annotated instances for training, which significantly reduces the human efforts required for data annotation.

5.1. Introduction

Early stage plant disease detection with severity estimation is still a major challenge in front of agrarian researchers as it hampers both food grain quality and quantity. Moreover, plant disease severity estimation is also necessary for tracking plant diseases and treatment planning. Conventionally, farmers and agricultural scientists estimate the severity of plant diseases with their expertise by manually examining the plant leaves. However, nowadays, the severity of plant diseases is estimated by applying different ML and DL techniques on digital leaf images.

In computer vision, the problem of plant disease severity estimation via digital leaf images can be conceptualized in two ways. In the first scenario, disease severity is estimated by classifying the plant leaf image with the help of any ML or DL models into various severity level classes like low, moderate, or severe. In literature, this approach of severity estimation has been utilized by various researchers (Wang, Sun, & Wang, 2017; Haque, et al., 2022). However, classifying leaf images into predefined severity classes instead of estimating disease severity by segmenting infected regions has few drawbacks, such as subjective interpretation and the inability to track disease changes over time.

In the second scenario, the plant disease severity estimation is done by segmenting the diseased regions of leaf images and then calculating the percentage of diseased pixels out of total leaf pixels, i.e., sum of healthy and diseased pixels. The research works based on this approach is broadly divided into three groups. First group of research

works utilizes various Digital Image Processing (DIP) techniques like image thresholding, Otsu segmentation, etc., to segment out the diseased area from leaf images (Bock, Poole, Parker, & Gottwald, 2010; Patil & Bodhe, 2011; Barbedo, 2014; Dhingra, Kumar, & Joshi, 2018). Although these DIP methods can segment the diseased areas from leaf images, but their performance significantly decreases when applied to leaf images captured from the real field with complex backgrounds.

Second category of research works segments diseased areas from plant leaf images by using different ML techniques like Fuzzy C-Means clustering, K-Means clustering, etc., (Biswas, Jagyasi, Singh, & Lal, 2014; Mwebaze & Owomugisha, 2016; Sethy, Negi, Barpanda, Behera, & Rath, 2018). Though the results achieved via ML techniques are much better than the DIP techniques, but they suffer from some major drawbacks. K-means clustering is sensitive to hyperparameter initialization, leading to variable segmentation outcomes. Furthermore, Fuzzy C-Means clustering faces high computational complexity and dependence on the fuzziness parameter, requiring careful parameter selection for accurate results.

The third type of research works have leveraged various DL techniques to segment the diseased areas from leaf images for estimating plant disease severity (Chen, et al., 2021; Wang, et al., 2021; Pal & Kumar, 2023). Nevertheless, training these models requires a large amount of annotated leaf images for precise segmentation of disease areas from leaf images, and in the real world, creating such datasets is a very laborious task. Furthermore, training any DL model with a limited amount of annotated leaf images can lead to model overfitting.

In order to conquer the problem of limited data availability, various researchers have primarily used two types of data augmentation techniques, namely, Digital Image Processing techniques (Haque, et al., 2022; Chohan, Khan, Chohan, Katpar, & Mahar, 2020) and Generative Adversarial Networks (Abbas, Jain, Gour, & Vankudothu, 2021; Zhang, Wa, Zhang, & Lv, 2022). Though these data augmentation techniques can generate an adequate amount of leaf images along with their annotations, but the performance of any model trained on these images drastically decreases when deployed in the real world.

Therefore, the advantages of FSL, which uses few instances for training, can be leveraged to develop a DL model for plant disease severity estimation. The FSL techniques are based on Meta-Learning or Learning to Learn approaches, and these techniques have been described earlier in section 2.3. Various researchers have also leveraged FSL in the agricultural sector for plant disease detection and severity estimation (Argüeso, et al., 2020; Liang X. , 2021; Tassis & Krohling, 2022).

However, most of these works estimate plant disease severity by classifying the leaf image into one of the several predefined severity level classes. To the best of our knowledge, none of the existing research works have estimated the severity level of plant diseases by segmenting the diseased areas with the help of few training instances. Hence, in order to bridge this research gap, a lightweight framework named "PDSE-Lite" based on CAE and FSL is proposed in this chapter for plant disease detection and severity estimation.

Rest of this chapter comprises of four sections. Section 5.2 delves into the pertinent literature related to the research work presented in this chapter. Section 5.3 describes the proposed PDSE-Lite framework. Section 5.4 provides the details of experimentation, and the results obtained from experimentation. Lastly, section 5.5 summarizes the chapter.

5.2. Related Work

Numerous research works have been done in recent years for automatic plant disease severity estimation by utilizing various ML or DL techniques on digital images of plant leaves. Some of these existing research works are discussed in this section.

In the literature, researchers have estimated the severity of plant disease using two approaches. In the first approach, plant disease severity is estimated by classifying the leaf images into one of the predefined severity level classes. These classes are defined with the assistance of plant pathologists. Most of the research works available in the literature are based on this approach have leveraged various predefined CNN models to classify leaf images into various predefined severity level classes. Liang et al. (Liang, et al., 2019) built a PD^{2}SE-Net CNN model by combining ShuffleNetV2 and residual network units for plant disease detection and severity estimation. They trained their

proposed model on a manually annotated PlantVillage dataset (plant pathologists manually divided diseased leaf images into general and serious severity classes) and achieved 90.81% accuracy.

Similarly, Zhao et al. (Zhao, Chen, Xu, Lei, & Zhou, 2021) also trained their proposed model named SEV-Net on manually annotated PlantVillage dataset for plant disease diagnosis with severity estimation. The SEV-Net model was built by adding Spatial and Channel Attention blocks in the existing ResNet-50 CNN architecture, and it achieved 95.37% testing accuracy. Some researchers like (Haque, et al., 2022; Shu, et al., 2023; Dhiman, et al., 2022) trained various state-of-the-art CNN models for classifying plant leaf images into one of the predefined disease severity level classes on their own collected in-field leaf images. Verma et al. (Verma, Chug, Singh, & Singh, 2023) developed a disease severity estimation framework named PDS-MCNet for early and late blight diseases in tomato plants. They first captured digital photographs of several infected and healthy tomato plants. Thereafter, these captured leaf images are manually categorized into three severity classes, namely, Early, Middle, and Late, with the assistance of agricultural scientists. The MobileNetV2 CNN model was utilized in this research work, and it achieved 94.47% accuracy.

Estimating plant disease severity via the first approach, i.e., classifying plant leaf images into predefined severity level classes, has few drawbacks, such as subjective interpretation and the inability to track disease changes over time. Thus, in order to conquer these drawbacks, some researchers tried to estimate plant disease severity by segmenting the infected regions of diseased leaf images. In literature, this approach was followed in several research works like (Wspanialy & Moussa, 2020; Wang, et al., 2021; Ji & Wu, 2022; Divyanth, Ahmad, & Saraswat, 2023). These research works utilized U-Net and DeepLab-based image segmentation models for segmenting diseased and healthy pixels from infected leaf images to further compute disease severity as the percentage of diseased pixels present out of total leaf pixels, i.e., sum of healthy and diseased pixels. Pal and Kumar (Pal & Kumar, 2023) built a novel AgriDet model by using the Inception-VGG Network model along with Kohonen Learning for plant disease detection, and it achieved 96% validation accuracy, as claimed in the paper. Furthermore, the Multi-Variate-Grabcut algorithm was also utilized for

segmenting the diseased lesions from leaf images. Thereafter, the percentage of diseased pixels out of total leaf pixels was calculated to measure plant disease severity.

Despite of high performance exhibited by aforementioned research works in segmenting diseased areas from leaf images, their training process requires a huge amount of labeled leaf images to precisely segment the disease areas from leaf images. However, in real-world scenarios, creating such datasets is a very challenging and laborious task. Additionally, when training an ML or DL model using limited labeled leaf images, there is a high risk of model overfitting. To address this issue, researchers have employed two types of data augmentation techniques, namely, DIP techniques and GANs. These techniques alleviate the problem by artificially generating leaf images with their corresponding segmentation masks. However, the performance of models trained on these artificially generated images significantly decreases when they are deployed in the real world.

Therefore, various researchers have designed and developed various DL models by leveraging the advantages of FSL in which only few annotated leaf images are required for training the model. Pan et al. (Pan, et al., 2022) proposed a two-stage severity estimation framework for leaf scorch disease of strawberry plants using FSL. In the first phase, they utilized the faster RCNN segmentation model to segment strawberry leaves from the captured image, encompassing of other objects like mud, plant stems, etc. Afterward, they applied the Siamese Network to classify the leaf images into either healthy, serious scorch, or general scorch severity levels. In order to assess their proposed framework on unseen data, they evaluated the model's performance on 60 new strawberry plant leaf images. They claimed that their framework achieved 88.33% accuracy on these images.

Tassis and Krohling (Tassis & Krohling, 2022) presented a case study on two FSL techniques, i.e., triplet networks and prototypical networks, which were applied to estimate disease severity in coffee plant leaves. Moreover, they reported that these FSL techniques achieved 93.25% accuracy in classifying coffee plant leaf images into one of the five severity levels: Very High, High, Low, Very Low, Healthy. Although, (Tassis & Krohling, 2022; Pan, et al., 2022) developed a state-of-the-art framework via

FSL for estimating the severity of plant diseases, but there is still a scope for performance improvement in the aforementioned frameworks.

Conclusively, it can be observed from the above discussion that most of the aforementioned research works are focused on estimating plant disease severity via classifying diseased leaf images into one of the predefined severity levels. Though few works also focus on plant disease severity estimation via segmenting diseased areas of infected leaf images, but they necessitate a huge amount of annotated leaf images for model training so that it could generalize well on new leaf images. However, annotating huge number of leaf images is a very challenging and laborious task. Thus, the objective of the research work presented in this chapter is to design and develop a cost-effective framework for automatically estimating the severity of plant diseases by segmenting diseased areas with the help of few training samples. Hence, a novel few-shot and lightweight framework named "PDSE-Lite" based on CAE and FSL has been proposed in the research work presented in this chapter for diagnosing plant diseases automatically and estimating the severity of identified disease by segmenting infected regions of leaf images. As the proposed framework leveraged the advantages of FSL, and thus, it utilizes only few training samples for training. Hence, in this way, the proposed framework significantly reduces the human efforts required for annotating leaf images. Next section of this chapter describes the proposed PDSE-Lite framework.

5.3. Proposed PDSE-Lite Framework for Plant Disease Severity Estimation

This chapter proposes a few-shot and lightweight framework named "PDSE-Lite" based on CAE and FSL for automatic plant disease detection and severity estimation. The PDSE-Lite framework's flow diagram has been given in Figure 5.1. The motivation to build such type of framework comes from the hypothesis that if a DL model (i.e., CAE) can reconstruct leaf images from original leaf images with minimal information loss, then leveraging its learned knowledge will enable the development of DL models for detecting plant diseases and segmenting disease areas from leaf images using limited training data.

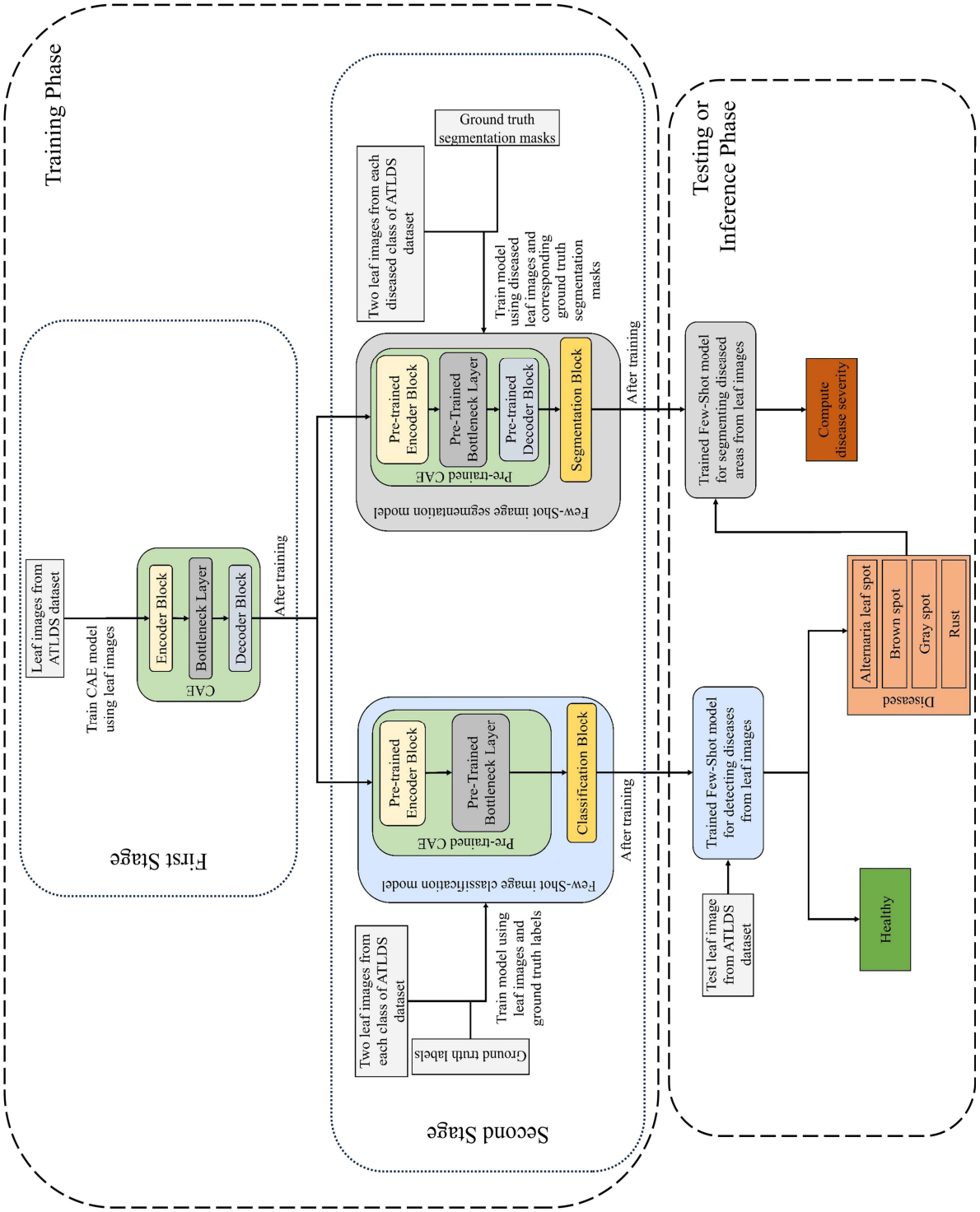


Figure 5.1: Flow diagram of proposed PDSE-Lite framework's training and testing phase

Thus, in order to design the PDSE-Lite framework, first, a lightweight CAE model has been designed and trained to reconstruct the leaf images from original leaf images with minimum reconstruction loss. Thereafter, in the second stage, a few-shot image classification and segmentation models are built by utilizing the pre-trained layers of the CAE model to detect plant diseases and segment diseased areas from leaf images, respectively.

The details of these models have been provided in subsections 5.3.1, 5.3.2, and 5.3.3, respectively. After training the few-shot image classification and segmentation models, these models are utilized in the testing or inference stage to detect plant diseases, segment diseased lesions of infected leaf images, and estimate the severity of identified diseases.

5.3.1. Lightweight Convolutional Auto Encoder (CAE)

First stage of the proposed PDSE-Lite framework focuses on learning to reconstruct the leaf images from original leaf images with minimum reconstruction loss, and this learning has been done via training a lightweight CAE model. CAE is a type of Auto Encoder that effectively and efficiently deals with image data as compared to other types of Auto Encoders (Bedi & Gole, 2021). Like other Auto Encoders, CAE also encompasses of one encoder block, bottleneck layer, and decoder block.

The encoder block of CAE captures different spatial features of leaf images with the help of multiple convolutional and downsampling (max-pooling) layers and encodes them to a compressed domain representation. This compressed domain representation is stored in the bottleneck layer of CAE, and it comprises of all essential features that are further used by the decoder block of CAE to reconstruct leaf images with minimum reconstruction loss.

The decoder block of CAE comprises of same number of layers as of encoder block but in reverse order. Moreover, upsampling layers are utilized instead of downsampling (max-pooling) layers in the decoder block of the CAE model. The layer-wise implementation details of the CAE model utilized in PDSE-Lite framework have been tabulated in Table 5.1, and its architectural diagram has been given in Figure 5.2.

Table 5.1: Implementation details of the CAE model used in the proposed PDSE-Lite framework

Layer No.		Layer Name	Input Shape	Connected to	Output Shape	Number of parameters
1		Input Layer	(256×256×3)	-	(256×256×3)	0
2	Encoder block of CAE	Conv2D #1	(256×256×3)	Input Layer	(256×256× 16)	448
3		MaxPool2D #1	(256×256× 16)	Conv2D #1	(128×128× 16)	0
4		Conv2D #2	(128×128× 16)	MaxPool2D #1	(128×128×8)	1160
5		MaxPool2D #2	(128×128×8)	Conv2D #2	(64×64×8)	0
6		Conv2D #3	(64×64×8)	MaxPool2D #2	(64×64×8)	584
7		MaxPool2D #3	(64×64×8)	Conv2D #3	(32×32×8)	0
8	Bottleneck layer of CAE	Conv2D #4	(32×32×8)	MaxPool2D #3	(32×32×8)	584
9	Decoder block of CAE	UpSample2D #1	(32×32×8)	Conv2D #4	(64×64×8)	0
10		Conv2D #5	(64×64×8)	UpSample2D #1	(64×64×8)	584
11		UpSample2D #2	(64×64×8)	Conv2D #5	(128×128×8)	0
12		Conv2D #6	(128×128×8)	UpSample2D #2	(128×128×8)	584
13		UpSample2D #3	(128×128×8)	Conv2D #6	(256×256×8)	0
14	Output Layer of CAE	Conv2D #7	(256×256×8)	UpSample2D #3	(256×256×3)	219
Total weight parameters						4163

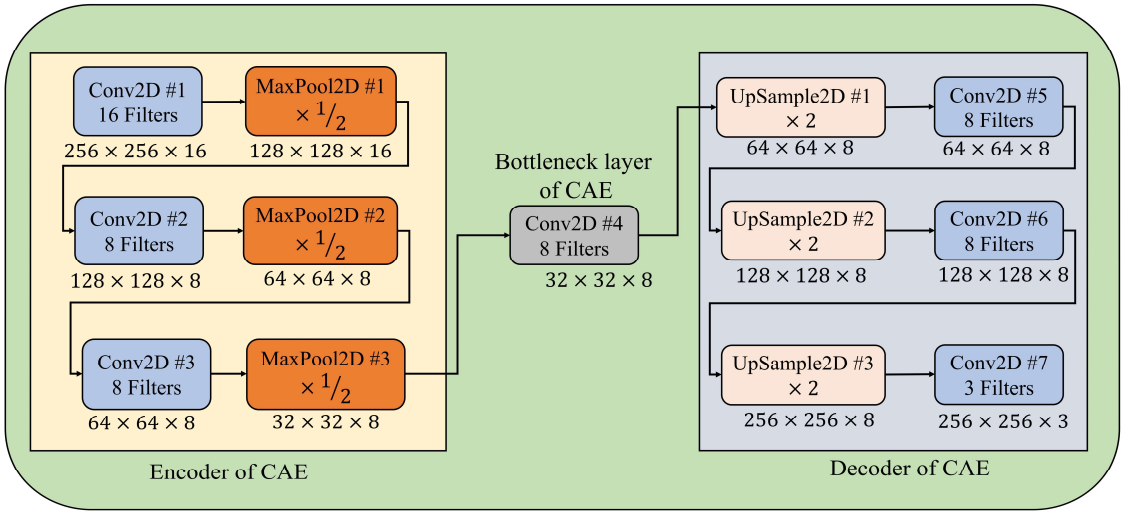


Figure 5.2: Architectural design of the CAE model used in the PDSE-Lite framework

The CAE model's encoder block used in this chapter comprises of three convolutional layers, each succeeded by a max-pooling layer that decreases the feature map's spatial dimension via a factor of two. Similarly, the decoder block of the CAE model also encompasses of three convolutional layers, each preceded by an upsampling layer, which increases the feature map's spatial dimension via a factor of two.

The following subsection of this chapter describes the few-shot image classification model of the proposed PDSE-Lite framework.

5.3.2. Few-Shot Image Classification Model for Detecting Diseases from Leaf Images

During the second stage of the PDSE-Lite framework, a few-shot image classification model is designed and developed to identify diseases in plants by using their leaf images. This model encompasses of a pretrained encoder block and bottleneck layer of the CAE model discussed in the previous subsection. Furthermore, a classification block is appended ahead of the CAE's pretrained bottleneck layer in order to fine-tune this model for plant disease detection. The classification block comprises of two convolutional layers, one max-pooling, global-average-pooling, and dense layers. The implementation details of this model are given in Table 5.2, and its architectural design is depicted in Figure 5.3.

Table 5.2: Implementation details of PDSE-Lite framework's few-shot image classification model used for detecting diseases from leaf images

Layer No.		Layer Name	InputShape	Connected to	Output Shape	Number of parameters
1		Input Layer	(256×256×3)	-	(256×256×3)	0
2	Layers from the Encoder block of CAE	Conv2D #1	(256×256×3)	Input Layer	(256×256× 16)	448
3		MaxPool2D #1	(256×256× 16)	Conv2D #1	(128×128× 16)	0
4		Conv2D #2	(128×128× 16)	MaxPool2D #1	(128×128×8)	1160
5		MaxPool2D #2	(128×128×8)	Conv2D #2	(64×64×8)	0
6		Conv2D #3	(64×64×8)	MaxPool2D #2	(64×64×8)	584
7		MaxPool2D #3	(64×64×8)	Conv2D #3	(32×32×8)	0
8	Bottleneck layer of CAE	Conv2D #4	(32×32×8)	MaxPool2D #3	(32×32×8)	584
9	Extra Layers added to the few-shot image classification model	Conv2D #8	(32×32×8)	Conv2D #4	(32×32×16)	1168
10		MaxPool #4	(32×32×16)	Conv2D #8	(16×16×16)	0
11		Conv2D #9	(16×16×16)	MaxPool #4	(16×16×32)	4640
12		GlobalAveragePooling2D	(16×16×32)	Conv2D #9	32	0
13		Dense Layer (SoftMax)	32	GlobalAverage Pooling2D	5	165
Total weight parameters						**8749**

This model has been trained on few training instances, and during training the categorical cross-entropy loss (denoted by $Loss_{CCE}$) between predicted labels Y_{pred} and actual labels Y_{true} of leaf images has been minimized. The mathematical formula of categorical cross-entropy loss, i.e., $Loss_{CCE}$ is shown in equation 5.1, where N

represents the number of instances taken into account, Y_{pred}^{i} denotes the predicted label of i^{th} instance, and Y_{true}^{i} represents the actual label of i^{th} instance.

$$Loss_{CCE} = -\frac{1}{N}\sum_{i=1}^{N} Y_{true}^{i} \log(Y_{pred}^{i}) \tag{5.1}$$

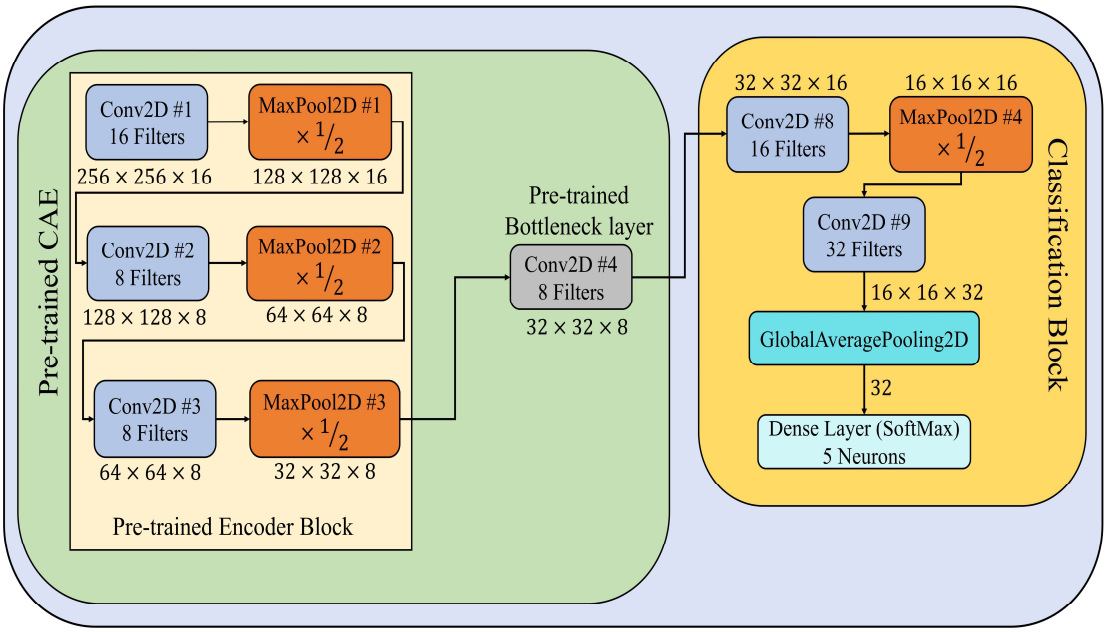


Figure 5.3: Architectural design of few-shot image classification model used for detecting diseases from leaf images

Subsequent subsection describes the few-shot image segmentation model of the proposed PDSE-Lite framework.

5.3.3. Few-Shot Image Segmentation Model for Segmenting Diseased Areas from Diseased Leaf Images

In order to estimate the severity of detected plant disease, a few-shot image segmentation model has been designed and implemented for segmenting disease areas from leaf images. This model encompasses of pretrained CAE (described in subsection 5.3.1) and segmentation block. In the segmentation block, first, the output features maps of pretrained bottleneck, Conv2D #5, and Conv2D #6 layers are upsampled by a factor of 8, 4, and 2, respectively. Thereafter, these up-sampled feature maps have been concatenated channel-wise. By this concatenation, all features extracted by different

convolutional layers of the CAE model's decoder block are merged to form a combined feature map.

Subsequently, this combined feature map is passed to three stacked convolutional layers having 12, 6, and 3 filters, respectively. The last convolutional layer, which has three filters, acts as an output layer that generates the segmentation mask for a given leaf image. Each pixel of this segmentation mask can have either of three values, i.e., 0 is for the background, 1 is for healthy pixels, and 2 is for diseased pixels. The layer-wise implementation details of this model are tabulated in Table 5.3, and its architectural diagram is given in Figure 5.4.

Table 5.3: Layer-wise implementation details of the PDSE-Lite framework's image segmentation model used to segment diseased areas from leaf images

Layer No.		Layer Name	Input Shape	Connected to	Output Shape	Number of parameters
1		Input Layer	(256×256×3)	-	(256×256×3)	0
2	Layers from the Encoder block of CAE	Conv2D #1	(256×256×3)	Input Layer	(256×256× 16)	448
3		MaxPool2D #1	(256×256× 16)	Conv2D #1	(128×128× 16)	0
4		Conv2D #2	(128×128× 16)	MaxPool2D #1	(128×128×8)	1160
5		MaxPool2D #2	(128×128×8)	Conv2D #2	(64×64×8)	0
6		Conv2D #3	(64×64×8)	MaxPool2D #2	(64×64×8)	584
7		MaxPool2D #3	(64×64×8)	Conv2D #3	(32×32×8)	0
8	Bottleneck layer of CAE	Conv2D #4	(32×32×8)	MaxPool2D #3	(32×32×8)	584
9	Layers from the Decoder block of CAE	UpSample2D #1	(32×32×8)	Conv2D #4	(64×64×8)	0
10		Conv2D #5	(64×64×8)	UpSample2D #1	(64×64×8)	584
11		UpSample2D #2	(64×64×8)	Conv2D #5	(128×128×8)	0
12		Conv2D #6	(128×128×8)	UpSample2D #2	(128×128×8)	584

Layer No.		**Layer Name**	**Input Shape**	**Connected to**	**Output Shape**	**Number of parameters**
13	Extra Layers added to the image segmentation model	UpSample2D #4	(32×32×8)	Conv2D #4	(256×256×8)	0
14		UpSample2D #5	(64×64×8)	Conv2D #5	(256×256×8)	0
15		UpSample2D #6	(128, 128, 8)	Conv2D #6	(256×256×8)	0
16		Concatenate	(256×256×8) (256×256×8) (256×256×8)	UpSample2D #4 UpSample2D #5 UpSample2D #6	(256×256×24)	0
17		Conv2D #10	(256×256×24)	Concatenate	(256×256×12)	2604
18		Conv2D #11	(256×256×12)	Conv2D #10	(256×256×6)	654
19		Conv2D #12 (1 × 1 filter size) (Output layer)	(256×256×6)	Conv2D #11	(256, 256, 3)	21
Total weight parameters						**7223**

This few-shot model for segmenting infected areas from leaf images has been trained on few leaf images from diseased classes of the ATLDS dataset. This model also minimizes the sum of categorical cross-entropy loss ($SegLoss_{CCE}$) and jaccard loss ($SegLoss_{jaccard}$) between predicted and ground truth segmentation masks during training. The mathematical formulas for $SegLoss_{CCE}$ and $SegLoss_{jaccard}$ are given in equations 5.2 and 5.3, respectively. In these equations, Y^i_{pmask} and Y^i_{gmask} represents predicted and ground truth segmentation masks for i^{th} leaf image, respectively. Furthermore, N denotes the number of instances taken into consideration.

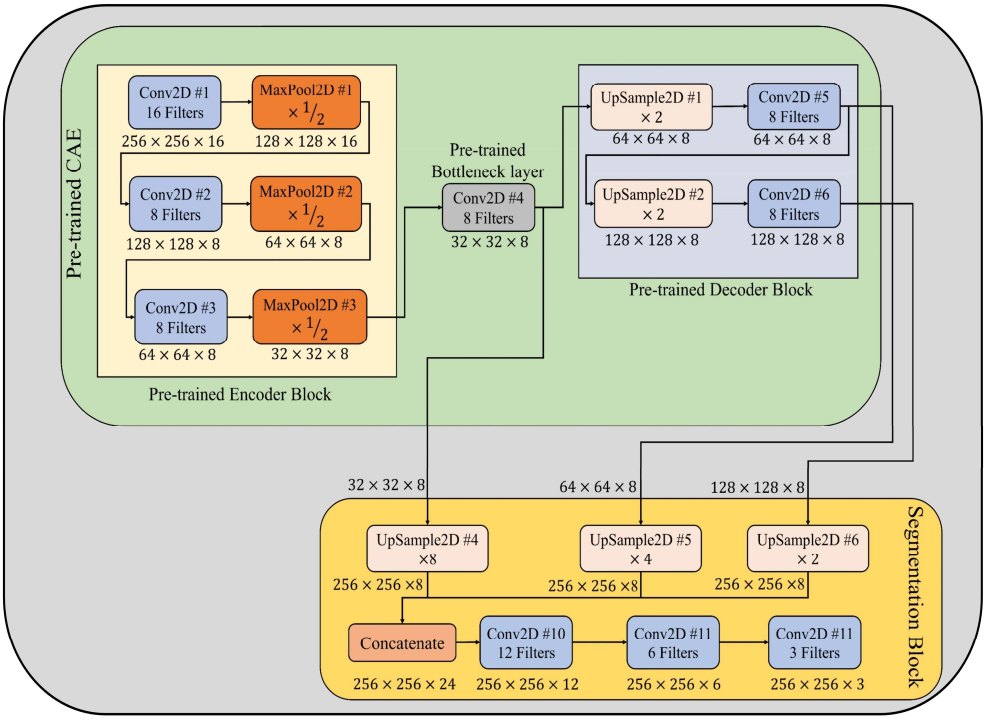


Figure 5.4: Architectural design of few-shot image segmentation model used to segment diseased areas from leaf images

$$SegLoss_{CCE} = -\frac{1}{N}\sum_{i=1}^{N} Y_{gmask}^{i} \log(Y_{pmask}^{i}) \tag{5.2}$$

$$SegLoss_{jaccard} = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{\left|Y_{gmask}^{i} \cap Y_{pmask}^{i}\right|}{\left|Y_{gmask}^{i} \cup Y_{pmask}^{i}\right|} \tag{5.3}$$

The flow of plant disease diagnosis and severity estimation through the proposed framework is given in the testing or inference stage of Figure 5.1. It can be observed from this figure that, in order to diagnose disease in a symptomatic leaf image, first, it is passed through the trained few-shot image classification model, which classifies the given leaf image into either healthy or one of diseased classes. If the given leaf image is classified as diseased, then only it is passed to the few-shot image segmentation model, which generates its segmentation mask. This segmentation mask encompasses of three values, i.e., 0 is for background, 1 is for healthy pixels, and 2 is for diseased pixels. After getting the segmentation mask from the few-shot image segmentation model, the disease severity is calculated by computing the percentage of diseased pixels

present in the given leaf image out of the total leaf pixels, i.e., sum of healthy and diseased pixels. The formula to compute the disease severity with the help of predicted segmentation mask has been given in equation 5.4, where N_d and N_h represents the diseased and healthy pixels of segmented leaf image, respectively.

$$Disease\ Severity\ (in\ \%) = \frac{N_d}{N_d + N_h} \times 100 \tag{5.4}$$

In the next section, the experiments performed in this chapter and the results obtained during experimentation have been discussed.

5.4. Experimental Study and Results

Experimentation of the research work presented in this chapter has been carried out on Nvidia DGX Server, which has been equipped with an Intel Xeon CPU, 528 Gigabytes of RAM, and Nvidia Tesla V100-SXM2 32 Gigabyte GPU. Scripts for the experimentation are written in the Python programming language, although other programming languages can also be used for experimentation. Furthermore, the models of the proposed framework are implemented using the Keras library, which is embedded in Tensorflow 2.6.0 (Chollet F. , 2015). This section is divided into three subsections. Subsection 5.4.1 provides the description of the Apple Tree Leaf Disease Segmentation (ATLDS) dataset, which is utilized to evaluate the effectiveness of the proposed PDSE-Lite framework. The details of different experiments conducted in the research work of this chapter are given in subsection 5.4.2. Furthermore, subsection 5.4.3 presents and discusses the results obtained from experimentation.

5.4.1. Dataset Description

In this chapter, the ATLDS dataset (Feng Jingze & Chao Xiaofei, 2022) has been employed to test the effectiveness of the PDSE-Lite framework in detecting plant diseases with severity estimation. This dataset comprises of healthy and four types of diseased apple tree leaf images, i.e., Alternaria Leaf Spot, Brown Spot, Gray Spot, and Rust. Few leaf images from each class of the ATLDS dataset, along with their annotated segmentation masks, are given in Figure 5.5, and their class-wise distribution is presented in Table 5.4.

The leaf images of ATLDS dataset were captured under varying degrees of disease, with approximately 51.9% acquired in controlled laboratory settings and 48.1% collected from real cultivation fields. These images were gathered across varied weather conditions and different times of the day. Furthermore, this dataset comprises of annotated segmentation masks corresponding to each leaf image of this dataset.

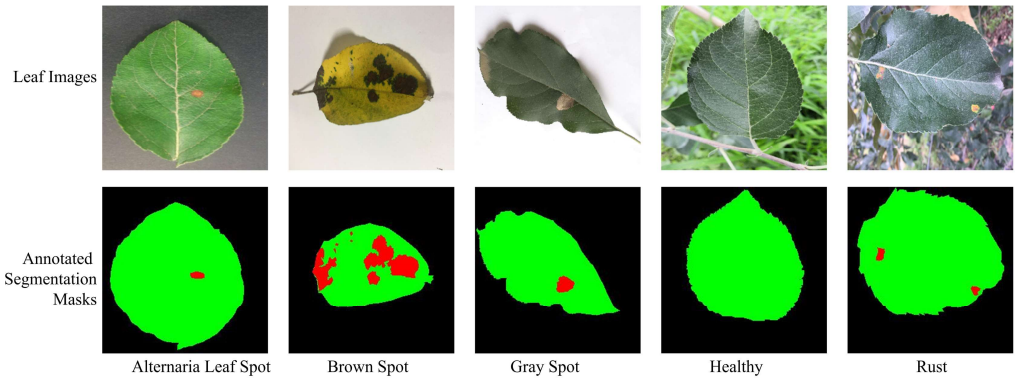


Figure 5.5: Leaf images representing each class within the ATLDS dataset, along with their annotated segmentation masks. The black, green, and red colors in segmentation masks represent the background, healthy, and diseased pixels, respectively

Table 5.4: Class-wise distribution of the ATLDS dataset

Class of leaf image	Alternaria leaf spot	Brown spot	Gray spot	Healthy	Rust	**Total**
Number of instances	278	215	395	409	344	**1641**

The details of experimentation performed to evaluate the performance of the proposed PDSE-Lite framework have been provided in the following subsection of this chapter.

5.4.2. Experimental Setup

The proposed framework has been designed and implemented in two stages. In the first stage, a lightweight CAE model has been built to reconstruct leaf images from the original leaf images with minimum reconstruction loss. In the second stage, a few-shot image classification and segmentation models are developed to identify plant diseases and segment diseased areas from symptomatic leaf images. The details of experimentation done to train and test the models of PDSE-Lite frameworks are given below:

Experiment 1: Training CAE Model of the PDSE-Lite Framework

In the first stage of the PDSE-Lite framework, a lightweight CAE model is designed and developed to reconstruct leaf images from original leaf images with minimal reconstruction loss. In order to train this model, the ATLDS dataset's leaf images are randomly divided into training, validation, and testing subsets with 70:15:15 ratio of sizes. This model is trained via Adam optimizer to minimize the NRMSE reconstruction loss. During training of the CAE model, the batch size has been kept as 32, and the number of epochs is 500. Furthermore, the ReLU activation function is applied to every convolutional layer of the CAE model. In order to prevent this model from overfitting, Keras' Early-stopping callback has been utilized with a patient value of 20. The values of these hyperparameters have been obtained through extensive experimentation. Next subsection discusses the experimental details to evaluate the performance of the few-shot image classification model of the PDSE-Lite framework in detecting diseases from leaf images.

Experiment 2: Training Few-Shot Image Classification Model of the PDSE-Lite Framework for Plant Disease Detection

In the second stage of the PDSE-Lite framework, a k-Shot (few-shot model trained on k leaf images per class) image classification model has been developed to detect plant diseases through their digital leaf image, where $k \in \{1,2,\dots,5\}$. Furthermore, $\left\lfloor \frac{N_c-k}{2} \right\rfloor$ and $\left\lceil \frac{N_c-k}{2} \right\rceil$ leaf images from different classes of dataset have been utilized for validating and testing the few-shot image classification model, where N_c is the number of leaf images present in c^{th} class of ATLDS dataset. This model is trained for 100 epochs with a batch size of k to minimize categorical cross-entropy loss (defined in equation 5.1.) using the Adam optimizer, and the ReLU activation function is utilized in every convolutional layer of the model.

Additionally, Early stopping callback of Keras with patience value 10 is applied during model training to prevent it from overfitting. Extensive experimentation has been conducted to determine the values of the aforementioned hyperparameters. This model's performance is compared with eight different state-of-the-art CNN

architectures, i.e., MobileNetV2 (Sandler, Howard, Zhu, Zhmoginov, & Chen, 2018), InceptionV3 (Szegedy, Vanhoucke, Ioffe, Shlens, & Wojna, 2016), GoogLeNet (Szegedy, et al., 2015), Xception (Chollet F. , 2017), ResNet-50 (He, Zhang, Ren, & Sun, 2016), NASNetMobile (Zoph, Vasudevan, Shlens, & Le, 2018), EfficientNetV2B0 (Tan & Le, 2021), and ConvNeXtTiny (Liu, et al., 2022). The details of experiments performed to evaluate the performance of the few-shot image segmentation model of the PDSE-lite framework have been given in the following subsection.

Experiment 3: Training Few-Shot Image Segmentation Model of the PDSE-Lite Framework to Segment Diseased Areas from Diseased Leaf Images

In order to estimate the severity of detected plant disease by segmenting the infected regions from leaf images, a *k*-Shot (few-shot model trained on *k* leaf images per class) image segmentation model has also been implemented in the second stage of the PDSE-Lite framework, where $k \in \{1,2,\dots,5\}$. On the other hand, remaining $\left\lfloor \frac{N_c-k}{2} \right\rfloor$ and $\left\lceil \frac{N_c-k}{2} \right\rceil$ leaf images from different diseased classes of the ATLDS dataset divided into validation and testing subsets, respectively. Furthermore, this few-shot image segmentation model has also been trained with a batch size of k for 100 epochs to minimize the sum of $SegLoss_{CCE}$ and $SegLoss_{jaccard}$ (defined in equations 5.2 and 5.3) via Adam optimizer (Kingma & Ba, 2014). Early stopping callback with patience value 10 is applied during model training to prevent the model from overfitting. The performance of this few-shot image segmentation model has been compared with U-Net3+ (Huang, et al., 2020) and DeepLabV3+ (Chen, Zhu, Papandreou, Schroff, & Adam, 2018) image segmentation models using MeanIoU and Dice-Score metrics. The mathematical formulas of MeanIoU and Dice-Score metrics have been given in equations 5.5 and 5.6, respectively. In these equations, Y^i_{pmask}, and Y^i_{gmask} represents predicted and ground truth segmentation masks for i^{th} leaf image, respectively. Additionally, N denotes the number of instances taken into consideration.

$$MeanIoU = \frac{1}{N}\sum_{i=1}^{N}\frac{\left|Y_{gmask}^{i} \cap Y_{pmask}^{i}\right|}{\left|Y_{gmask}^{i} \cup Y_{pmask}^{i}\right|} \tag{5.5}$$

$$Dice\text{-}Score = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \times \left|Y_{gmask}^{i} \cap Y_{pmask}^{i}\right|}{\left|Y_{gmask}^{i}\right| + \left|Y_{pmask}^{i}\right|} \tag{5.6}$$

In this section, the experimental details of the research work are discussed, and the next section presents the experimental results obtained from the experimentation.

5.4.3. Results and Discussion

In this chapter, a few-shot and lightweight framework named "PDSE-Lite" has been designed and developed for detecting plant diseases and estimating the severity of identified disease by utilizing digital plant leaf images. During the first stage of the proposed framework's development, a lightweight CAE model is built, which aims to learn how to reconstruct leaf images from original leaf images without losing much information. The CAE model's training, validation, and testing results have been given in subsection *A*. In the second stage of the proposed framework's development, a few-shot image classification and segmentation models are implemented. The results obtained from the training, validation, and testing phases of these models have been provided in subsections *B* and *C*, respectively. In subsection *D*, an ablation study to test the significance of the pretrained CAE model has been presented. Moreover, subsection *E* provides the statistical analysis of the proposed PDSE-Lite framework.

A. Results Obtained from Experiment 1

During the first phase of the proposed framework's development, a lightweight CAE model is trained to reconstruct the leaf images from the original leaf images without losing much information. In order to ensure this, the CAE model has been trained to minimize the NRMSE loss between original and reconstructed leaf images. Trends of the CAE model's training and validation NRMSE losses with respect to the number of epochs are shown in Figure 5.6. Furthermore, few leaf images and their reconstructed images from each class of the ATLDS dataset using the CAE model have been given in Figure 5.7.

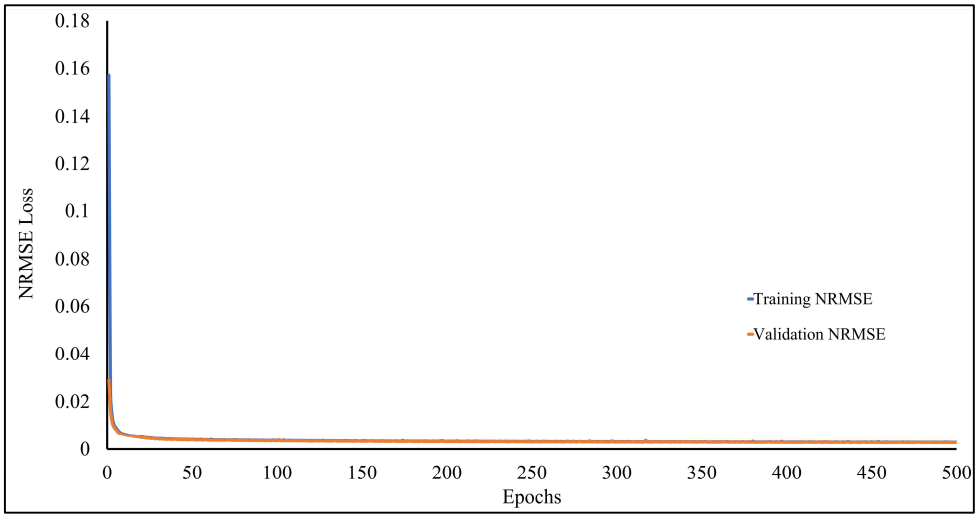


Figure 5.6: Trend of CAE model's training and validation loss

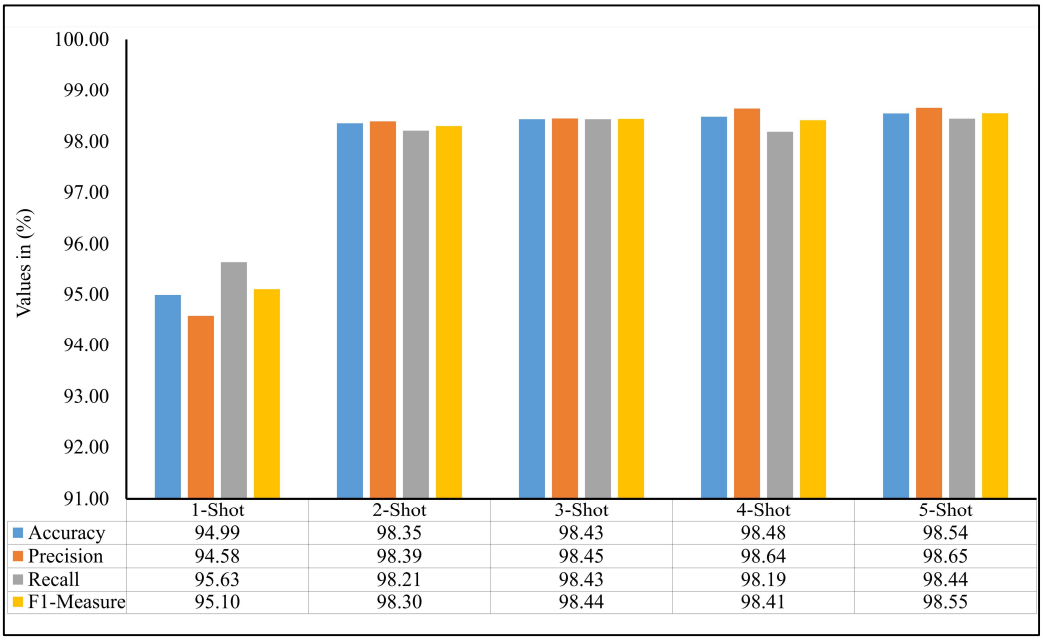
It can be seen from this figure that both the training and validation NRMSE losses of the CAE model are infinitesimally close to zero by the end of the 500th epoch. This can also qualitatively be verified from Figure 5.7, in which the reconstructed leaf images and original leaf images look very similar to each other. Next subsection provides the results obtained from experiment 2, in which the performance of the few-shot image classification model of the PDSE-Lite framework has been evaluated.

Figure 5.7: Few leaf images and their reconstructed images from each class of ATLDS dataset using the CAE model of the PDSE-Lite framework

B. Results Obtained from Experiment 2

This section provides and discusses the results obtained from experiment 2, in which a k-Shot image classification model is developed to diagnose plant diseases through their leaf images. In order to select the best value of k, accuracy, precision, recall, and f1-

measure of k-shot model for different values of k ranging from 1 to 5 has been evaluated and compared in Figure 5.8 on the test subset of the ATLDS dataset. The models obtained by using different values of k are referred by 1-Shot, 2-Shot, 3-Shot, 4-Shot, and 5-Shot image classification models in this thesis.



	1-Shot	2-Shot	3-Shot	4-Shot	5-Shot
Accuracy	94.99	98.35	98.43	98.48	98.54
Precision	94.58	98.39	98.45	98.64	98.65
Recall	95.63	98.21	98.43	98.19	98.44
F1-Measure	95.10	98.30	98.44	98.41	98.55

Figure 5.8: Accuracy, precision, recall, and f1-measure of various few-shot image classification models used to detect plant diseases

It can be seen by Figure 5.8 that the 2-Shot image classification model has attained 98.35% accuracy and 98.30% f1-measure on the dataset's test subset. Moreover, the performances of the 3-Shot, 4-Shot, and 5-Shot models are comparable to the 2-Shot model. Therefore, the 2-Shot image classification model has been employed in the PDSE-Lite framework to detect plant diseases, as it requires a minimum number of leaf images for training. Performance of the best k-Shot (i.e., 2-Shot) image classification model has been further compared with eight state-of-the-art CNN architectures on both validation and test subsets of the ATLDS dataset.

The variation of validation accuracies and losses with respect to the number of epochs for these models has been depicted in Figure 5.9 and Figure 5.10, respectively. It can be observed from these figures that the 2-Shot image classification model has achieved maximum accuracy and minimum loss. Furthermore, ResNet-50 achieved minimum accuracy and maximum loss among other models.

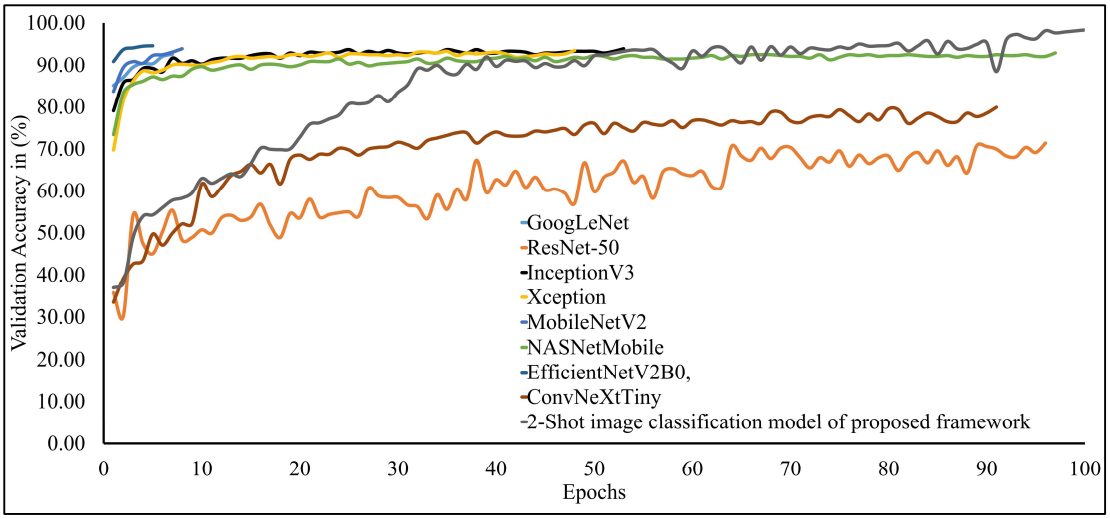


Figure 5.9: Trend of validation accuracy for the 2-Shot image classification model of PDSE-Lite framework and eight different CNN architectures

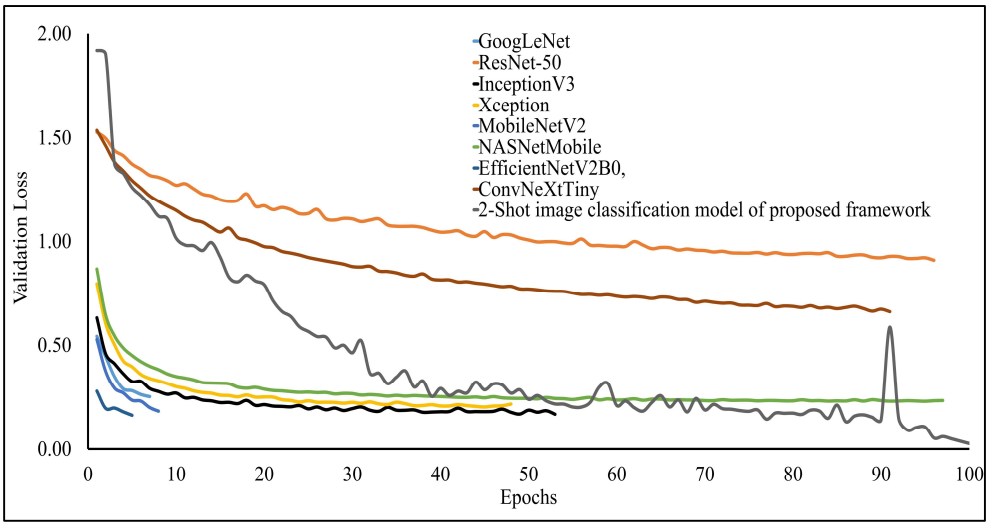
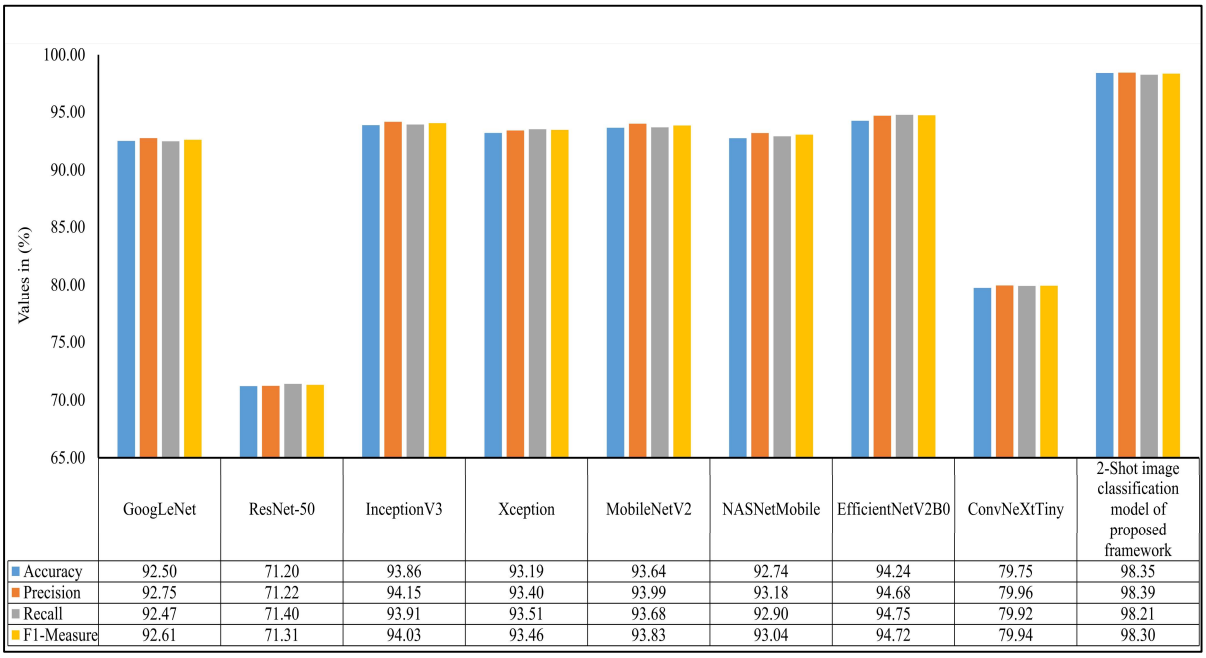


Figure 5.10: Trend of validation loss for the 2-Shot image classification model of PDSE-Lite framework and eight different CNN architectures

The values of accuracy, precision, recall, and f1-measure for the 2-Shot image classification model of the PDSE-Lite framework and eight state-of-the-art CNN architectures on the test subset of the ATLDS dataset are given in Figure 5.11. It can be seen from this figure that the 2-Shot image classification model of the PDSE-Lite framework outperformed other CNN architectures with 98.35% testing accuracy and 98.30% f1-measure. In addition, GoogLeNet, InceptionV3, Xception, MobileNetV2, NASNetMobile, and EfficientNetV2B0 achieved comparable performance. On the

other hand, ResNet-50 and ConvNeXtTiny attained minimum values for the aforementioned metrics.



	GoogLeNet	ResNet-50	InceptionV3	Xception	MobileNetV2	NASNetMobile	EfficientNetV2B0	ConvNeXtTiny	2-Shot image classification model of proposed framework
Accuracy	92.50	71.20	93.86	93.19	93.64	92.74	94.24	79.75	98.35
Precision	92.75	71.22	94.15	93.40	93.99	93.18	94.68	79.96	98.39
Recall	92.47	71.40	93.91	93.51	93.68	92.90	94.75	79.92	98.21
F1-Measure	92.61	71.31	94.03	93.46	93.83	93.04	94.72	79.94	98.30

Figure 5.11: Accuracy, precision, recall, and f1-measure of 2-Shot image classification model of PDSE-Lite framework and eight CNN models on ATLDS dataset's test subset

In order to examine the lightweight nature of the 2-Shot image classification model, the number of trainable weight parameters employed in this model and eight other state-of-the-art CNN architectures have been compared in Table 5.5. It can be perceived from this table that the 2-Shot image classification model requires the least trainable weight parameters, i.e., 8749, among other CNN architectures. Furthermore, ResNet-50 and ConvNeXtTiny architectures utilized comparable trainable weight parameters.

The predictions obtained from the 2-Shot image classification model for some sample leaf images representing each class within the ATLDS dataset, along with their ground truth labels, have been given in Figure 5.12. It can be perceived from this figure that 2-Shot image classification model correctly identifies the healthy and diseased classes by visualizing the given leaf images. The following subsection discusses the results obtained from experiment 3, in which the k-Shot image segmentation model has been developed for segmenting the infected regions from diseased leaf images.

Table 5.5: Number of trainable weight parameters employed in 2-Shot image classification of PDSE-Lite framework and eight CNN architectures

Models	**Number of trainable weight parameters (approximately)**
GoogLeNet	7 million
ResNet-50	25.6 million
InceptionV3	23.9 million
Xception	22.9 million
MobileNetV2	3.5 million
NASNetMobile	5.3 million
EfficientNetV2B0	7.2 million
ConvNeXtTiny	28.6 million
2-Shot image classification model of PDSE-Lite framework	**8749**

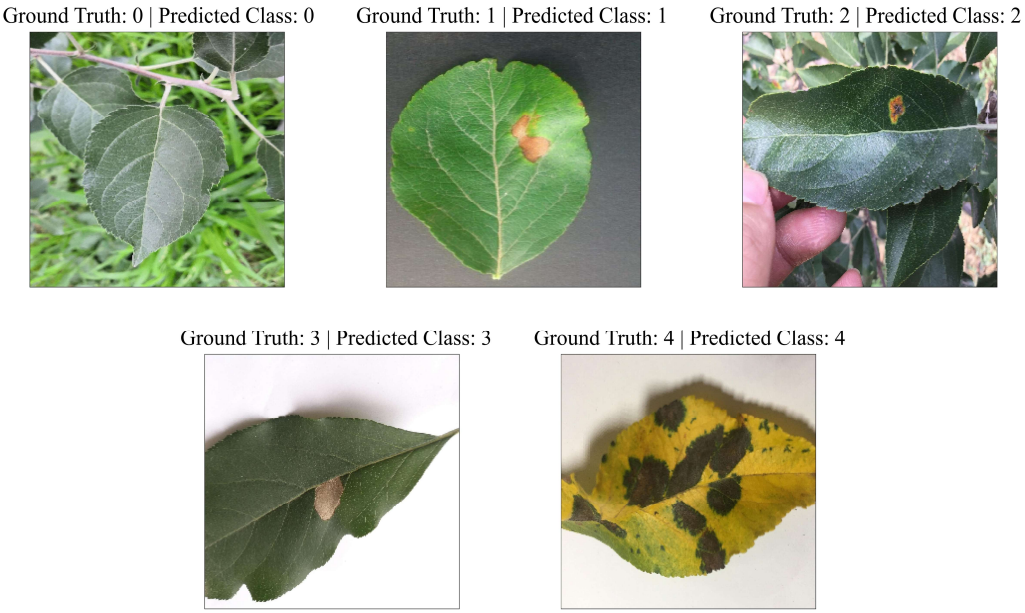
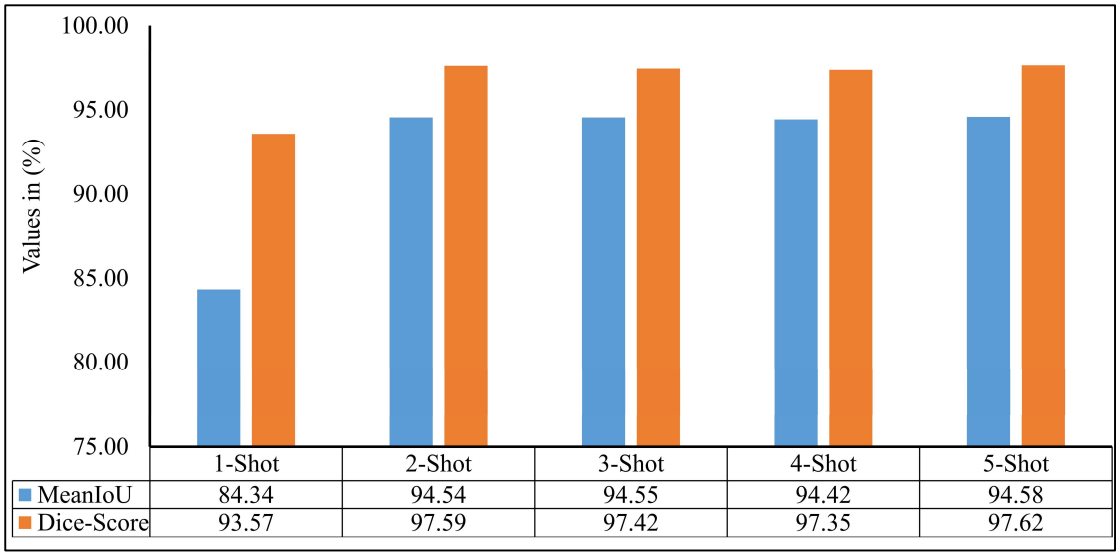


Figure 5.12: Predictions obtained from the 2-Shot image classification model of PDSE-Lite framework for some sample leaf images from each class of ATLDS dataset, along with their ground truth labels

C. Results Obtained from Experiment 3

The few-shot image segmentation model developed in experiment 3 has been evaluated with the help of two evaluation metrics, namely, MeanIoU and Dice-Score. The mathematical expressions for these metrics are given in equations 5.4 and 5.5, respectively. In order to choose the best value of $k \in \{1, 2, 3, 4, 5\}$ for k-Shot image segmentation model, the MeanIoU and Dice-Score have been computed for each value of k. The models obtained by using different values of k are referred by 1-Shot, 2-Shot, 3-Shot, 4-Shot, and 5-Shot image segmentation models in this thesis. The MeanIoU and Dice-Score for different k-shot models have been given in Figure 5.13

It can be perceived from this figure that the MeanIoU and Dice-Score for 2-Shot, 3-Shot, 4-Shot, and 5-Shot image segmentation models are comparable. Thus, the 2-Shot image segmentation model has been selected to segment the diseased from leaf images, as it uses minimum leaf images per class in model training. Performance of the best k-Shot (i.e., 2-Shot) image segmentation model has been further compared with DeepLabV3+ and U-Net3+ image segmentation models on both validation and test subsets of the ATLDS dataset.



	1-Shot	2-Shot	3-Shot	4-Shot	5-Shot
MeanIoU	84.34	94.54	94.55	94.42	94.58
Dice-Score	93.57	97.59	97.42	97.35	97.62

Figure 5.13: MeanIoU and Dice-Score of various few-shot image segmentation models used to segment diseased areas from leaf images

The plot of validation MeanIoU and loss with respect to the number of epochs for the 2-Shot image segmentation model along with the U-Net3+ and DeepLabV3+ models is

given in Figure 5.14 and Figure 5.15, respectively. It can be perceived from these figures that the 2-Shot image segmentation model of the PDSE-Lite framework outperformed U-Net3+ and DeepLabV3+ models by achieving the highest validation MeanIoU score, i.e., 94.87% and least validation loss. On the other hand, U-Net3+ and DeepLabV3+ models achieved comparable performances.

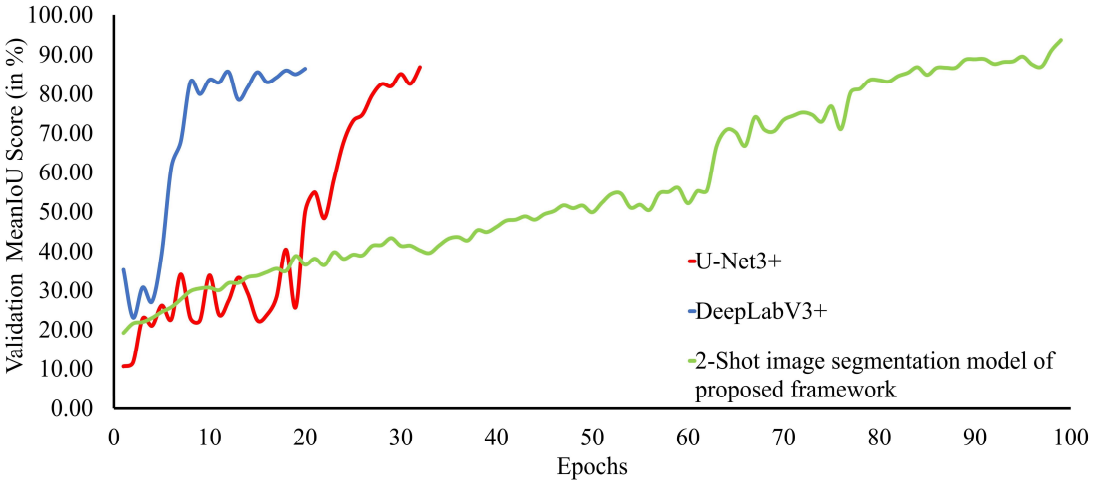


Figure 5.14: Plot of validation MeanIoU for the 2-Shot image segmentation model of PDSE-Lite framework along with U-Net3+ and DeepLabV3+ models

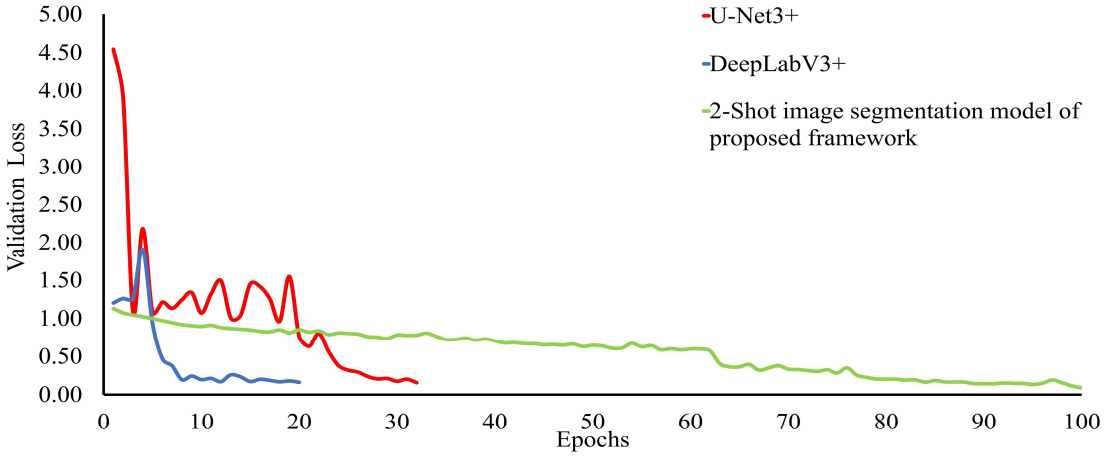


Figure 5.15: Plot of validation loss for the 2-Shot image segmentation model of PDSE-Lite framework along with U-Net3+ and DeepLabV3+ models

The performance of the 2-Shot image segmentation, U-Net3+, and DeepLabV3+ models on the test subset of the ATLDS dataset has been compared in Figure 5.16 using MeanIoU and Dice-Score. It can be observed from Figure 5.16 that the 2-Shot image

segmentation model outperformed the DeepLabV3+ and UNet3+ models on the test subset by attaining 94.54% and 97.59% MeanIoU score and Dice-Score, respectively.

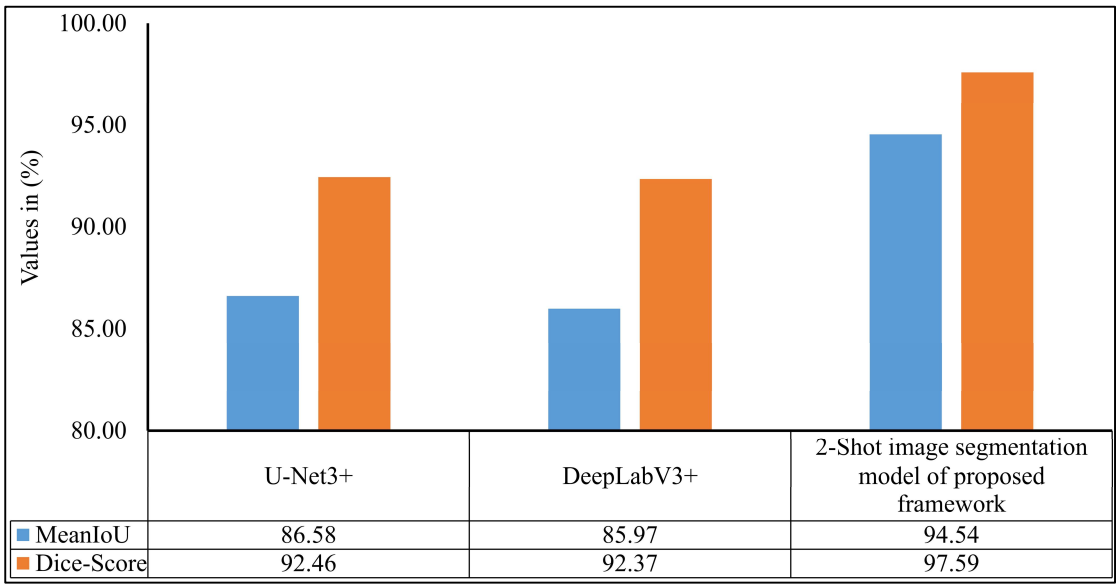


Figure 5.16: MeanIoU and Dice-Score of 2-Shot image segmentation model of PDSE-Lite framework, U-Net3+, and DeepLabV3+ models

In order to measure the lightweight nature of 2-Shot image segmentation model, its number of trainable weight parameters is compared in Table 5.6 with the trainable weight parameters used by U-Net3+ and DeepLabV3+ models. It can be observed from this that the 2-Shot image segmentation model uses significantly fewer trainable weight parameters, i.e., 7223, as compared to the U-Net3+ and DeepLabV3+ models.

Table 5.6: Number of trainable weight parameters used by 2-Shot image segmentation model, U-Net3+, and DeepLabV3+ models

Models	**Number of trainable weight parameters (approximately)**
U-Net3+	26.99 million
DeepLabV3+	11.85 million
2-Shot image segmentation model of PDSE-Lite framework	**7223**

The predicted segmentation masks for some sample leaf images from each diseased class of dataset and ground truth segmentation masks have been given in Figure 5.17.

Furthermore, the severity percentages obtained from predicted and ground truth segmentation masks have been computed and written above the segmentation masks. It can be seen from Figure 5.17 that predicted and ground truth masks are looking very similar to each other. In addition, the severity percentages computed for these segmentation masks are also comparable, which confirms the effectiveness of the proposed framework in plant disease severity estimation task.

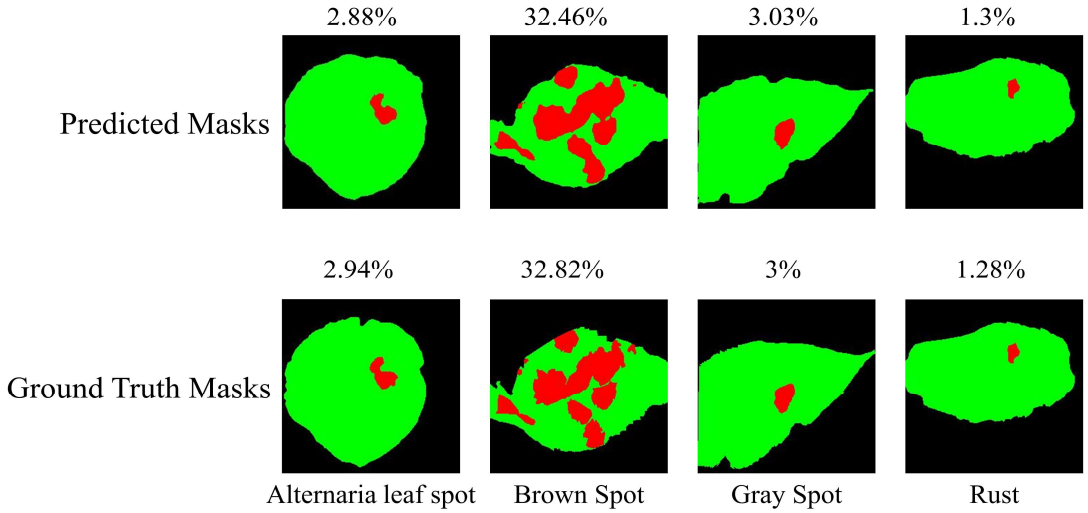


Figure 5.17: Predicted segmentation masks for some sample leaf images from each diseased class of dataset along with ground truth segmentation masks. The severity percentage obtained from predicted and ground truth segmentation masks have been written above the segmentation masks

In the following subsection, an ablation study has been conducted to test the significance of the pretrained CAE model in the development of the PDSE-Lite framework.

D. Ablation Study to Assess the Significance of Pretrained CAE Model

In order to test the significance of the pretrained CAE model in the 2-Shot image classification and 2-Shot image segmentation models of the PDSE-Lite framework, these models are also trained without utilizing the pre-trained CAE model. The results obtained from this experiment on the test subset of ATLDS dataset have been tabulated in Table 5.7 along with the results of 2-Shot image classification and 2-Shot image segmentation models of the PDSE-Lite framework in which the pre-trained CAE model is employed.

It can be concluded from Table 5.7 that the results of 2-Shot image classification and 2-Shot image segmentation models significantly improve when the pretrained CAE model has been utilized.

Table 5.7: Results obtained with and without utilizing the pretrained CAE model in the 2-Shot image classification and 2-Shot image segmentation models of the PDSE-Lite framework

Models of PDSE-Lite framework	**Without pretrained CAE model**		**With pretrained CAE model**	
2-Shot image classification model	Accuracy (%)	72.58	Accuracy (%)	98.35
	Precision (%)	74.64	Precision (%)	98.39
	Recall (%)	71.76	Recall (%)	98.21
	F1-Measure (%)	73.17	F1-Measure (%)	98.30
2-Shot image segmentation model	MeanIoU (%)	68.99	MeanIoU (%)	94.54
	Dice-Score (%)	72.54	Dice-Score (%)	97.59

In the subsequent subsection of this chapter, statistical hypothesis testing has been performed to evaluate the performance of the PDSE-Lite framework in estimating plant disease severity.

E. Statistical Analysis of PDSE-Lite Framework

The applicability of the PDSE-Lite framework in estimating plant disease severity has also been evaluated using statistical hypothesis testing via the Student t-test on the severity values obtained for ground truth and predicted segmentation masks. The paired t-test with two samples for means assuming unequal variance is employed in this chapter to test the null and alternate hypothesis given in equations 5.7 and 5.8, respectively. In these equations μ_{gt} denotes the mean of disease severity values obtained from ground truth segmentation masks and μ_{pred} represents the mean of disease severity values obtained from predicted segmentation masks.

$$H_0 \coloneqq \mu_{gt} - \mu_{pred} \neq 0 \tag{5.7}$$

$$H_1 \coloneqq \mu_{gt} - \mu_{pred} = 0 \tag{5.8}$$

During experimentation, the probability p value is computed for the t-test at $\alpha = 0.01$, i.e., if the obtained p value is lesser than 0.01, then the null hypothesis (H_0) is rejected, and the alternate hypothesis is accepted with 99% confidence. After analyzing the experimental results of the t-test, p value for the t-test is obtained as 0.008, which is less than 0.01. Thereby, the null hypothesis is rejected, and the alternate hypothesis is accepted with 99% confidence. Hence, this showcases the applicability of the PDSE-Lite framework in precisely estimating plant disease severity.

The proposed PDSE-Lite framework is compared in Table 5.8 with existing state-of-the-art research works available in the literature. It can be seen from this table that the proposed PDSE-Lite framework has achieved state-of-the-art performance in plant disease detection and severity estimation despite of using minimum trainable weight parameters and limited number of training samples.

Table 5.8: Comparison of proposed PDSE-Lite framework with state-of-the-art research works present in the literature

Research Work	Technique	Crop	Performance	Few-Shot	Disease Detection	Severity Estimation	Number of weight parameters
(Chao, Sun, Zhao, Li, & He, 2020)	Xception DenseNet (XDNet)	Apple	98.82% accuracy	×	✓	×	10.16 million
PDSE-Lite framework	**CAE and FSL**	**Apple**	**98.35% accuracy in plant disease detection**	✓	✓	✓	**8749 for few-shot image classification model**
			97.59% Dice-Score and 94.54% MeanIoU in segmenting diseased areas from leaf images				**7223 for few-shot image segmentation model**

Hence, it can be concluded that the proposed PDSE-Lite framework has several advantages over existing state-of-the-art research works present in the literature. First advantage of the PDSE-Lite framework lies in its ability to significantly reduce the reliance on large-scale manually annotated datasets, thereby minimizing the human efforts required to create such datasets. Moreover, the lightweight nature of the PDSE-Lite framework makes it suitable to be deployed on low-powered edge devices for on-site plant disease monitoring and timely intervention, aiding farmers in decision-making and crop management.

5.5. Chapter Summary

In this chapter, a few-shot and lightweight framework named "PDSE-Lite" was proposed to recognize plant diseases and estimate the severity of identified disease by segmenting the diseased regions of infected leaf images. The PDSE-Lite framework was designed and developed in two stages with the help of CAE and FSL. In the first stage, a lightweight CAE model was used to reconstruct the leaf images from the original leaf images with minimum loss of information. In the second phase of the proposed framework's implementation, a few-shot image classification and segmentation models were developed to accurately identify plant diseases and precisely segment the diseased areas from given leaf images, respectively.

Although the PDSE-Lite framework can perform plant disease detection and severity estimation very effectively and efficiently with the help of only two training samples per class. However, a recommender system is needed to assist the farmers by providing advisory to cure the identified plant disease at the estimated severity level. Hence, a recommender system named PlantD^2R^2S-Lite has been designed and developed in the next chapter of this thesis to conquer this drawback.

6. Plant Disease Diagnosis and Remedy Recommendation using Lightweight and Bilingual Recommender System

This chapter presents the Lightweight Plant Disease Diagnosis and Remedy Recommender System named PlantD^2R^2S-Lite which can generate bilingual advisory to cure the plant diseases by diagnosing the plant disease and estimating its severity. The proposed recommender system is implemented as an Android mobile application by embedding the models of the PDSE-Lite framework directly into the application. Therefore, the developed mobile application can also work seamlessly in remote locations where Internet connectivity may not be strong.

6.1. Introduction

Diagnosing plant diseases and providing advisory for disease remediation is crucial for the agricultural sector's growth, as it increases the crop yield and farmer's profit. In the previous chapter, the PDSE-Lite framework was proposed which can identify plant diseases and estimate their severity very effectively when trained on two leaf images per class. Moreover, the models used in the PDSE-Lite framework were lightweight in terms of the number of trainable weight parameters and thereby can be deployed in any low computational powered devices for on-site plant disease monitoring. However, it cannot assist farmers in taking the necessary steps to cure the identified plant disease. Thus, a recommender system is needed to support the farmers by providing advisory to remediate the identified plant disease.

In literature, various agricultural researchers have designed and developed different frameworks for automatic plant disease diagnosis and remedy recommendation (Siddiqua, Kabir, Ferdous, Ali, & Weston, 2022; Tembhurne, et al., 2023; Balasubramanian, et al., 2023; Ngugi, Abdelwahab, & Abo-Zahhad, 2020). However, none of the existing research work has considered the disease severity to provide the advisory for disease remediation, as it can significantly affect the advisory for curing the plant disease. Moreover, some of the existing frameworks are based on client-server architecture in which the Farmer (Client) captures the image of an infected leaf through

their smartphone, and then this image is sent to the centralized server. Afterward, the server identifies the disease using the received leaf image, and then the corresponding remedy is recommended to the farmer. This process requires an active Internet connection, which may not be available in remote locations. Hence, it limits the usage of existing frameworks in the plant disease diagnosis process and hampers in-time delivery of advisory for curing plant diseases.

Therefore, in order to conquer the aforementioned limitations, a lightweight and few-shot recommender system named PlantD^2R^2S-Lite is proposed in this chapter for plant disease diagnosis and generating disease remediation advisory. The proposed PlantD^2R^2S-Lite framework is developed as an Android mobile application for automatic plant disease identification and severity estimation. Furthermore, it also facilitates farmers in both English and Hindi language.

In this research work, the PlantD^2R^2S-Lite can diagnose and provide remedies for two diseases of Apple tree leaves, namely, Alternaria Leaf Spot and Brown Spot (Marssonina Leaf Blotch). The healthy and diseased leaf images of Apple trees are taken from the ATLDS dataset (Feng Jingze & Chao Xiaofei, 2022). However, the system can be trained to diagnose and provide remedies for other plant diseases as well. The proposed framework utilizes the PDSE-Lite framework's 2-shot image classification and segmentation models for plant disease identification and severity estimation, respectively (Bedi, Gole, & Marwaha, 2024). Nevertheless, any DL based image classification and segmentation models can be used in the application for identifying diseases in plants and severity estimation of diagnosed disease.

The advisory for curing the diagnosed plant disease at the estimated severity value of disease has been generated by fine-tuning the Bidirectional Encoder Representations from Transformer (BERT) model (Devlin, Chang, Lee, & Toutanova, 2019). The BERT model is fine-tuned on the text of two research papers (Nabi, et al., 2022; Madhu GS, Un Nabi, Iqbal Mir, & Hassan Raja, 2020) which described the management practices required for the ailment of Alternaria Leaf Spot, Brown Spot (Marssonina Leaf Blotch) diseases of Apple tree leaves. However, one can also fine-tune the BERT model to generate advisory of other plant diseases in a similar way. Moreover, all DL models utilized in the PlantD^2R^2S-Lite framework have been embedded into the mobile

application. Thereby, it can also work seamlessly in remote locations where Internet connectivity may not be strong.

Rest of this chapter is organized into four sections. Section 6.2 discusses the details of different frameworks developed in the literature for plant disease diagnosis and their remedy recommendation. In section 6.3, the proposed PlantD^2R^2S-Lite framework has been explained. Section 6.4 provides the details of experimentation done in this research work to evaluate the PlantD^2R^2S-Lite application's effectiveness and the results obtained from experimentation. Lastly, section 6.5 summarizes the chapter.

6.2. Related Work

In literature, various researchers have proposed different frameworks to diagnosis diseases in plants and recommend remedies for their cure. For example, Shrimali S (Shrimali, 2021) developed a mobile application named "PlantifyAI" for identifying different diseases from plants. The author utilized the PlantVillage dataset for training sixteen different CNN models to get the best model for plant disease detection. Out of the sixteen CNN models, MobileNetV2 with Canny Edge Detection filters outperformed other counterparts by achieving 95.7% accuracy and 96.1% f1-measure. Thereafter, the trained MobileNetV2 model was converted into tflite format for further deploying it into the Android mobile application using the Tensorflow Lite library of Python. Their mobile application also provides the advisory for the ailment of identified disease. Similar, work was also done by Ng et al. (Ng, Lin, Chuah, Tan, & Leung, 2021) for identifying diseases from grape plant leaf images. They have trained the Faster-R-CNN model for localizing infected lesions in grape plant leaf images. They also converted the trained model to tflite format for using it in Android mobile application. Their mobile application does not provide advisory for the remediation of diagnosed diseases.

Ahmed et al. (Ahmed & Reddy, 2021) developed an Android mobile application for identifying plant diseases from digital leaf images. They utilized the PlantVillage dataset for training a novel CNN model having approximately four million trainable weight parameters. As per their research work, the CNN model achieved 94% accuracy in detecting plant diseases. The authors of the paper developed an Android mobile application that can identify the disease of plants with the help of trained CNN model

deployed on a centralized server. In another work, Kumar et al. (Kumar, Raghavendran, Silambarasan, Kannan, & Krishnan, 2023) developed a cloud-based framework named Deeplens Classification and Detection Model (DCDM) to diagnose diseases from plant leaf images. Their proposed solution was built on the SageMaker framework of the Amazon Web Services (AWS) portal, which utilizes the PlantVillage dataset for training. The major disadvantage of the aforementioned research works (Ahmed & Reddy, 2021; Kumar, Raghavendran, Silambarasan, Kannan, & Krishnan, 2023) is that they send the suspected leaf image to the centralized server for disease detection, which cannot work if the mobile device does not have an active Internet connection. Moreover, they do not provide remedies in their application for curing the identified disease.

Janarthan et al. (Janarthan, Thuseethan, Rajasegarar, & Yearwood, 2022) developed a novel web-based framework named "Plant Pathology on Palms (P2OP)" for detecting plant diseases from farm fields. They proposed a novel model for plant disease detection based on CNN embedded Siamese Networks. They trained it on the combination of three datasets, namely, the Apple dataset (Thapa, Zhang, Snavely, Belongie, & Khan, 2020), Citrus leaf dataset (Rauf, et al., 2019), Tomato leaf images extracted from the PlantVillage dataset (Hughes, Salathé, & others, 2015). Furthermore, their proposed model achieved 97.35%, 97.01%, and 96.36% accuracy in detecting diseases of Apple, Citrus, and Tomato leaves, respectively. Thereafter, the authors of paper (Janarthan, Thuseethan, Rajasegarar, & Yearwood, 2022) converted their model into tflite format for further deployment on mobile devices. Rimon et al. (Rimon, Islam, Dey, & Das, 2022) built an Android mobile application named "PlantBuddy" for plant disease identification through their digital leaf images. They trained MobileNetV2 and InceptionV3 CNN models on the PlantVillage dataset (Hughes, Salathé, & others, 2015) for identifying plant diseases through smartphones. As per their paper, the MobileNetV2 model outperformed the InceptionV3 model by achieving 95.75% accuracy. The authors also provided the advisory to cure the identified disease along with a facility to contact the nearest agricultural office based on the user's current location.

Yulita et al. (Yulita, Amri, & Hidayat, 2023) built an Android mobile application for Tomato plant disease diagnosis. Their application utilized the DenseNet-201 CNN model for diagnosing diseases in Tomato plants, and it obtained an accuracy of 95.4% in detecting nine diseases of Tomato plants. In order to facilitate mobile-based disease diagnosis, the authors of paper (Yulita, Amri, & Hidayat, 2023) embedded the trained DenseNet-201 model in the mobile application via the Tensorflow Lite library of Python. In another study, Khan et al. (Khan, Nawaz, Kshetrimayum, Seneviratne, & Hussain, 2023) utilized the ViT model for disease detection in Tomato plants. In their paper, authors referred the Transformer model as "TomFormer" and its training was done on tomato plant leaf images obtained from three datasets, namely, PlantDoc, PlantVillage, and KuTomaDATA (collected from farm fields of Abu-Dhabi). Furthermore, their proposed TomFormer model achieved 87%, 81%, and 83% accuracy on the above-mentioned datasets, respectively. They also deployed the proposed model in a robotic device for easy and accurate identification of disease in farm fields.

Pineda et al. (Pineda Medina, et al., 2024) explored the potential of five predefined CNN models, namely, VGG-16, VGG-19, InceptionV3, MobileNetV2, and Xception, in diagnosing Potato plant diseases. These models were trained on Potato plant leaf images extracted from the PlantVillage dataset. The authors claimed in their paper that the MobileNetV2 model outperformed others by attaining 98.7% accuracy. After finding the best model among others, the authors of the paper developed an Android mobile application to facilitate the farmers in disease identification. Elinisa and Mduma (Elinisa & Mduma, 2024) evaluated the efficacy of their custom CNN model in Banana plant disease detection. As per the paper, their proposed model achieved 80.77% accuracy. This CNN model is converted to tflite format to be further utilized in the development of Android mobile application. This paper also provided the advisory for curing the identified disease. However, they have not considered the disease severity for advisory generation.

It can be concluded from the above discussion that most of the applications developed in literature can identify plant diseases and some of these applications can also provide advisory for disease remediation. However, the advisory for disease ailment given by these applications does not consider the severity of identified disease, which can

significantly affect the disease management practices. Moreover, the mobile applications developed in research work done by Ahmed and Reddy (Ahmed & Reddy, 2021) and Kumar et al. (Kumar, Raghavendran, Silambarasan, Kannan, & Krishnan, 2023) follow the client-server architecture in which an active Internet connection is required for diagnosing plant diseases which limits the usage of their systems in remote locations. Hence, in order to conquer these drawbacks of existing works, a novel lightweight and few-shot framework named PlantD^{2}R^{2}S-Lite has been proposed in this chapter. The proposed framework has been implemented as an Android mobile application for identifying diseases in plants and recommending advisory to cure the identified plant diseases even in remote locations. The developed application is bilingual in nature and can be used in both English and Hindi languages. The details of proposed PlantD^{2}R^{2}S-Lite framework have been described in the subsequent section.

6.3. Proposed PlantD^{2}R^{2}S-Lite Framework for Plant Disease Diagnosis and Remedy Recommendation

In this chapter, a lightweight and few-shot framework named PlantD^{2}R^{2}S-Lite is proposed as well as implemented to diagnose diseases in plants automatically and recommend different disease management measures. Unlike existing research works, the proposed PlantD^{2}R^{2}S-Lite generates advisory for managing plant diseases by considering the disease and its severity both. Moreover, to the best of our knowledge, there is no research work in the literature that has proposed a framework that can generate advisory to cure the disease by taking the severity value into consideration. The flow diagram of proposed PlantD^{2}R^{2}S-Lite framework is depicted in Figure 6.1, and its different modules are described in subsequent subsections.

6.3.1. Leaf Image Capturing Module

This module of the proposed PlantD^{2}R^{2}S-Lite framework captures the suspected leaf image by utilizing the digital camera of the device, i.e., mobile phone, tablet, etc. Alternatively, farmers can also use the leaf image stored on their device's storage for disease diagnosis and to get advisory for disease remediation. In both scenarios, end-users can crop, rotate, and rescale the leaf image obtained from the aforementioned methods. These options help the end-users in selecting the part of the image that

comprises of only leaf. This step can significantly improve the performance of disease diagnosis because objects other than leaves can distract the models of the PlantD^{2}R^{2}S-Lite framework in disease diagnosis. After selecting the suspected leaf image either from the device's camera or storage, this image is then pre-processed with the help of the *Image pre-processing module* of PlantD^{2}R^{2}S-Lite, which is described in the following subsection.

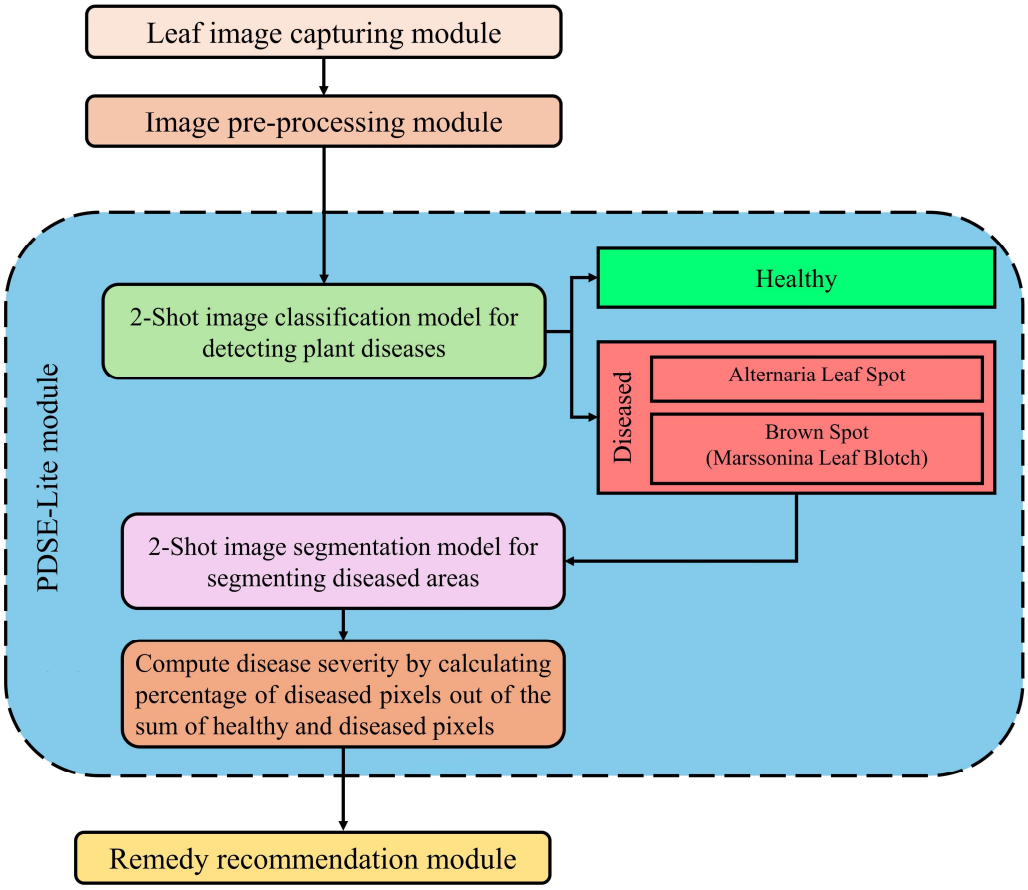


Figure 6.1: Flow diagram of proposed PlantD^{2}R^{2}S-Lite framework

6.3.2. Image Pre-processing Module

The proposed PlantD^{2}R^{2}S-Lite framework is implemented as an Android mobile application using Java programming language. In the Android operating system, an image is stored as a Bitmap in which each pixel value of the image is represented by a 32-bit signed integer. Each 8 bits of 32-bit signed integer from the left denotes the value of Alpha (Opacity), Red, Green, and Blue components of image pixels, respectively.

This module first resizes the captured leaf image into 256×256, as the models of the PDSE-Lite framework have been trained on the input size of 256×256. Thereafter, this module extracts the Red, Green, and Blue components of each image pixel in three arrays, namely *red-array, green-array, and blue-array* with the help of equations 6.1, 6.2, and 6.3, respectively. In these equations, $I_{x,y}$ represents the pixel of image *I* located in x^{th} row and y^{th} column, where $x, y \in [0,255]$. Furthermore, $0 \times FF$ denotes the hexadecimal representation of the decimal number 255. Additionally, $\gg$ and & represents the "binary right shift" and "binary AND" operator, respectively.

$$red\text{-}array = \frac{\left(I_{x,y} \gg 16\right) \,\&\, 0 \times FF}{255.0} \tag{6.1}$$

$$green\text{-}array = \frac{\left(I_{x,y} \gg 8\right) \,\&\, 0 \times FF}{255.0} \tag{6.2}$$

$$blue\text{-}array = \frac{I_{x,y} \,\&\, 0 \times FF}{255.0} \tag{6.3}$$

After extracting the Red, Green, and Blue components of image pixels, a three-dimensional integer array of size 256×256×3 is formed by stacking the red-array, green-array, and blue-array. This three-dimensional array is further passed to the *PDSE-Lite module* for detecting the probable disease in the given leaf image and estimating its severity. The *PDSE-Lite module* of the proposed framework is described in the subsequent subsection.

6.3.3. PDSE-Lite Module

This module identifies the probable plant disease, segments the diseased areas, and estimates the severity of diagnosed disease. The details of PDSE-Lite framework have been given in the previous chapter. The PDSE-Lite module first identifies the probable disease in the leaf image obtained from *Image pre-processing module* with the help of the 2-Shot image classification model. If the given leaf image is classified into one of the diseased classes, then its diseased regions are segmented via a 2-Shot image segmentation model. Thereafter, the severity of disease is estimated by calculating the percentage of diseased pixels out of the sum of healthy and diseased pixels. Once the

disease and its severity value are obtained, the disease curing advisory is generated with the help of the *Remedy recommendation module* of the proposed framework, which is described in the next subsection.

6.3.4. Remedy Recommendation Module

This module generates advisory to the farmers in natural language for remediation of identified disease under the calculated severity value. In order to generate disease curing advisory, a pre-trained BERT model (Devlin, Chang, Lee, & Toutanova, 2019) is fine-tuned on the text of two research papers (Madhu GS, Un Nabi, Iqbal Mir, & Hassan Raja, 2020; Nabi, et al., 2022) which have detailed descriptions of Alternaria Leaf Spot and Brown Spot (Marssonina Leaf Blotch) disease of Apple tree leaves and their management. *Remedy recommendation module* first builds a question, "How to manage disease d at severity s", where d and s represents the identified plant disease and severity value of d in percentage, respectively. This question is then answered with the help of the BERT model's question-answering pipeline by utilizing the text of research papers (Madhu GS, Un Nabi, Iqbal Mir, & Hassan Raja, 2020; Nabi, et al., 2022). The question-answering pipeline of the BERT model generates answers to the given questions by finding similar text in the given text corpus. In this research work, the answer having maximum matching text is selected and then divided into bullet points for easy understanding of farmers.

All six modules of the proposed PlantD^2R^2S-Lite are combined together to build an Android mobile application that can diagnose plant disease, estimate its severity, and provide advisory to cure the identified disease. Moreover, the application developed in this research work can be used by farmers in English or Hindi language, and it can even work in remote locations on any device having an Android operating system and camera with limited or no Internet connectivity. The following section describes the details of different experiments performed to evaluate the effectiveness of PlantD^2R^2S-Lite application in plant disease diagnosis and generating advisory for their remediations.

6.4. Experimental Study and Results

This section is divided into four subsections. Section 6.4.1 describes the specifications of different hardware and software that have been utilized in this research work for implementing the proposed PlantD^{2}R^{2}S-Lite framework. Furthermore, the effectiveness of proposed PlantD^{2}R^{2}S-Lite application has been evaluated with the help of ATLDS dataset, which is described in section 6.4.2. This section also discusses the steps performed to evaluate the performance of proposed PlantD^{2}R^{2}S-Lite mobile application in plant disease detection, severity estimation, and remedy recommendation. Furthermore, subsection 6.4.3 presents and discusses the results obtained from different experiments described in subsection 6.4.2. The features of PlantD^{2}R^{2}S-Lite application, along with its various user interfaces, are given in subsection 6.4.4.

6.4.1. Hardware and Software Specifications

Implementation of the proposed PlantD^{2}R^{2}S-Lite framework has been done on a Dell G3 3500 laptop having Intel® Core™ i7-10750H CPU, 32 GB RAM, and Nvidia GTX1650 GPU with 4GB GPU memory. Moreover, the Samsung A54 5G mobile has an Exynos 1380 Octa-core CPU with 8GB RAM and 256 GB internal storage, and it has been utilized to test and execute the Android application developed in this research work. The Dell G3 3500 laptop is running on Windows 11 Home edition with 23H2 version, and the Samsung A54 5G mobile is running on One UI 6.1 (powered by Samsung) based on the Android 14 operating system.

PyCharm professional edition version 2023.3.4 and Android Studio Iguana (version 2023.2.1) have been used in the work to write Python scripts and develop the PlantD^{2}R^{2}S-Lite Android mobile application, respectively. The user interface of the mobile application is developed with the help of eXtensible Markup Language (XML), and its working is implemented through Java programming language. Additionally, Tensorflow version 2.4.4 has been used to convert the disease detection and segmentation models of the PDSE-Lite framework into tflite format for embedding these models into the mobile application. Tensorflow Lite libraries ("*org.tensorflow:tensorflow-lite-support:0.1.0*", "*org.tensorflow:tensorflow-lite-meta*

data:0.1.0", "*org.tensorflow:tensorflow-lite-gpu:2.3.0*") for Android operating system has also been imported from Maven Repository into Android Studio for interacting with the pre-trained disease detection and segmentation models. Moreover, the "*com.github.yalantis:ucrop:2.2.6*" library provides the image cropping functionality in the developed mobile application, and "*me.biubiubiu.justifytext:library:1.1*" aligns the text of advisory provided by the application.

The advisory for curing the identified disease in the given leaf image under measured severity value is generated via the pre-trained BERT model imported from *the cdQA* library (Pietsch, et al., 2019). Next subsection provides the description of ATLDS dataset, which is used to evaluate the effectiveness of mobile application. Moreover, the details of different experiments that have been conducted to evaluate the performance of the proposed PlantD^2R^2S-Lite mobile application are also included in the following subsection.

6.4.2. Experimentation to Evaluate the Performance of Proposed PlantD^2R^2S-Lite Framework

This research work utilizes the healthy and diseased leaf images of Apple trees infected with Alternaria Leaf Spot and Brown Spot (Marssonina Leaf Blotch) disease. These images are extracted from the ATLDS (Feng Jingze & Chao Xiaofei, 2022) dataset to measure the effectiveness of PlantD^2R^2S-Lite mobile application in plant disease detection and severity estimation. The Apple tree leaf images belonging to the ATLDS dataset have been taken at different stages of infection and varying weather conditions.

Moreover, approximately 52% of leaf images were captured in the controlled environment of the laboratory, and 48% of leaf images were taken from real farm fields. This dataset also consists of segmentation masks for each leaf image to highlight the diseased and leaf regions in the image. The ATLDS dataset comprises of 278, 215, and 409 leaf images belonging to Alternaria Leaf Spot, Brown Spot, and Healthy class, respectively. Some healthy and diseased leaf images and their segmentation masks obtained from ATLDS dataset are shown in Figure 6.2.

In order to evaluate the effectiveness of PlantD^2R^2S-Lite mobile application, thirty healthy and diseased leaf images of Apple trees infected from Alternaria Leaf Spot and

Brown Spot (Marssonina Leaf Blotch) have been extracted randomly from the ATLDS dataset. These leaf images are utilized to measure the performance of application's disease detection abilities based on accuracy, precision, recall, and f1-measure metrics (Zaki & Wagner Meira, 2014). Furthermore, the ability of PlantD^{2}R^{2}S-Lite mobile application to segment diseased areas from leaf images has been assessed via MeanIoU and Dice-Score metrics defined in equations 5.5 and 5.6, respectively.

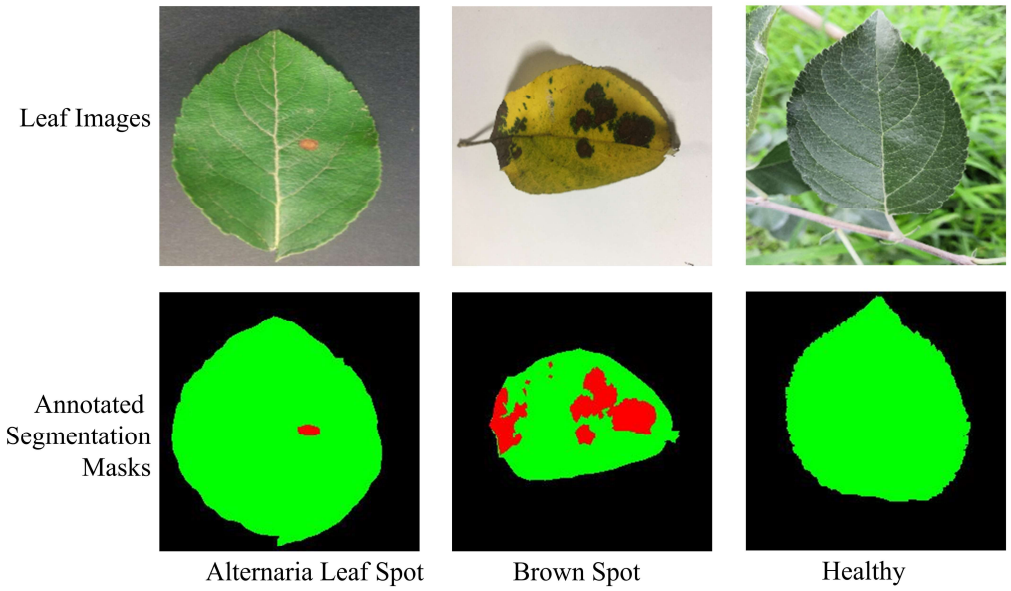


Figure 6.2: Some healthy and diseased leaf images, along with their segmentation masks obtained from ATLDS dataset

Effectiveness of the developed mobile application's severity estimation abilities has been evaluated by performing statistical hypothesis testing on actual and estimated disease severity values with the help of the Student-t-test. This statistical testing has been done to test null (H_0) and alternate (H_1) hypotheses are given in equation 6.4 and 6.5, respectively, where $\overline{s_g}$ and $\overline{s_e}$ represents the mean of actual and estimated disease severity values. Moreover, the probability (p) is calculated in the Student-t-test at $\alpha = 0.01$ or 99% confidence interval.

$$H_0: \overline{s_g} - \overline{s_e} \neq 0 \tag{6.4}$$

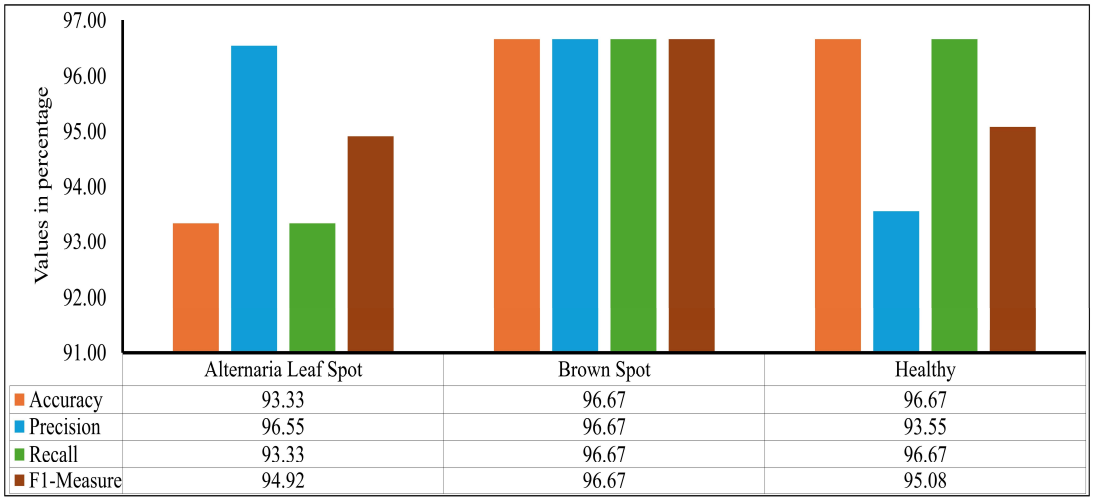
$$H_1: \overline{s_g} - \overline{s_e} = 0 \tag{6.5}$$

Next subsection discusses the results obtained from different experiments conducted in this research work to validate the PlantD^{2}R^{2}S-Lite application's effectiveness.

6.4.3. Experimental Results

This subsection provides and discusses the results obtained from different experiments conducted in this research work to evaluate the performance of the PlantD^{2}R^{2}S-Lite mobile application. These results are obtained on thirty healthy and diseased leaf images of Apple trees infected with Alternaria Leaf Spot and Brown Spot disease.

The values obtained for accuracy, precision, recall, and f1-measure metrics for healthy and diseased leaf images infected from the aforementioned diseases have been plotted in Figure 6.3. The results plotted in Figure 6.3 demonstrate that the PlantD^{2}R^{2}S-Lite application has detected Alternaria Leaf Spot disease with 93.33% accuracy and 94.92% f1-measure. On the other hand, leaf images infected with Brown Spot disease are diagnosed with 96.67% accuracy and f1-measure. Additionally, the healthy leaf images have been classified with 96.67% accuracy and 95.08% f1-measure via the PlantD^{2}R^{2}S-Lite mobile application.



	Alternaria Leaf Spot	Brown Spot	Healthy
Accuracy	93.33	96.67	96.67
Precision	96.55	96.67	93.55
Recall	93.33	96.67	96.67
F1-Measure	94.92	96.67	95.08

Figure 6.3: Accuracy, precision, recall, and f1-measure of PlantD^{2}R^{2}S-Lite application for detecting healthy or diseased Apple tree leaf images infected from Alternaria Leaf Spot and Brown Spot

Performance of mobile application in segmenting diseased areas from leaf images has been measured in terms of MeanIoU and Dice-Score metrics. The values for these metrics have been plotted in Figure 6.4. which demonstrates that PlantD^{2}R^{2}S-Lite can

segment the diseased lesions of Alternaria Leaf Spot and Brown Spot diseases with 94.92% and 95.04% MeanIoU, respectively. Moreover, it has achieved 96.57% and 96.49% Dice-Score in segmenting diseased areas of Alternaria Leaf Spot and Brown Spot diseases, respectively.

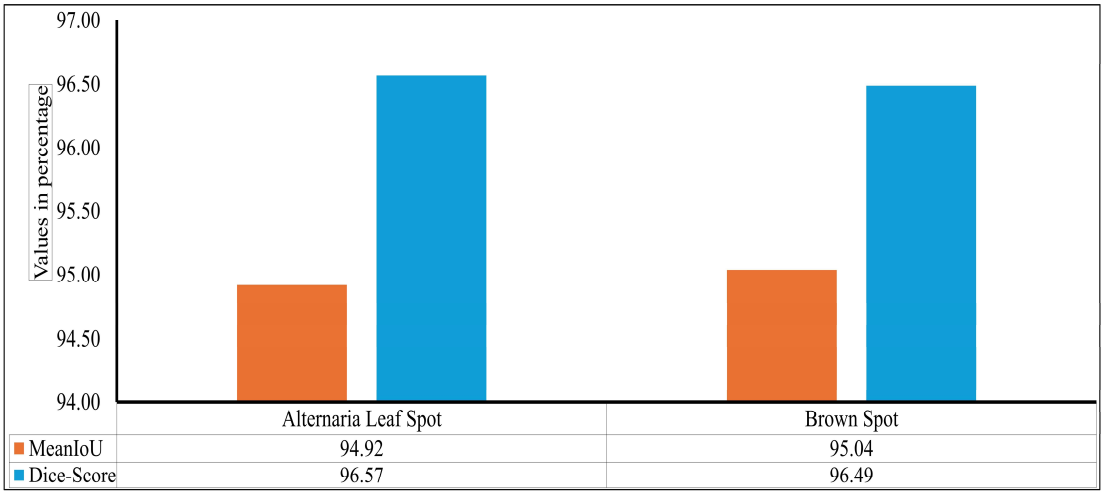


Figure 6.4: MeanIoU and Dice-score of PlantD²R²S-Lite in segmenting diseased areas from leaf images

In order to evaluate the effectiveness of PlantD²R²S-Lite mobile application in disease severity estimation, statistical hypothesis testing has been performed with the help of the Student-t-test by considering actual and predicted disease severity values. The probability value (p) is computed during experimentation at $\alpha = 0.01$, i.e., 99% confidence interval, and the p value obtained from experimentation is 0.0078. As the p value is lesser than the value of α, therefore, the null hypothesis (given in equation 6.4) is rejected, and the alternate hypothesis (stated in equation 6.5) is accepted with 99% confidence interval. Hence, it can be concluded that the developed application can estimate the plant disease severity accurately.

After evaluating the effectiveness of each module of the PlantD²R²S-Lite mobile application, its functionalities are further compared in Table 6.1 with existing mobile applications available in the literature. It can be observed from Table 6.1 that the PlantD²R²S-Lite mobile application developed in this research work performs plant disease diagnosis, segments diseased lesions, predicts disease severity, and generates recommendations to remediate the detected disease by considering the estimated severity level. Moreover, these functionalities are provided without any active Internet

connection, and hence, the proposed PlantD^{2}R^{2}S-Lite application can also work in remote locations. In order to facilitate non English speaking farmers, the application can also be used in Hindi language. Additionally, the mobile application designed and implemented in this paper requires the least space in the mobile device as compared to other applications developed in the literature. Therefore, it can also work on less computationally powered smartphones.

Table 6.1: Comparison of functionalities provided in various mobile applications developed in the literature for plant disease diagnosis and remedy recommendation

Mobile application name	**Crop/ Dataset**	**Advisory (✓ /×)**	**Require Internet connection (✓ /×)**	**Considered severity in advisory (✓ /×)**	**Is bilingual (✓ /×)**	**Application size in Megabytes (MBs)**	**Accuracy**
Potato Crop Diseases (PCD)	Leaf images of potato plants extracted from the PlantVillage dataset	×	×	×	×	77.57	92.86%
Artificial Intelligence based Disease Identification System for Crops (AI-DISC)	Rice, Wheat, Maize, Tomato, Mustard, Cotton, Brinjal, Soybean, Apple, Peach, Kinnow, Mandarin, Lemon, Chickpea, Green gram, Clusterbean, Mothbean, Cucurbits, Chilli, Coriander	✓	✓	×	×	97.34	Not computed by the developers of this application
PlantD^{2}R^{2}S-Lite	**Apple tree leaf images extracted from the ATLDS dataset**	✓	×	✓	✓	**22.78**	**95.56%**

Next subsection discusses the features of PlantD^{2}R^{2}S-Lite application via its various user interfaces.

6.4.4. Features and User-Interfaces of PlantD^{2}R^{2}S-Lite Application

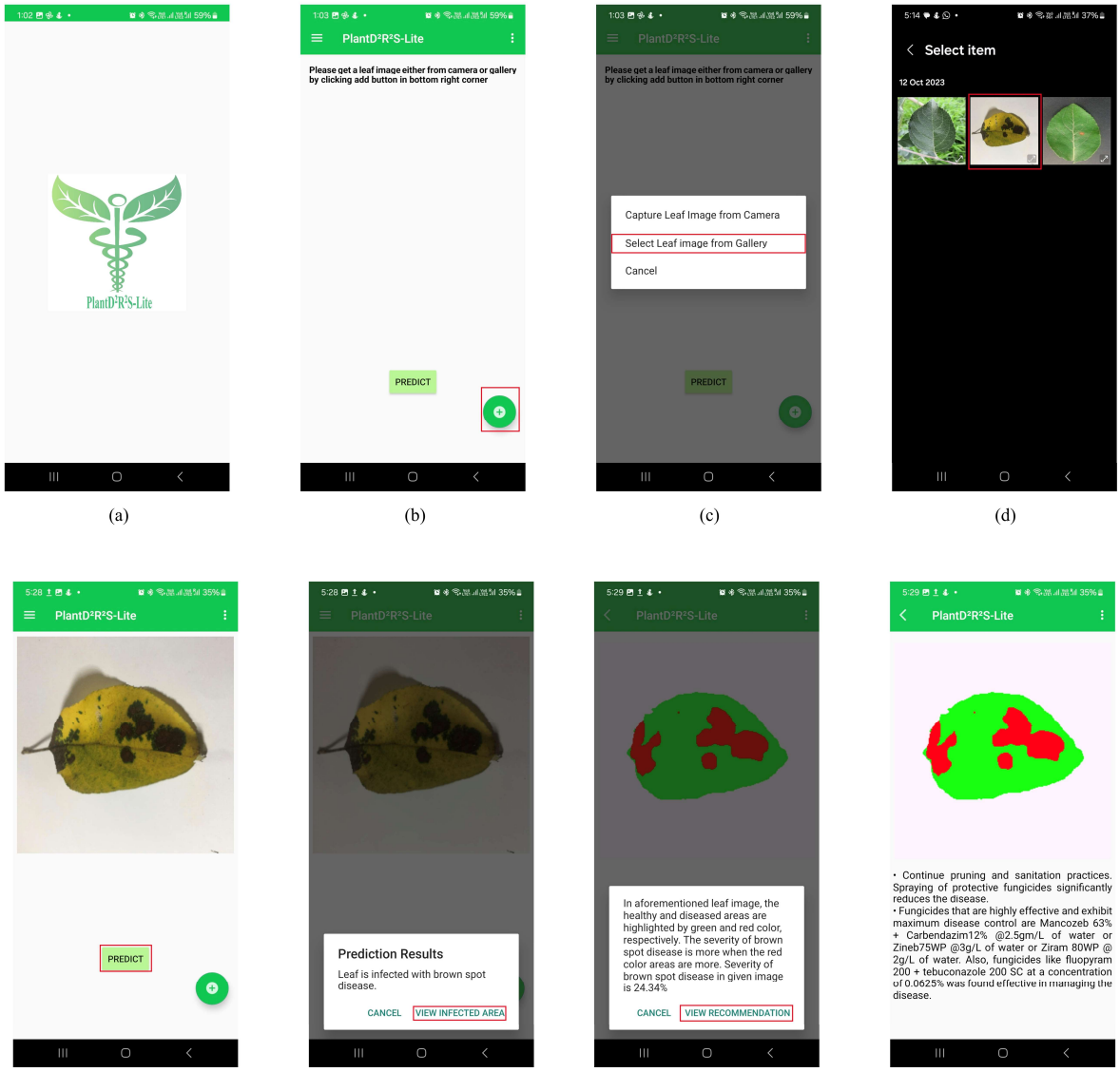
The mobile application developed in this research work is compatible with a smartphone with an Android operating system version 10 or higher and a minimum of 8GB storage along with 2GB of RAM. Moreover, camera of the device should work properly in order to take photographs of suspected leaf images.

PlantD^{2}R^{2}S-Lite application would start with a splash screen showing its logo, and the screenshot of welcome screen is shown in Figure 6.5(a). Once the application is loaded, it shows the user interface for capturing the leaf image. The leaf image can be captured either from the camera or from the gallery of the mobile device. In order to start capturing the leaf image from the camera or gallery, the user needs to click ⊕ icon given in the bottom right corner of the screen shown in Figure 6.5(b). This will provide two choices to the user, as depicted in Figure 6.5(c). First choice is to capture the suspected leaf image from the camera, and second choice is to use an already captured image from the device gallery.

If the user selects the second option, then the user's device gallery is opened up (as shown in Figure 6.5(d)) for choosing the suspected leaf image. Once the leaf image is selected from the device gallery, the leaf image is displayed on the screen, as shown in Figure 6.5(e). Thereafter, in order to identify the probable disease in the given leaf image, the user needs to click the predict button highlighted with a red colored rectangle in Figure 6.5(e), and the diagnosis result is then prompted to the user as shown in Figure 6.5(f).

It can be seen from Figure 6.5(f) that an option to view the infected areas of the input leaf image (highlighted with a red colored rectangle) is also provided in the PlantD^{2}R^{2}S-Lite application. Selecting this option will invoke the segmentation model of the PDSE-Lite framework, which takes the input leaf image and generates its segmentation mask, as shown in Figure 6.5(g). In this segmentation mask, the diseased areas are highlighted in red color, and the healthy regions are highlighted with green color. After segmenting diseased areas from given leaf image, the severity of identified disease is computed as

per equation 5.4, and its output is displayed on the user interface of the application, as shown in Figure 6.5(g).



(a) (b) (c) (d)

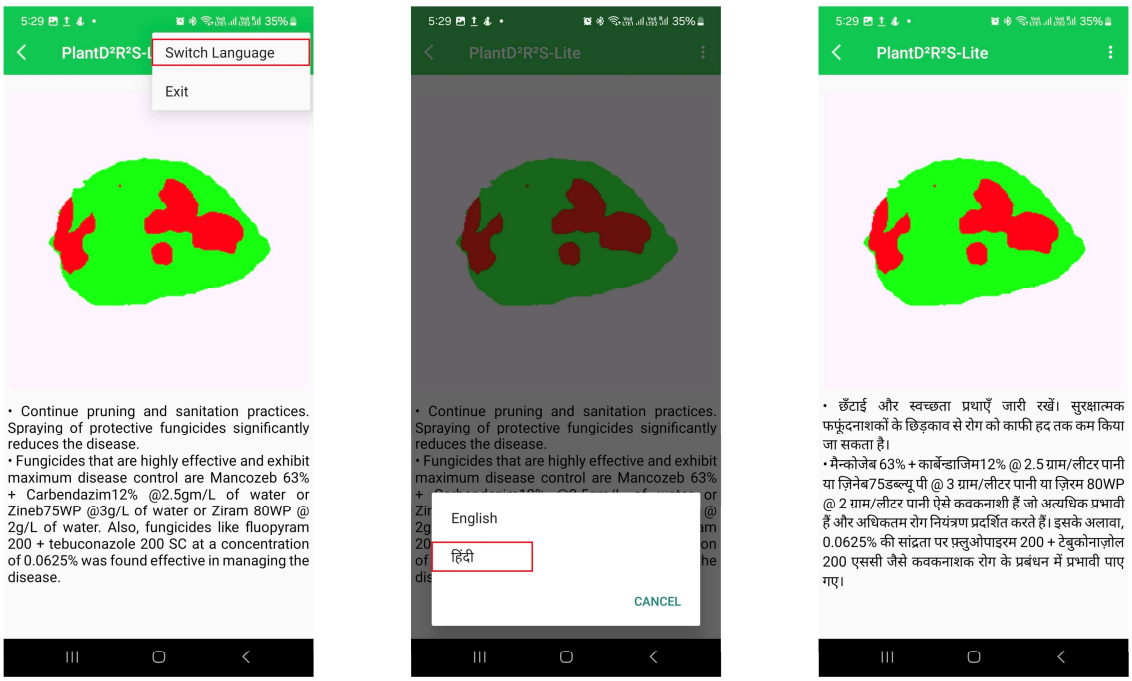
(e) (f) (g) (h)

Figure 6.5: User interfaces showing the working of the PlantD²R²S-Lite application

Once the disease and its severity value are determined, the user is prompted to view recommendations for curing the identified disease. If the user clicks on the '*View Recommendation*' option (highlighted with a red colored rectangle in Figure 6.5(g)), then *the Remedy recommendation module* is executed for generating the advisory to cure the identified disease under the calculated severity value. The user interface for

providing the advisory to cure the identified plant disease under the estimated severity value is given in Figure 6.5(h).

In order to facilitate non-English speaking Indian farmers, the PlantD^{2}R^{2}S-Lite application also provides an option to switch the language of the application (as shown in Figure 6.6(a)) to Hindi via clicking the *three dots icon* present at the top right corner of the screen. This will allow the user to choose the default language of the application, as shown in Figure 6.6(b). If the user has selected Hindi language as the default language, all user interfaces of the application are translated into Hindi language. A snapshot of the PlantD^{2}R^{2}S-Lite application's user interface in the Hindi language is given in Figure 6.6(c).



(a) (b) (c)

Figure 6.6: User interfaces to change the language of PlantD^{2}R^{2}S-Lite application

All of the aforementioned features of the PlantD^{2}R^{2}S-Lite application are provided without utilizing an Internet connection, as the models used in *PDSE-Lite* and *Remedy recommendation* modules are embedded in the application. Hence, the PlantD^{2}R^{2}S-Lite application can be used by farmers even in remote locations for diagnosing plant diseases and getting advisory to cure the identified disease.

6.5. Chapter Summary

In this chapter, a lightweight and bilingual recommender system named PlantD^{2}R^{2}S-Lite was proposed for diagnosing plant diseases through their leaf images and providing disease curing advisory. The proposed framework comprised of four modules, namely, *Leaf image capturing, Image pre-processing, PDSE-Lite, and Remedy recommendation* modules. All of these modules were integrated to develop an Android mobile application that can help farmers in diagnosing plant diseases, highlighting diseased areas of leaf image, estimating the severity of identified disease, and taking timely actions to cure the identified disease. The developed application can be used by farmers in either English or Hindi language, and it can work even in remote areas where Internet connectivity may not be strong. The next chapter concludes the research work presented in this thesis and discusses its few limitations along with the direction for future research.

7. Conclusion and Directions for Future Work

This chapter concludes the research work by summarizing all the contributions made in the thesis. Additionally, it provides limitations of current research work and various future research directions to conquer these limitations.

7.1. Thesis Summary and Contributions

Early stage plant disease diagnosis can be helpful to minimize crop yield loss and maximize the farmer's profit, and it is a big challenge in the growth of farming sector. A lot of work has been done in the literature in this area using various ML or DL techniques, but most of these techniques utilize large number of trainable weight parameters and large number of annotated leaf images for training. Therefore, the aim of this thesis is to develop a lightweight DL model which requires a smaller number of annotated leaf images for training, as annotating leaf images is laborious and time-consuming task. The reason for choosing the DL model over the ML model lies in its ability to automatically extract important features from raw data, which eliminates the requirement of a separate feature extraction module.

Hence, an attempt has been made in this thesis to first develop a lightweight DL model named PlantGhostNet for diagnosing a plant disease from leaf images. Although PlantGhostNet model has utilized significantly lesser number of trainable weight parameters as compared to other state-of-the-art DL models, however, they are still high in number. Next, a lightweight and hybrid DL model based on CAE, and CNN has been proposed in this thesis for detecting a plant disease from their leaf images. Though the hybrid DL model can identify a plant disease very efficiently and effectively, this model cannot diagnose multiple plant diseases with high accuracy. Therefore, in order to deal with this issue, a lightweight and improved ViT model named TrIncNet has been proposed in this thesis.

Disease severity is required in order to take necessary steps for curing the identified disease as it can provide the quantitative assessment of the damage caused by the pathogen of the identified disease. Hence, this thesis proposed a lightweight and few-

shot framework named PDSE-Lite for plant disease diagnosis and severity estimation. Lastly, an android mobile application named "PlantD^{2}R^{2}S-Lite" has been developed in this thesis to identify plant diseases after capturing the leaf image, estimate the severity of the identified disease, and generate an advisory for curing the plant disease. The developed mobile application can be used in English or Hindi language and it can also work in remote locations where Internet connectivity may not be strong. The major contributions made in this thesis are summarized below:

Contribution 1: PlantGhostNet: Lightweight CNN Model for Identifying a Plant Disease

The PlantGhostNet model has been proposed in this thesis for diagnosing a plant disease from their digital leaf images. This model utilized the Ghost and Squeeze-and-Excitation modules to reduce the trainable weight parameters and improve the model's performance, respectively. Ghost Module generates the feature maps in two phases. First, it generates few feature maps via conventional convolution operation. After that, the cheap linear operations were applied to the feature maps generated in the first phase to obtain the desired number of feature maps. Thereby, it requires lesser number of trainable weight parameters used to generate target feature maps as compared to conventional convolution operation. Squeeze-and-Excitation Module adaptively prioritizes each channel of the input feature map by assigning them weights, resulting in better performance of the model.

The effectiveness of PlantGhostNet model was evaluated on the leaf images of peach plants extracted from the PlantVillage dataset, which contains healthy and diseased leaf images infected from Bacterial Spot disease. Experimental results demonstrated that the PlantGhostNet model obtained the highest accuracy along with the minimum number of trainable weight parameters as compared to other state-of-the-art models.

Contribution 2: A Lightweight Hybrid DL Model based on CAE and CNN for Identifying a Plant Disease

A novel lightweight hybrid DL model was designed and developed by combining the CAE and CNN for diagnosing single type of plant disease using their leaf images. This model reduced the spatial dimension of input leaf images via CAE before classifying it

with CNN, which resulted in significantly lesser number of training parameters. CNN model used the pretrained encoder block of CAE, which further reduced the training time of the proposed model.

Performance of the lightweight hybrid DL model was evaluated on healthy and diseased leaf images of peach plants extracted from the PlantVillage dataset. Experimental results showed that the proposed model achieved slightly less accuracy than the PlantGhostNet model, but it utilized roughly 86.41% less trainable weight parameters as compared to the PlantGhostNet model.

Contribution 3: TrIncNet: Lightweight and Improved Vision Transformer Model for Identifying Multiple Plant Diseases

Trans-Inception Network (TrIncNet) model was proposed to diagnose multiple plant diseases of different crops. TrIncNet encompasses of multiple linearly connected Trans-Inception blocks, which was designed by replacing the MLP module with the Inception module in the encoder block of the ViT model. As a result of this replacement, the Trans-Inception block utilized 32.67% lesser number of trainable weight parameters than the original encoder block of ViT. Unlike the ViT model, the TrIncNet model also uses the skip connections around each Trans-Inception block to make the model more resistant towards the vanishing gradient problem.

In order to showcase the applicability of TrIncNet model in detecting plant diseases, it has been trained and tested on two plant disease datasets, namely PlantVillage and Maize datasets. Experimental results showed that the TrIncNet model outperformed existing state-of-the-art research works by achieving 99.93% and 96.93% accuracies on PlantVillage and Maize datasets, respectively.

Contribution 4: PDSE-Lite: Lightweight and Few-Shot Framework for Plant Disease Severity Estimation

Plant Disease Severity Estimation-Lite (PDSE-Lite) framework was proposed for estimating the severity of identified plant diseases. This framework was designed and developed in two stages. In the first stage, a lightweight CAE model was implemented and trained to reconstruct leaf images from original leaf images with minimal

reconstruction loss. In the subsequent stage, pretrained layers of the CAE model built in the first stage were utilized to develop the few-shot image classification and segmentation models.

The proposed PDSE-Lite framework can diagnose multiple plant diseases of a crop and estimate the severity of identified disease. Performance of the PDSE-Lite framework has been evaluated on the ATLDS dataset, which comprises of healthy and four types of diseased leaf images of apple trees. Experimental results demonstrated that the proposed framework achieved 98.35% accuracy in disease detection and 94.54% MeanIoU value in segmenting diseased areas from leaf images by utilizing only two leaf images per class for model training. Therefore, the PDSE-Lite framework reduces the reliance on large manually annotated datasets and minimizes the human efforts required to create such datasets. This framework can also estimate the severity of identified disease by calculating the percentage of diseased pixels out of the total leaf pixels (i.e., sum of healthy and diseased pixels) present in the segmented leaf image. The severity estimation ability of the PDSE-Lite framework was verified by applying the Student-t-test on the actual and predicted severity values.

Contribution 5: PlantD^{2}R^{2}S-Lite: Lightweight and Bilingual Plant Disease Diagnosis and Remedy Recommender System

A lightweight Plant Disease Diagnosis and Remedy Recommender System named PlantD^{2}R^{2}S-Lite was proposed to diagnose plant diseases and provide advisory to cure them. The proposed framework was comprised of four modules, namely, Leaf image capturing module, Image pre-processing module, PDSE-Lite module, and Remedy recommendation module.

Leaf image capturing module uses the device camera or gallery for capturing the leaf image. This leaf image is then passed to the Image pre-processing module, which resizes the image into 256×256 dimension and converts it to a three-dimensional array. Thereafter, this three-dimensional array is sent to the PDSE-Lite module, which identifies the disease from the leaf image. Furthermore, it also segments the diseased lesions of input leaf images and calculates the disease severity by computing the percentage of diseased pixels present in the segmented leaf image. Once the disease

and its severity value are found then the Remedy recommendation module generates the disease remediation advisory. This advisory has been generated by fine-tuning the pre-trained BERT model on the text of two research papers that have the description and disease management practices of apple tree leaves.

The proposed PlantD^{2}R^{2}S-Lite framework is implemented as an Android mobile application. It provides a simplified and bilingual user interface for farmers through which they can diagnose diseases of their plants and get remedies to cure them. Effectiveness of the proposed PlantD^{2}R^{2}S-Lite mobile application was evaluated on thirty randomly selected leaf images of apple trees from Alternaria Leaf Spot, Brown Spot, and Healthy classes of the ATLDS dataset. Experimental results revealed that the PlantD^{2}R^{2}S-Lite mobile application can effectively and efficiently diagnose plant disease from the captured leaf images. Additionally, it can also segment the diseased lesions from leaf images and provide the severity value of identified disease accurately.

Hence, the PlantD^{2}R^{2}S-Lite application designed and developed in this thesis can help farmers in diagnosing plant diseases, highlighting diseased areas of leaf image, estimating the severity of identified diseases, and taking timely actions to cure the identified disease even in remote areas where internet connectivity may not be strong. Next section discusses a few limitations of the research work presented in this thesis, along with some directions for future research work.

7.2. Limitations and Directions for Future Work

The research work presented in this thesis can identify plant diseases and estimate the severity of identified disease under the assumption that the leaf is infected only one disease at a time. However, in the real world, leaves can be infected with multiple diseases at a time, and the models or frameworks presented in this thesis cannot identify co-occurring plant diseases. PlantD^{2}R^{2}S-Lite application developed in this thesis works only in English and Hindi language, which limits its usage to the farmers who know either of these languages. These limitations of the research work presented in this thesis can be conquered in the future by extending this work. Some future research directions related to the work presented in this thesis are given below:

1. In the future, the dataset which encompasses of leaf images suffering from multiple plant diseases simultaneously, can be utilized for training a cost-effective DL model that can identify co-occurring plant diseases.

2. The proposed PlantD^{2}R^{2}S-Lite application can be extended to provide advisory for curing the identified plant disease in other regional languages. This would help the regional farmers who do not know either English or Hindi language.

3. The PlantD^{2}R^{2}S-Lite application can also be extended to give spoken disease remediation advisory in multiple languages and dialects.

References

Abbas, A., Jain, S., Gour, M., & Vankudothu, S. (2021, 8). Tomato plant disease detection using transfer learning with C-GAN synthetic images. *Computers and Electronics in Agriculture, 187*, 106279. doi:10.1016/j.compag.2021.106279

Ahmad, A., Gamal, A., & Saraswat, D. (2023, 1). Toward Generalization of Deep Learning-Based Plant Disease Identification Under Controlled and Field Conditions. *IEEE Access, 11*, 9042-9057. doi:10.1109/ACCESS.2023.3240100

Ahmed, A., & Reddy, G. (2021, 7). A Mobile-Based System for Detecting Plant Leaf Diseases Using Deep Learning. *AgriEngineering, 3*(3), 478-493. doi:10.3390/agriengineering3030032

Ahmed, K., Shahidi, T. R., Alam, S. M., & Momen, S. (2019). Rice Leaf Disease Detection Using Machine Learning Techniques. In *2019 International Conference on Sustainable Technologies for Industry 4.0 (STI)* (pp. 1-5). Dhaka, Bangladesh: IEEE.

Argüeso, D., Picon, A., Irusta, U., Medela, A., San-Emeterio, M., Bereciartua, A., & Alvarez-Gila, A. (2020, 8). Few-Shot Learning approach for plant disease classification using images taken in the field. *Computers and Electronics in Agriculture, 175*, 105542. doi:10.1016/j.compag.2020.105542

Arjovsky, M., Chintala, S., & Bottou, L. (2017). Wasserstein Generative Adversarial Networks. *34th International Conference on Machine Learning (ICML)* (pp. 214-223). Sydney, NSW, Australia: JMLR.org. doi:10.5555/3305381.3305404

Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021, 3). Plant leaf disease classification using EfficientNet deep learning model. *Ecological Informatics, 61*, 101182. doi:10.1016/j.ecoinf.2020.101182

Ba, J., Kiros, J., & Hinton, G. (2016, 7). Layer Normalization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, & R. Garnett (Ed.), *30th Conference on Neural*

Information Processing Systems. arXiv, pp. 1-14. Barcelona Spain: Curran Associates, Inc.

Bahdanau, D., Cho, K., & Bengio, Y. (2015, 9). Neural Machine Translation by Jointly Learning to Align and Translate. *3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings* (pp. 1-15). San Diego, CA, USA: International Conference on Learning Representations, ICLR.

Balasubramanian, S., Rajaram, S., Selvabharathy, S., Krishnamurthy, V., Sakthivel, V., & Ramprasad, A. (2023). Mobile Application for Leaf Disease Classification. *2023 International Conference on Advances in IoT and Security with AI. 755 LNNS*, pp. 297-307. New Delhi, India: Springer Science. doi:10.1007/978-981-99-5085-0_29

Bana, T., Loya, J., & Kulkarni, S. (2022, 2). ViT - Inception - GAN for Image Colourisation. *International Conference on Machine Learning, Optimization, and Data Science. 13163 LNCS*, pp. 105-118. Grasmere, United Kingdom: Springer Science and Business Media Deutschland GmbH. doi:10.1007/978-3-030-95467-3_8

Barbedo, J. (2014, 12). An Automatic Method to Detect and Measure Leaf Disease Symptoms Using Digital Image Processing. *Plant Disease, 98*(12), 1709-1716. doi:10.1094/PDIS-03-14-0290-RE

Bedi, P., & Gole, P. (2021, 5). Plant disease detection using hybrid model based on convolutional autoencoder and convolutional neural network. *Artificial Intelligence in Agriculture, 5*, 90-101. doi:10.1016/j.aiia.2021.05.002

Bedi, P., Gole, P., & Agarwal, S. (2021, 2). 18 Using deep learning for image-based plant disease detection. In V. Jain, J. M. Chatterjee, A. Kumar, & P. S. Rathore (Eds.). Germany: De Gruyter. doi:10.1515/9783110691276-018

Bedi, P., Gole, P., & Marwaha, S. (2024, 1). PDSE-Lite: lightweight framework for plant disease severity estimation based on Convolutional Autoencoder and Few-Shot Learning. *Frontiers in Plant Science, 14*, 1-20. doi:10.3389/fpls.2023.1319894

Biswas, S., Jagyasi, B., Singh, B., & Lal, M. (2014, 6). Severity identification of Potato Late Blight disease from crop images captured under uncontrolled environment. *2014 IEEE Canada International Humanitarian Technology Conference - (IHTC)* (pp. 1-5). Montreal, QC, Canada: IEEE. doi:10.1109/IHTC.2014.7147519

Bock, C., Poole, G., Parker, P., & Gottwald, T. (2010, 3). Plant Disease Severity Estimated Visually, by Digital Photography and Image Analysis, and by Hyperspectral Imaging. *Critical Reviews in Plant Sciences, 29*(2), 59-107. doi:10.1080/07352681003617285

Borhani, Y., Khoramdel, J., & Najafi, E. (2022, 7). A deep learning based approach for automated plant disease classification using vision transformer. *Scientific Reports, 12*(1), 11554. doi:10.1038/s41598-022-15163-0

Bromley, J., Guyon, I., LeCun, Y., Säckinger, E., & Shah, R. (1994). Signature verification using a "Siamese" time delay neural network. *6th International Conference on Neural Information Processing Systems* (pp. 737-744). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. doi:10.5555/2987189.2987282

Chao, X., Sun, G., Zhao, H., Li, M., & He, D. (2020, 6). Identification of Apple Tree Leaf Diseases Based on Deep Learning Models. *Symmetry, 12*(7), 1065. doi:10.3390/sym12071065

Chen, J., Chen, J., Zhang, D., Sun, Y., & Nanehkaran, Y. (2020). Using deep transfer learning for image-based plant disease identification. *Computers and Electronics in Agriculture, 173*, 105393. doi:10.1016/j.compag.2020.105393

Chen, J., Wang, W., Zhang, D., Zeb, A., & Nanehkaran, Y. (2021, 4). Attention embedded lightweight network for maize disease recognition. *Plant Pathology, 70*(3), 630-642. doi:10.1111/ppa.13322

Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018, 9). Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation.

European Conference on Computer Vision (ECCV) 2018 (pp. 833-851). Munich, Germany: Springer. doi:10.1007/978-3-030-01234-2_49

Chen, S., Zhang, K., Zhao, Y., Sun, Y., Ban, W., Chen, Y., . . . Yang, T. (2021, 5). An Approach for Rice Bacterial Leaf Streak Disease Segmentation and Disease Severity Estimation. *Agriculture, 11*(5), 420. doi:10.3390/agriculture11050420

Chen, W.-Y., Liu, Y.-C., Kira, Z., Wang, Y.-C., & Huang, J.-B. (2019, 4). A Closer Look at Few-shot Classification. *Seventh International Conference on Learning Representations (ICLR)* (pp. 1-17). New Orleans, USA: OpenReview.net.

Chohan, M., Khan, A., Chohan, R., Katpar, S. H., & Mahar, M. S. (2020, 5). Plant Disease Detection using Deep Learning. *International Journal of Recent Technology and Engineering (IJRTE), 9*(1), 909-914. doi:10.35940/ijrte.A2139.059120

Chollet, F. (2015). *Keras*. Retrieved December 19, 2020, from Keras: https://keras.io/api/

Chollet, F. (2017, 7). Xception: Deep Learning with Depthwise Separable Convolutions. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1800-1807). Honolulu, HI, USA: IEEE. doi:10.1109/CVPR.2017.195

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (pp. 4171-4186). Stroudsburg, PA, USA: Association for Computational Linguistics. doi:10.18653/v1/N19-1423

Dhaka, V., Meena, S., Rani, G., Sinwar, D., Kavita, K., Ijaz, M., & Woźniak, M. (2021, 7). A Survey of Deep Convolutional Neural Networks Applied for Prediction of Plant Leaf Diseases. *Sensors, 21*(14), 4749. doi:10.3390/s21144749

Dhiman, P., Kukreja, V., Manoharan, P., Kaur, A., Kamruzzaman, M., Dhaou, I., & Iwendi, C. (2022, 2). A Novel Deep Learning Model for Detection of Severity

Level of the Disease in Citrus Fruits. *Electronics, 11*(3), 495. doi:10.3390/electronics11030495

Dhingra, G., Kumar, V., & Joshi, H. (2018, 8). Study of digital image processing techniques for leaf disease detection and classification. *Multimedia Tools and Applications, 77*(15), 19951-20000. doi:10.1007/s11042-017-5445-8

Divyanth, L., Ahmad, A., & Saraswat, D. (2023, 2). A two-stage deep-learning based segmentation model for crop disease quantification based on corn field imagery. *Smart Agricultural Technology, 3*, 100108. doi:10.1016/j.atech.2022.100108

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., . . . Houlsby, N. (2021, 5). An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *9th International Conference on Learning Representations* (pp. 1-22). Austria: OpenReview.net.

Elinisa, C., & Mduma, N. (2024, 3). Mobile-Based convolutional neural network model for the early identification of banana diseases. *Smart Agricultural Technology, 7*, 100423. doi:10.1016/j.atech.2024.100423

Feng Jingze, & Chao Xiaofei. (2022). *Apple Tree Leaf Disease Segmentation Dataset.* Beijing, China: Science data bank. doi:10.11922/sciencedb.01627

Feng, D., Feng, M., Ozer, E., & Fukuda, Y. (2015, 7). A Vision-Based Sensor for Noncontact Structural Displacement Measurement. *Sensors, 15*(7), 16557-16575. doi:10.3390/s150716557

Ferentinos, K. (2018). Deep learning models for plant disease detection and diagnosis. *Computers and Electronics in Agriculture, 145*, 311 - 318. doi:10.1016/j.compag.2018.01.009

Finn, C., Abbeel, P., & Levine, S. (2017). Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks. *34th International Conference on Machine Learning (ICML)* (pp. 1126-1135). Sydney, NSW, Australia: JMLR.org.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning* (1st ed.). Cambridge, United Kingdom: The MIT Press.

Hadsell, R., Chopra, S., & LeCun, Y. (2006). Dimensionality Reduction by Learning an Invariant Mapping. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR'06)* (pp. 1735-1742). New York, NY, USA: IEEE. doi:10.1109/CVPR.2006.100

Han, K., Wang, Y., Tian, Q., Guo, J., Xu, C., & Xu, C. (2020, 6). GhostNet: More Features From Cheap Operations. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 1577-1586). Seattle, WA, USA: IEEE. doi:10.1109/CVPR42600.2020.00165

Haque, M., Marwaha, S., Arora, A., Deb, C., Misra, T., Nigam, S., & Hooda, K. (2022, 12). A lightweight convolutional neural network for recognition of severity stages of maydis leaf blight disease of maize. *Frontiers in Plant Science, 13*, 5252. doi:10.3389/fpls.2022.1077568

Haque, M., Marwaha, S., Deb, C., Nigam, S., & Arora, A. (2023, 4). Recognition of diseases of maize crop using deep learning models. *Neural Computing and Applications, 35*(10), 7407-7421. doi:10.1007/s00521-022-08003-9

Haque, M., Marwaha, S., Deb, C., Nigam, S., Arora, A., Hooda, K., . . . Agrawal, R. (2022, 4). Deep learning-based approach for identification of diseases of maize crop. *Scientific Reports, 12*(1), 6334. doi:10.1038/s41598-022-10140-z

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778). Las Vegas, NV, USA: IEEE.

Hernández, S., & López, J. (2020, 11). Uncertainty quantification for plant disease detection using Bayesian deep learning. *Applied Soft Computing, 96*, 106597. doi:10.1016/j.asoc.2020.106597

Howard, A., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., . . . Adam, H. (2017, 4). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint, arXiv: 1704.04861*, 1-9. Retrieved from http://arxiv.org/abs/1704.04861

Hu, J., Shen, L., & Sun, G. (2018, 6). Squeeze-and-Excitation Networks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 7132-7141). Salt Lake City, UT, USA: IEEE. doi:10.1109/CVPR.2018.00745

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. (2017, 7). Densely Connected Convolutional Networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2017-January*, pp. 2261-2269. Honolulu, HI, USA: IEEE. doi:10.1109/CVPR.2017.243

Huang, H., Lin, L., Tong, R., Hu, H., Zhang, Q., Iwamoto, Y., . . . Wu, J. (2020, 5). UNet 3+: A Full-Scale Connected UNet for Medical Image Segmentation. *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1055-1059). Barcelona, Spain: IEEE. doi:10.1109/ICASSP40776.2020.9053405

Hughes, D., & Salathe, M. (2015, 11). An open access repository of images on plant health to enable the development of mobile disease diagnostics. *arXiv, arXiv preprint*, 1-13. Retrieved from http://arxiv.org/abs/1511.08060

Janarthan, S., Thuseethan, S., Rajasegarar, S., & Yearwood, J. (2022, 11). P2OP—Plant Pathology on Palms: A deep learning-based mobile solution for in-field plant disease detection. *Computers and Electronics in Agriculture, 202*, 107371. doi:10.1016/j.compag.2022.107371

Ji, M., & Wu, Z. (2022, 2). Automatic detection and severity analysis of grape black measles disease based on deep learning and fuzzy logic. *Computers and Electronics in Agriculture, 193*, 106718. doi:10.1016/j.compag.2022.106718

Karthik, R., Hariharan, M., Anand, S., Mathikshara, P., Johnson, A., & R., M. (2020, 1). Attention embedded residual CNN for disease detection in tomato leaves. *Applied Soft Computing, 86*, 105933. doi:10.1016/j.asoc.2019.105933

Kaya, Y., & Gürsoy, E. (2023, 7). A novel multi-head CNN design to identify plant diseases using the fusion of RGB images. *Ecological Informatics, 75*, 101998. doi:10.1016/j.ecoinf.2023.101998

Khamparia, A., Saini, G., Gupta, D., Khanna, A., Tiwari, S., & de Albuquerque, V. (2020, 2). Seasonal Crops Disease Prediction and Classification Using Deep Convolutional Encoder Network. *Circuits, Systems, and Signal Processing, 39*(2), 818-836. doi:10.1007/s00034-019-01041-0

Khan, A., Nawaz, U., Kshetrimayum, L., Seneviratne, L., & Hussain, I. (2023, 12). Early and Accurate Detection of Tomato Leaf Diseases Using TomFormer. *21st International Conference on Advanced Robotics (ICAR)* (pp. 645-651). Abu Dhabi, United Arab Emirates: IEEE. doi:10.1109/ICAR58858.2023.10436499

Kingma, D., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *3rd International Conference on Learning Representations* (pp. 1-15). San Diego, CA, USA: DBLP.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (pp. 1097-1105). Red Hook, NY, USA: Curran Associates Inc.

Kumar, A., & Vani, M. (2019). Image Based Tomato Leaf Disease Detection. In *2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT)* (pp. 1-6). Kanpur, India: IEEE.

Kumar, P., Raghavendran, S., Silambarasan, K., Kannan, K., & Krishnan, N. (2023, 1). Mobile application using DCDM and cloud-based automatic plant disease detection. *Environmental Monitoring and Assessment, 195*(1), 44. doi:10.1007/s10661-022-10561-3

Lecun, Y. (1989). Generalization and network design strategies. *Connectionism in Perspective, 6*(19), 143-155.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*, 2278-2324.

Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., . . . Shi, W. (2017, 11). Photo-realistic single image super-resolution using a generative adversarial network. *30th IEEE Conference on Computer Vision and Pattern*

Recognition, CVPR 2017 (pp. 105-114). Honolulu, HI, USA: Institute of Electrical and Electronics Engineers Inc. doi:10.1109/CVPR.2017.19

Li, E., Wang, L., Xie, Q., Gao, R., Su, Z., & Li, Y. (2023, 7). A novel deep learning method for maize disease identification based on small sample-size and complex background datasets. *Ecological Informatics, 75*, 102011. doi:10.1016/j.ecoinf.2023.102011

Liang, Q., Xiang, S., Hu, Y., Coppola, G., Zhang, D., & Sun, W. (2019, 2). PD2SE-Net: Computer-assisted plant disease diagnosis and severity estimation network. *Computers and Electronics in Agriculture, 157*, 518-529. doi:10.1016/j.compag.2019.01.034

Liang, X. (2021, 12). Few-shot cotton leaf spots disease classification based on metric learning. *Plant Methods, 17*(1), 1-11. doi:10.1186/s13007-021-00813-7

Liu, Z., Mao, H., Wu, C.-Y., Feichtenhofer, C., Darrell, T., & Xie, S. (2022, 6). A ConvNet for the 2020s. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 11966-11976). New Orleans, LA, USA: IEEE. doi:10.1109/CVPR52688.2022.01167

Lu, X., Yang, R., Zhou, J., Jiao, J., Liu, F., Liu, Y., . . . Gu, P. (2022, 5). A hybrid model of ghost-convolution enlightened transformer for effective diagnosis of grape leaf disease and pest. *Journal of King Saud University - Computer and Information Sciences, 34*(5), 1755-1767. doi:10.1016/j.jksuci.2022.03.006

Madhu GS, Un Nabi, S., Iqbal Mir, J., & Hassan Raja, W. (2020, 8). Alternaria leaf and fruit spot in apple: Symptoms, cause and management. *European Journal of Biotechnology and Bioscience, 8*(3), 24-26. Retrieved from https://www.researchgate.net/publication/345063644

Mohameth, F., Bingcai, C., & Sada, K. (2020). Plant Disease Detection with Deep Learning and Feature Extraction Using Plant Village. *Journal of Computer and Communications, 08*(06), 10-22. doi:10.4236/jcc.2020.86002

Mohanty, S., Hughes, D., & Salathé, M. (2016, 11). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science, 7*, 1-10. doi:10.3389/fpls.2016.01419/

Mwebaze, E., & Owomugisha, G. (2016, 12). Machine Learning for Plant Disease Incidence and Severity Measurements from Leaf Images. *15th IEEE International Conference on Machine Learning and Applications (ICMLA)* (pp. 158-163). Anaheim, CA, USA: IEEE. doi:10.1109/ICMLA.2016.0034

Nabi, S., Parveen, S., Raja, W., Javid, I., Yasmin, S., Sheikh, M., & Sharma, O. (2022). Diseases of Apple (Malus domestica) and Their Management. *Agrica, 11*(1), 32-40. doi:10.5958/2394-448X.2022.00003.7

Naik, B., Malmathanraj, R., & Palanisamy, P. (2022, 7). Detection and classification of chilli leaf disease using a squeeze-and-excitation-based CNN model. *Ecological Informatics, 69*, 101663. doi:10.1016/j.ecoinf.2022.101663

Ng, H., Lin, C.-Y., Chuah, J., Tan, H., & Leung, K. (2021, 7). Plant Disease Detection Mobile Application Development using Deep Learning. *2021 International Conference on Computer & Information Sciences (ICCOINS)* (pp. 34-38). Kuching, Malaysia: IEEE. doi:10.1109/ICCOINS49721.2021.9497190

Ngugi, L., Abdelwahab, M., & Abo-Zahhad, M. (2020, 11). Tomato leaf segmentation algorithms for mobile phone applications using deep learning. *Computers and Electronics in Agriculture, 178*, 105788. doi:10.1016/j.compag.2020.105788

Nigam, S., Jain, R., Marwaha, S., Arora, A., Haque, M., Dheeraj, A., & Singh, V. (2023, 7). Deep transfer learning model for disease identification in wheat crop. *Ecological Informatics, 75*, 102068. doi:10.1016/j.ecoinf.2023.102068

Pal, A., & Kumar, V. (2023, 3). AgriDet: Plant Leaf Disease severity classification using agriculture detection framework. *Engineering Applications of Artificial Intelligence, 119*, 105754. doi:10.1016/j.engappai.2022.105754

Pan, J., Xia, L., Wu, Q., Guo, Y., Chen, Y., & Tian, X. (2022, 9). Automatic strawberry leaf scorch severity estimation via faster R-CNN and few-shot learning. *Ecological Informatics, 70*, 101706. doi:10.1016/j.ecoinf.2022.101706

Pandey, A., & Jain, K. (2022, 9). A robust deep attention dense convolutional neural network for plant leaf disease identification and classification from smart phone captured real world images. *Ecological Informatics, 70*, 101725. doi:10.1016/j.ecoinf.2022.101725

Patil, S., & Bodhe, S. (2011). Leaf Disease Severity Measurement using Image Processing. *International Journal of Engineering and Technology, 3*(5), 297-301.

Pedregosa, F., Varoquaux, G., Gramfort, A., & V., M. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825-2830. Retrieved from https://jmlr.csail.mit.edu/papers/volume12/pedregosa11a/pedregosa11a.pdf

Pietsch, M., Möller, T., Kostic, B., Risch, J., Pippi, M., Jobanputra, M., . . . Lee, S. (2019, 11 21). *Haystack: the end-to-end NLP framework for pragmatic builders*. Retrieved 05 20, 2024, from Haystack: https://github.com/deepset-ai/haystack

Pineda Medina, D., Miranda Cabrera, I., de la Cruz, R., Guerra Arzuaga, L., Cuello Portal, S., & Bianchini, M. (2024, 2). A Mobile App for Detecting Potato Crop Diseases. *Journal of Imaging, 10*(2), 47. doi:10.3390/jimaging10020047

Rauf, H., Saleem, B., Lali, M., Khan, M., Sharif, M., & Bukhari, S. (2019, 10). A citrus fruits and leaves dataset for detection and classification of citrus diseases through machine learning. *Data in Brief, 26*, 104340. doi:10.1016/j.dib.2019.104340

Ribeiro, M., Singh, S., & Guestrin, C. (2016, 8). "Why Should I Trust You?". *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 13-17-August-2016*, pp. 1135-1144. New York, NY, USA: ACM. doi:10.1145/2939672.2939778

Rimon, S., Islam, M., Dey, A., & Das, A. (2022). PlantBuddy: An Android-Based Mobile Application for Plant Disease Detection Using Deep Convolutional Neural Network. *ICRTAC: International Conference on Recent Trends in*

Advance Computing. 806, pp. 275-285. Chennai, India: Springer Science. doi:10.1007/978-981-16-6448-9_28

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018, 6). MobileNetV2: Inverted Residuals and Linear Bottlenecks. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4510-4520). Los Alamitos, CA, USA: IEEE. Retrieved from https://ieeexplore.ieee.org/document/8578572/

Sanga, S., Machuve, D., & Jomanga, K. (2020, 6). Mobile-based Deep Learning Models for Banana Disease Detection. *Engineering, Technology & Applied Science Research, 10*(3), 5674-5677. doi:10.48084/etasr.3452

Schroff, F., Kalenichenko, D., & Philbin, J. (2015, 6). FaceNet: A unified embedding for face recognition and clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 815-823). Boston, MA, USA: IEEE. doi:10.1109/CVPR.2015.7298682

Sethy, P., Negi, B., Barpanda, N., Behera, S., & Rath, A. (2018). Measurement of Disease Severity of Rice Crop Using Machine Learning and Computational Intelligence. *Cognitive Science and Artificial Intelligence* (pp. 1-11). Singapore: Springer. doi:10.1007/978-981-10-6698-6_1

Sharma, V., Tripathi, A., & Mittal, H. (2023, 7). DLMC-Net: Deeper lightweight multi-class classification model for plant leaf disease detection. *Ecological Informatics, 75*, 102025. doi:10.1016/j.ecoinf.2023.102025

Shrimali, S. (2021, 8). PlantifyAI: A Novel Convolutional Neural Network Based Mobile Application for Efficient Crop Disease Detection and Treatment. *International Workshop on Edge IA-IoT for Smart Agriculture (SA2IOT). 191*, pp. 469-474. Leuven, Belgium: Elsevier. doi:10.1016/j.procs.2021.07.059

Shrivastava, V., & Pradhan, M. (2020, 2). Rice plant disease classification using color features: a machine learning paradigm. *Journal of Plant Pathology, 103*(1), 17-26. doi:10.1007/s42161-020-00683-3

Shruthi, U., Nagaveni, V., & Raghavendra, B. (2019, 3). A Review on Machine Learning Classification Techniques for Plant Disease Detection. *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)* (pp. 281-284). Coimbatore, India: IEEE. doi:10.1109/ICACCS.2019.8728415

Shu, H., Liu, J., Hua, Y., Chen, J., Zhang, S., Su, M., & Luo, Y. (2023, 6). A grape disease identification and severity estimation system. *Multimedia Tools and Applications, 82*(15), 23655-23672. doi:10.1007/s11042-023-14755-w

Si, C., Yu, W., Zhou, P., Zhou, Y., Wang, X., & Yan, S. (2022, 5). Inception Transformer. *36th Conference on Neural Information Processing Systems. arXiv Preprint*, pp. 1-17. Los Angels, USA: Curran Associates, Inc. Retrieved from http://arxiv.org/abs/2205.12956

Siddiqua, A., Kabir, M., Ferdous, T., Ali, I., & Weston, L. (2022, 8). Evaluating Plant Disease Detection Mobile Applications: Quality and Limitations. *Agronomy, 12*(8), 1869. doi:10.3390/agronomy12081869

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (pp. 770-778). San Diego, CA, USA: IEEE. Retrieved from http://arxiv.org/abs/1409.1556

Snell, J., Swersky, K., & Zemel, R. (2017, 3). Prototypical Networks for Few-shot Learning. *31st International Conference on Neural Information Processing Systems* (pp. 4080-4090). Red Hook, NY, USA: Curran Associates Inc. doi:10.5555/3294996.3295163

Song, Y., Wang, T., Cai, P., Mondal, S., & Sahoo, J. (2023, 12). A Comprehensive Survey of Few-shot Learning: Evolution, Applications, Challenges, and Opportunities. *ACM Computing Surveys, 55*(13s), 1-40. doi:10.1145/3582688

Sun, S., & Gao, H. (2023). Meta-AdaM: An Meta-Learned Adaptive Optimizer with Momentum for Few-Shot Learning. *Thirty-seventh Conference on Neural*

Information Processing Systems (pp. 1-15). New Orleans, USA: OpenReview.Net. Retrieved from https://openreview.net/forum?id=d85pPNBHLt

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., . . . Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1-9). Boston, MA, USA: IEEE.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016, 6). Rethinking the Inception Architecture for Computer Vision. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 2818-2826). Las Vegas, NV, USA: IEEE. doi:10.1109/CVPR.2016.308

Tan, M., & Le, Q. (2021). EfficientNetV2: Smaller Models and Faster Training. In M. Meila, & T. Zhang (Ed.), *Proceedings of the 38th International Conference on Machine Learning* (pp. 10096-10106). Organized in Virtual Mode: ACM Digital.

Tassis, L., & Krohling, R. (2022, 1). Few-shot learning for biotic stress classification of coffee leaves. *Artificial Intelligence in Agriculture, 6*, 55-67. doi:10.1016/j.aiia.2022.04.001

Tembhurne, J., Gajbhiye, S., Gannarpwar, V., Khandait, H., Goydani, P., & Diwan, T. (2023, 7). Plant disease detection using deep learning based Mobile application. *Multimedia Tools and Applications, 82*(18), 27365-27390. doi:10.1007/s11042-023-14541-8

Thai, H.-T., Tran-Van, N.-Y., & Le, K.-H. (2021, 10). Artificial Cognition for Early Leaf Disease Detection using Vision Transformers. *2021 International Conference on Advanced Technologies for Communications (ATC). 2021-October*, pp. 33-38. Ho Chi Minh City, Vietnam: IEEE. doi:10.1109/ATC52653.2021.9598303

Thapa, R., Zhang, K., Snavely, N., Belongie, S., & Khan, A. (2020, 9). The Plant Pathology Challenge 2020 data set to classify foliar disease of apples. *Applications in Plant Sciences, 8*(9), e11390. doi:10.1002/aps3.11390

Tiwari, D., Ashish, M., Gangwar, N., Sharma, A., Patel, S., & Bhardwaj, S. (2020, 5). Potato Leaf Diseases Detection Using Deep Learning. *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)* (pp. 461-466). Madurai, India, India: IEEE. doi:10.1109/ICICCS48265.2020.9121067

Tiwari, V., Joshi, R., & Dutta, M. (2021, 7). Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images. *Ecological Informatics, 63*, 101289. doi:10.1016/j.ecoinf.2021.101289

Tomar, N. S. (2023, 3). *Contribution of Agricultural Sector in GDP.* Minister of Agriculture and Farmers Welfare. Delhi: Press Information Bureau. Retrieved 2 20, 2024, from https://pib.gov.in/PressReleaseIframePage.aspx?PRID=1909213

Trivedi, J., Shamnani, Y., & Gajjar, R. (2020). Plant Leaf Disease Detection Using Machine Learning. *Third International Conference on Emerging Technology Trends in Electronics Communication and Networking. 1214 CCIS*, pp. 267-276. Surat, India: Springer, Singapore. doi:10.1007/978-981-15-7219-7_23

Varshney, D., Babukhanwala, B., Khan, J., Saxena, D., & Singh, A. (2021, 6). Machine Learning Techniques for Plant Disease Detection. *2021 5th International Conference on Trends in Electronics and Informatics (ICOEI)* (pp. 1574-1581). Tirunelveli, India: IEEE. doi:10.1109/ICOEI51242.2021.9453053

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., . . . Polosukhin, I. (2017). Attention Is All You Need. In I. Guyon and U. V. Luxburg and S. Bengio and H. Wallach and R. Fergus and S. Vishwanathan and R. Garnett (Ed.), *31st Conference on Neural Information Processing Systems* (pp. 1-11). California, USA: Curran Associates, Inc. Retrieved from https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

Verma, S., Chug, A., Singh, A., & Singh, D. (2023, 9). PDS-MCNet: a hybrid framework using MobileNetV2 with SiLU6 activation function and capsule

networks for disease severity estimation in plants. *Neural Computing and Applications, 35*(25), 18641-18664. doi:10.1007/s00521-023-08693-9

Wang, C., Du, P., Wu, H., Li, J., Zhao, C., & Zhu, H. (2021, 10). A cucumber leaf disease severity classification method based on the fusion of DeepLabV3+ and U-Net. *Computers and Electronics in Agriculture, 189*, 106373. doi:10.1016/j.compag.2021.106373

Wang, G., Sun, Y., & Wang, J. (2017). Automatic Image-Based Plant Disease Severity Estimation Using Deep Learning. *Computational Intelligence and Neuroscience, 2017*, 1-8. doi:10.1155/2017/2917536

Wang, Y., Yao, Q., Kwok, J., & Ni, L. (2020, 5). Generalizing from a Few Examples. *ACM Computing Surveys, 53*(3), 1-34. doi:10.1145/3386252

Weng Lilian. (2018, Novemeber 30). *Meta-Learning: Learning to Learn Fast*. Retrieved January 25, 2024, from GitHub.io: https://lilianweng.github.io/posts/2018-11-30-meta-learning/

Wspanialy, P., & Moussa, M. (2020, 11). A detection and severity estimation system for generic diseases of tomato greenhouse plants. *Computers and Electronics in Agriculture, 178*, 105701. doi:10.1016/j.compag.2020.105701

Wu, S., Sun, Y., & Huang, H. (2021, 12). Multi-granularity Feature Extraction Based on Vision Transformer for Tomato Leaf Disease Recognition. *2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST)* (pp. 387-390). Guangzhou, China: IEEE. doi:10.1109/IAECST54258.2021.9695688

Xiang, S., Liang, Q., Sun, W., Zhang, D., & Wang, Y. (2021, 4). L-CSMS: novel lightweight network for plant disease severity recognition. *Journal of Plant Diseases and Protection, 128*(2), 557-569. doi:10.1007/s41348-020-00423-w

Yulita, I., Amri, N., & Hidayat, A. (2023, 1). Mobile Application for Tomato Plant Leaf Disease Detection Using a Dense Convolutional Network Architecture. *Computation, 11*(2), 20. doi:10.3390/computation11020020

Zaki, M. J., & Wagner, M. J. (2020). *Data Mining and Machine Learning: Fundamental Concepts and Algorithms* (2nd ed.). Cambridge, England: Cambridge University Press. Retrieved from https://dataminingbook.info/book_html/

Zhang, Y., Wa, S., Zhang, L., & Lv, C. (2022, 5). Automatic Plant Disease Detection Based on Tranvolution Detection Network With GAN Modules Using Leaf Images. *Frontiers in Plant Science, 13*, 1-20. doi:10.3389/fpls.2022.875693

Zhao, Y., Chen, J., Xu, X., Lei, J., & Zhou, W. (2021, 5). SEV-Net: Residual network embedded with attention mechanism for plant disease severity detection. *Concurrency and Computation: Practice and Experience, 33*(10), 1-18. doi:10.1002/cpe.6161

Zhao, Y., Chen, Z., Gao, X., Song, W., Xiong, Q., Hu, J., & Zhang, Z. (2021). Plant Disease Detection using Generated Leaves Based on DoubleGAN. *IEEE/ACM Transactions on Computational Biology and Bioinformatics, 19*(3), 1817-1826. doi:10.1109/TCBB.2021.3056683

Zhao, Y., Sun, C., Xu, X., & Chen, J. (2022, 2). RIC-Net: A plant disease classification model based on the fusion of Inception and residual structure and embedded attention mechanism. *Computers and Electronics in Agriculture, 193*, 106644. doi:10.1016/j.compag.2021.106644

Zhong, Y., & Zhao, M. (2020). Research on deep learning in apple leaf disease recognition. *Computers and Electronics in Agriculture, 168*, 1-6. doi:10.1016/j.compag.2019.105146

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. (2018, 6). Learning Transferable Architectures for Scalable Image Recognition. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 8697-8710). Salt Lake City, UT, USA: IEEE. doi:10.1109/CVPR.2018.00907

Appendix A

Asymptomatic analysis on the number of weight parameters used by the Inception module

Let the input to the Inception module is $I \in \mathbb{R}^{M \times N}$ containing M patches, and each patch is of size N. Since the Inception module uses two-dimensional convolution operations, therefore I must be reshaped to $I' \in \mathbb{R}^{\sqrt{N} \times \sqrt{N} \times M}$, assuming that the value of N is a perfect square (i.e., $N \in 2^{2i}, i = 1,2,3, \ldots$). The asymptomatic analysis is done on each operation present in the Inception module (shown by numbers in Figure A.1).

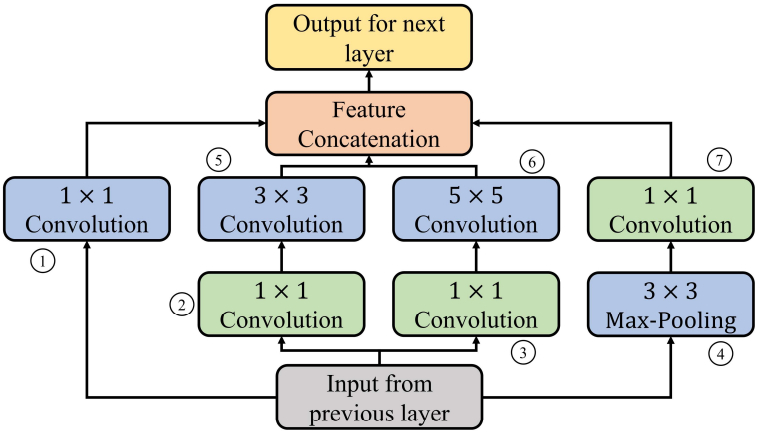


Figure A.1: Operations of Inception module

The number of weight parameters used in convolution operation can be expressed by equation A.1.

$$W = I_F \times D \times k \times k \tag{A.1}$$

where:

W: number of weight parameters used in convolution operation

I_F: number of filters applied

D: depth of input feature map

k: convolution filter's size

Let the number of filters used in operation i is F_i, where $i = 1,2, \dots, 7$ (shown in Figure A.1). Then the trainable number of weight parameters used in each operation is computed in Table A.1.

Table A.1: Trainable weight parameters used in each operation of the Inception module

Operation number	**Input shape**	**Output shape**	**Number of weight parameters (asymptomatically)**
Operation 1	$\sqrt{N} \times \sqrt{N} \times M$	$\sqrt{N} \times \sqrt{N} \times F_1$	$F_1 \times M \times 1 \times 1 = F_1 \times M = \mathcal{O}(F_1 M)$
Operation 2	$\sqrt{N} \times \sqrt{N} \times M$	$\sqrt{N} \times \sqrt{N} \times F_2$	$F_2 \times M \times 1 \times 1 = F_2 \times M = \mathcal{O}(F_2 M)$
Operation 3	$\sqrt{N} \times \sqrt{N} \times M$	$\sqrt{N} \times \sqrt{N} \times F_3$	$F_3 \times M \times 1 \times 1 = F_3 \times M = \mathcal{O}(F_3 M)$
Operation 4	$\sqrt{N} \times \sqrt{N} \times M$	$\sqrt{N} \times \sqrt{N} \times M$	0 (max-pooling operation does not require any weight parameters)
Operation 5	$\sqrt{N} \times \sqrt{N} \times F_2$	$\sqrt{N} \times \sqrt{N} \times F_5$	$F_2 \times F_5 \times 3 \times 3 = 9 \times F_2 \times F_5 = \mathcal{O}(F_2 F_5)$
Operation 6	$\sqrt{N} \times \sqrt{N} \times F_3$	$\sqrt{N} \times \sqrt{N} \times F_6$	$F_3 \times F_6 \times 5 \times 5 = 25 \times F_3 \times F_6 = \mathcal{O}(F_3 F_6)$
Operation 7	$\sqrt{N} \times \sqrt{N} \times M$	$\sqrt{N} \times \sqrt{N} \times F_7$	$F_7 \times M \times 1 \times 1 = F_7 \times M = \mathcal{O}(F_7 M)$

By combining the number of weight parameters used in each operation of the Inception module (given in Table A.1), the total number of weight parameters used in this module can be calculated as $\mathcal{O}(F_1 M + F_2 M + F_3 M + F_2 F_5 + F_3 F_6 + F_7 M)$. If $F = \max(F_1, F_2, F_3, F_5, F_6, F_7)$, then the above expression can be simplified to $\mathcal{O}(4 \times FM + 2 \times F^2) \Rightarrow \mathcal{O}(FM + F^2)$. Hence, the total number of trainable weight parameters required to implement an inexpensive Inception module can be expressed by $W_{Inception}$ in equation A.2.

$$W_{Inception} = \max(M^2, F^2) = \begin{cases} \mathcal{O}(M^2), & F < M \\ \mathcal{O}(F^2), & F \geq M \end{cases} \quad \text{(A.2)}$$