MEKELLE UNIVERSITY
Ethiopian Institute of Technology-Mekelle
School of Computing
Computer Science Department

Enhanced Inception ResNet V2 Model for Grape Leaf Disease Detection and Classification

By

Efrem Gebrewahd Gebreslassie

A Thesis
Submitted in Partial Fulfilment of The Requirements for The Master of Science Degree in Computer Science

Advisor: Mohammed Ahmed (PhD)

January 2025
Mekelle, Ethiopia

MEKELLE UNIVERSITY
Ethiopian Institute of Technology-Mekelle
School of Computing
Computer Science Department

CERTIFICATION

This is to certify that the thesis entitled "Enhanced Inception ResNet V2 Model for Grape Leaf Disease Detection and Classification" is submitted in partial fulfillment of the requirements for the degree of Masters in computer Science and has been carried out by **Efrem Gebrewahd Gebreslassie**, **ID No: Eitm/pr175706/12** under my supervision. Therefore, I recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the Department.

Mohammed Ahmed (ph.D.)		07/04/2025
Name of Major Advisor	Signature	Date

DECLARATION

I hereby declare that this Masters thesis is my original work and has not been presented for a degree in any other university and all sources of material used for this thesis have been duly acknowledged.

Name: Efrem Gebrewahd Gebreslassie

Signature: _____

Date: 07/04/2025

This Masters thesis has been submitted for examination with my approval as thesis advisor.

Name: Mohammed Ahmed (ph.D.)

Signature: _____

Date: 07/04/2025

EXAMINERS' APPROVAL SHEET

We, the undersigned, members of the Board of Examiners of the final open defense by **Efrem Gebrewahd Gebreslassie**, have read and evaluated his thesis **"Enhanced Inception ResNet V2 Model for Grape Leaf Disease Detection and Classification"** and evaluated the candidate. This is therefore to certify that the thesis has been accepted in partial fulfillment of the requirements for the Masters Degree in Computer Science.

_____	_____	_____
Name of Chairperson	Signature	Date
Mohammed Ahmed (ph.D.)	_____	07/04/2025
Name of Major Advisor	Signature	Date
Guesh Dagnew (Ph.D.)	_____	07/04/2025
Name of Internal Examiner	Signature	Date
Solomon Gebremariam (ph.D.)	_____	07/04/2025
Name of External Examiner	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the candidate's Department through the office of the Department Graduate Program Coordinator. Thesis Approved by:

Ferede Ali		07/04/2025
Graduate Program Coordinator		Date

Certification of the Final Thesis

I hereby certify that all the corrections and recommendations suggested by the Board of Examiners are incorporated into the final thesis entitled “Enhanced Inception Res Net V2 Model for Grape Leaf Disease Detection and Classification” by Efrem Gebrewahd Gebreslassie.

ሰለሞን ገብረመስቀል አዳነ
Solomon Gebremeskel Adane
የኮምፒዩቲንግ ት/ቤት ዲን
School of Computing Dean

Department Head Signature 07/04/2025 Date

Acknowledgements

First and foremost, I would like to express my sincere thanks to my advisor, Mohammed Ahamed (PhD). The completion of this thesis without my advisor would not have been possible and I am truly grateful for his time, helpful feedback and consistent support which plays critical role in shaping my work. Finally, my appreciation goes to my family and friends who have been by my side whenever I needed assistance.

Abstract

Grapes are one of the most widely consumed and globally traded crops, benefiting both agricultural economy and healthy wise. However, their vulnerability to diseases can negatively affect the quality and quantity of the grapes being produced. The most common way of detecting and classifying these disease is through the use of human experts (manual inspection mehtod), such as pathologists and botanists. This manual inspection method is prone to error, time consuming and inefficient for large scale farms. To tackle this problem several researchers have built an automated system that detects and classifies plant diseases in a more accurate and faster way. While these studies show a reasonable result but still struggle with several issues, like poor accuracy, increased computational complexity and strong overfitting. Thus, our model addressed these issues by introducing an enhanced version of Pretrained Inception ResNet V2 architecture, By integrating a lightweight Reduction Module C which is responsible for reducing the grid size of the input image from 8 X 8 to 3 X 3 while increasing the feature maps from 1536 to 1600. This module allows the model to capture more complex features while ignoring relevant information leading to improved feature extraction capability. The proposed model achieved a superior F1 score of 99.89% on the validation set with only 0.21 million trainable parameters, outperforming EfficientNet-B4 (99.81% F1 score, 0.46 million parameters), Xception (99.73% F1 score, 1.05 million parameters), and the baseline Inception ResNet V2 (99.87% F1 score, 0.79 million parameters) in both accuracy and computational efficiency. The results show that the suggested model presents a promising solution for accurate and efficient grape leaf disease detection and classification.

Keywords

Grape leaf diseases, disease classification, image processing, machine learning, convolutional neural networks (CNN), Inception ResNet V2, EfficentNet-B4, Xception, reduction module C.

List of Abbreviations and Acronyms

CapsNet	Capsule Neural Network
CBAM	Convolution Block Attention Module
CNN	Convolutional Neural Network
DR-IACNN	Deep Residual–Inception Attention Convolutional Neural Network
FPS	Frames Per Second
GLCM	Gray Level Co-occurrence Matrix
GLDD	Grape Leaf Disease Dataset
GSMo-CNNs	Generalized Stacking Multi-output CNNs
MAP	Mean Average Precision
RFFBs	Residual Feature Fusion Blocks
RMSprob	Root Mean Square Propagation
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine

Table of Contents

List of Tables

List of Figures

Chapter One

1. Introduction

1.1 Background of Study

Grapes due to their huge contribution in agriculture income and due to their richer nutrient content, are valuable both economically and health wise. Therefore, growing grapes is one of the most useful forms of agriculture. In the world, grapes come in over 10,000 varieties. Popular grape varieties include Vitis vinifera, which is mostly produced for wine, Vitis labrusca, which is typically grown for fresh juice, and Vitis rotundifolia, which is well-known for its hardiness. Because of their variety, grapes can be used for a wide range of purposes and have a wide range of flavors and nutritional characteristics [1].

However, various diseases extremely reduce the quality and the quantity of grapes being produced, causing direct impact on the agricultural economy. And this study specifically focused on three common and impactful diseases, namely black rot, black measles and leaf blight. Black rot is a fungal disease affecting grapevines, caused by the pathogen Guignardia bidwellii. It is a significant threat to viticulture, continuing to arise as a problem in various regions [2]. Black measles, also known as Esca, foliar symptoms of esca start to appear as chlorotic spots that eventually merge to turn a dark red color. Depending on the grape type, these patches eventually turn necrotic, with the dead tissue appearing dark brown to red brown. The symptoms frequently extend to the areas between the leaf veins, resulting in a thin margin of healthy tissue running parallel to the main veins, giving the leaves their characteristic tiger-stripe pattern [3]. Leaf blight is caused by Isariopsis clavispora, appears as brownish lesions on the leaves that have the potential to combine and seriously defoliate the leaves [4].

So, continuous monitoring and identification of the type of disease is required to take proper action for controlling it before it causes huge damage. To do that, farmers and industries working with the production of grapes use a traditional way of using human experts such as botanists and pathologies in identifying types of disease. Of course, this

could help for small scale farms, but this will be a very tedious and time-consuming method for large scale farms.

To address this problem various research used machine learning techniques. Machine learning is a branch of artificial intelligence that enables computers to learn from and make predictions based on data. By enabling fast and precise detection of diseases, machine learning plays a critical role in agriculture by helping farmers manage their crops more effectively and increasing crop yield [5]. Some of the machine learning techniques used for plant disease detection and classification include Support Vector Machine (SVM), Artificial Neural Networks (ANN), K-Nearest Neighbor (KNN) and Convolutional Neural Network (CNN) [5]**.** CNN is one of the most famous and commonly used algorithms [6]. Key benefits of CNN involve improving overall accuracy, automatic feature extraction from the given image without human supervision, able to process large datasets, sparsity of the connection, and parameter sharing [7]. Due to the above listed key benefits of CNN various CNN architectures have been introduced enabling the detection and classification of plant disease simpler and faster.

To identify the best performing model for grape leaf disease detection and classification, eighteen (18) models were developed under different hyperparameter configurations using pre-trained CNN architectures namely EfficientNet-B4, Xception, and Inception ResNet V2. Following careful evaluation on the validation set, Inception ResNet V2 was chosen to be the most efficient model. The Inception ResNet V2 architecture was then enhanced further by the integration of a new reduction module, called Reduction Module C. With this modification, the model's efficiency and accuracy in identifying grape leaf diseases increased, providing a more accurate method for automated disease detection and classification.

1.2 Motivation

One of the most challenging factors that can reduce the quality and quantity grapes being produced is disease. manual inspection methods used for monitoring and identifying these diseases are time consuming, require human expertise , prone to error and are inefficient for large scale farms. To enhance disease management and reduce economic losses, an

automated system that can accurately detect and classify grape leaf diseases is desperately needed.

Using the current manual method the demands of large-scale grape production industries for monitoring and identifying grape leaf disease in an easier and faster way cannot be addressed, as they are laborious and prone to errors. We were then motivated to conduct this study with the goal of building an automated system that detects and classifies grape leaf diseases in a more accurate and efficient manner, helping the grape industry manage diseases in a faster and easier way resulting in maximized production of crops.

Existing researches on plant disease detection and classification despite their advancements in machine learning, they still struggle with several issues like accuracy problems, computational complexity and overfitting. These challenges limit the practical use of these models in accurately detecting and classifying grape leaf diseases. This study aims at improving accuracy, mitigating overfitting and reducing computational complexity by using an enhanced pre-trained Inception-ResNet-v2.

This study is relevant in two ways. Initially, it advances the field of machine learning by presenting an enhanced version of Inception ResNet V2 model that is highly accurate and computationally efficient, making it suitable for resource constrained devices and for practical use in the agricultural environment. Second, it offers an automated system that detects and classifies grape leaf diseases, which makes it extremely relevant to the agricultural sector, especially for large-scale grape production. This research could change how diseases are managed, which would benefit the grape sector by producing more profitable and sustainable results.

As previously discussed, this study contributes to machine learning by presenting an enhanced version of Inception ResNet V2 pretrained model that includes the new module called Reduction Module C. This module is responsible for reducing the grid size of the input, while increasing the number of feature maps. This adjustment allows the model to focus on the most important features to the detection and classification of grape leaf diseases, while learning a wide range of detailed information from the input. In general, this study offers a more accurate and efficient automated system, making it suitable for practical applications in the real-world scenarios.

Having achieved these remarkable benefits, there are some key challenges We faced during the development of an accurate and efficient model for detecting and classifying grape leaf disease. The first one is balancing accuracy with computational efficiency. The addition of a Reduction Module C to the pre-trained CNN architecture, while intended to enhance accuracy, also introduces increased computational complexity. This research was primarily concerned with making sure that this module improves accuracy without making the model excessively complex. The second one is determining the optimal design for Reduction Module C. To find a design that successfully improves performance while maintaining computational demands manageable requires an iterative experimentation.

By addressing these challenges, this study produced a highly accurate and computationally efficient model that will give grape growers a useful tool for detecting and managing grape leaf diseases in an easier and faster way.

1.3 Problem Statement

Most of the time plant diseases occur on the leaf part of the plant [8], this forces researchers to tackle the manual inspection method's problem by proposing an automated system using image processing and machine learning techniques. Which is capable of accepting plant leaf images as an input and generates the corresponding disease type as an output. However, these systems still suffer with issues like overfitting, poor accuracy, and high computational demands. These issues are mostly caused by the way that current approaches rely on handcrafted methods, small datasets, and irrelevant features [9-24]. For this reason, it is difficult to apply these systems into the real world.

Therefore, our study addresses these issues by introducing an enhanced version of pretrained Inception ResNet V2 CNN architecture. Which integrates a lightweight Reduction Module C into the architecture resulting in improved accuracy while reducing computational complexity. This makes it suitable for resource constrained devices and for practical use in the agricultural sector, making continuous monitoring and identification of grape leaf disease easier and faster for large scale farms.

1.4 Research questions

The research questions that are addressed in this study are:

Research question 1: Which existing models are more suitable and how to enhance from existing one?

Research question 2: what are the techniques used to determine the principal features that have significant impact on the detection and classification of grape leaf diseases?

Research question 3: What are the principal features that have a significant impact on the detection and classification of grape leaf diseases?

Research question 4: what are the techniques used to measure the performance of the model?

1.5 Objectives

1.5.1 General objectives

The general objective of our study is to build a more accurate and efficient model that detects and classifies grape leaf disease using enhanced pretrained Inception ResNet V2 CNN architecture.

1.5.2 Specific objectives

To achieve the general objective of the study, this study includes the following specific objectives:

- To review existing research focusing on the strength and limitation of each key approaches.
- To collect sets of grape leaf images covering four classes from Kaggle.
- To preprocess the datasets to prepare the data for further training and evaluation.
- To build several models using pretrained CNN architectures to identify the best performing pretrained model.
- To choose pre-trained Inception ResNet V2 architecture as the primary model for the detection and classification of grape leaf disease.
- To enhance the performance of baseline model by integrating a lightweight Reduction Module C into the pretrained Inception ResNet V2 CNN architecture.

- To determine key features that have significant impact on the detection and classification of grape leaf diseases.
- To test and evaluate performance of the enhanced Inception ResNet V2 model.
- To Compare and analyze the improved model's performance with cutting-edge models.

1.6 Scope and Limitation

1.6.1 Scope

- This study focused on developing eighteen different models on different hyperparameter configurations that accurately detect and classify grape leaf disease using pre-trained CNN architecture such as EfficientNet-B4, Xception, and Inception ResNet V2, where further comparison between these models were made for selecting the best performing pre-trained CNN architecture.
- Upon selecting the best performing CNN architecture, this study is followed by modifying structural form of the pre-trained architecture specifically Inception ResNet V2 as it is the one performing well than the others.
- The model’s performance was then tested and evaluated using metrics like confusion matrix, accuracy, precision, recall and F1 score.

1.6.2 Limitation

- The target object to be detected and classified by this model is limited to grape leaf disease.
- The model is limited to detecting and classifying grape leaf diseases which are seen most of the time.
- The scope does not involve the development of a new CNN architecture from scratch, it utilizes pre-trained CNN architecture.

1.6.3 Constraints Affecting Scope, Validity, and Generalizability

- Dataset: the model is trained using only grape leaf dataset covering only four types of classes, this usage of limited dataset could directly or indirectly affect the model’s scope, validity, and generalizability.

- Image quality: one of the most challenging factors when applying our model into the real world is condition such as lighting variation, background noise and inconsistency in resolution. This could slightly affect the performance of the model.

1.7 Significance of study

This study introduces an extended version of existing technology to the agricultural field of study. In Ethiopia there are various grape production industries providing grape in the form of fruit or liquor to local and global markets. However, the overall grape production situation in Ethiopia has very little impact on the global grape trade. This is due to the lack of improved technologies and less attention given for research. This study targets Ethiopian grape production industries providing well-behaved performance automated system useful for timely and accurate detection and classification of grape leaf disease. This makes continuous monitoring and identifying grape plant disease easier and faster for large scale farms.

1.8 Organization of thesis

The thesis is structured as follows, chapter one provides an overview of the study's background, problem statement, objective, research questions, scope and limitation and significance. Chapter two reviews several existing research focusing on the strengths and limitations of each key approach. Chapter three discusses the tools we used and the steps we followed to build our system. Chapter four presents the result of the study. Finally, chapter five provides detailed answers to each research question and focuses on our future implementation.

Chapter Two

2. Literature review

The most challenging issue in agriculture is plant disease. This significantly reduces the quality and the quantity of the crops being produced, leading to economic losses. To identify these diseases, farmers most commonly apply manual inspection methods, which is time consuming, prone to error and inefficient for large scale farm. Here, we discuss several related studies that address the issue of manually detecting and classifying plant leaf diseases. This section is organized into two sub sections. In the first section, traditional machine learning methods are covered and in the second section, modern machine learning techniques are discussed with particular focus on deep learning.

Traditional methods are based on image processing techniques and handcrafted features. Image processing techniques work by applying segmentation, which separates diseased area of the leaf from the healthy one and feature extraction (identifying the most discriminative patterns like texture, shape and color) using Gray Level Co-occurrence Matrix (GLCM) and K-means clustering [9,10]. These features are then given as an input to a machine learning algorithm like SVM or ANN for classification. However, this method still struggles with several issues like poor accuracy and the requirement of domain expertise useful for extracting the right features.

In contrast deep learning methods, like CNN which allow us to use pretrained models to a new task, capture the most relevant features automatically and perform both feature extraction and image classification tasks into single processes, improving accuracy, saving time and resource usage. For example, [11,17] use pretrained models and adapt it for new tasks, avoiding the need for training the model from scratch saving time and resource consumption. While these researches provides a more promising result but still struggle with issues like increased computational cost and overfitting issues.

Detailed discussion about each related study focusing on the strength and limitations of each approach is provided below.

2.1 Traditional Machine Learning approaches

- **Unhealthy Region of Citrus Leaf Detection Using Image Processing Techniques [9]**

  This research built a model that detects and classify citrus leaf diseases using image processing and machine learning methods. This study first enhances the quality of the leaf image, then segments diseased portion of the leaf image from the healthy one using K-means clustering and use Gray-Level Co-occurrence Matrix (GLCM) to extract texture information. Finally, the extracted features are then fed to a machine learning algorithm called a Support Vector Machine (SVM) for image classification. Even though this model achieved a higher accuracy of 96% it still struggles with high processing requirements and dependence on manually created features limit its performance and can make large-scale implementation challenging. The requirement for manual parameter selection in GLCM increases bias risk and decreases generalization to other diseases.

- **A Smartphone Application to Detection and Classification of Coffee Leaf Miner and Coffee Leaf Rust [10]**

  A smartphone app that detects coffee leaf diseases is introduced by this study. The study applied Ostu approach to separate diseased area of the leaf from the healthy one, it then apply GLCM and additional statistical techniques to extract texture and color features and ANN is used to classify the disease type based on the extracted features. The app demonstrates great promise, with accuracy rates ranging from 91.5% to 99.1%. However, rely on manually chosen features, which might not fully reflect the unique characteristics of diseases. Furthermore, GLCM feature computation can be tedious and time-consuming, particularly for large images.

2.2 Modern Machine Learning approaches

- **A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks [11]**

  This study used a CNN architecture, which is composed of ResNet34 with SE blocks, Inception-V1 module, and Inception ResNet v2 called Deep Residual–Inception Attention Convolutional Neural Network (DR-IACNN) to build a model. This model

was built to detect grape leaf diseases as it was trained using the grape leaf disease dataset (GLDD). The model had a detection performance of 81.1% MAP (mean Average Precision) with detection speed of 15.01 frames per second (FPS). However, the effectiveness of the model in real world application is limited due to its lower accuracy value.

- **Transfer learning with Densenet201 architecture model for potato leaf disease classification [12]**

  This study presented a model that classifies potato leaf disease that is built using pretrained DenseNet201 CNN architecture. This model achived training accuracy of 99.5%, validation accuracy of 95.2, and 92.5 test accuracy. While the model shows higher accuracy but its increased gap between training and validation accuracy indicates a strong overfitting issue.

- **Deep Learning for Plant Identification and Disease Classification from Leaf Images [13]**

  This research presents a strategy that uses Generalized Stacking Multi-output CNNs (GSMo-CNNs) for plant identification and disease classification. The GSMo-CNNs design improves prediction accuracy for both plant species and 20 disease categorizations in a single run by stacking multiple CNN models and integrating their outputs. This method efficiently mixes model outputs to improve feature extraction and classification, achieving a 99.31% accuracy on benchmark datasets. Further investigation is necessary to determine the degree to which this methodology adapts to a range of agricultural situations and distinct imaging contexts, as this could potentially affect its practical effectiveness.

- **Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks [14]**

  This work presents a memory-efficient, two-stage CNN model for rice disease and pest identification that has been optimized for mobile device deployment and inspired by VGG16. The architecture, which achieves an accuracy of 93.3% on rice datasets, is made to reduce model size while keeping effective detection capabilities, with only 0.8 million parameters. The two-stage training method, however, involves manually

classifying the dataset into symptom groups, which can be time-consuming and run the risk of skipping some symptom changes in big datasets.

- **Using Deep Learning for Image-Based Plant Disease Detection [15]**

  Using 54,306 images from the PlantVillage dataset which includes 38 crop-disease pairs, the study builds a model to identify the crop and disease from a picture of a plant leaf. The research finds that an 80%–20% training–testing split using GoogleNet combined with transfer learning on color images produce an accuracy of 99.35% after analyzing 60 experimental configurations. But when the model was tested on images that were captured in different circumstances than the ones it was trained on, its accuracy dropped significantly to about 31%. This suggests a serious overfitting problem.

- **ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network [16]**

  The research created a CNN-based algorithm to identify tomato plant diseases. The design consists of an output layer, a fully connected layer, and three convolutional layers, each followed by a pooling layer. Ten categories make up this layer: one represents healthy tomato leaves, while the other nine indicate specific tomato diseases, including Target Spot, Mosaic Virus, Bacterial Spot, Late Blight, Leaf Mold, Yellow Leaf Curl Virus, Two-Spotted Spider Mite, Early Blight, and Septoria Leaf Spot. According to experimental results, testing accuracy is 91.2% on average. However, because it currently exists more than 2 million trainable parameters in the model, more study is required to improve both its accuracy and computing efficiency.

- **An Efficient Transfer Learning-based Approach for Apple Leaf Disease Classification [17]**

  With a 99.21% accuracy rate on a portion of the PlantVillage dataset, the research presents a transfer learning technique for identifying apple leaf diseases. With this method, the pre-trained EfficientNetV2S architecture is used as a feature extractor. Accurate predictions are then obtained by feeding the extracted features into a classification block. The study uses runtime data augmentation to correct class imbalance, which improves the performance of the model. Even with these promising outcomes, the study points out the drawbacks of utilizing a dataset created in a lab and

recommends additional testing in real-world settings to confirm the correctness and dependability of the model.

- **An optimized capsule neural networks for tomato leaf disease classification [18]**
  This study explores the use of an optimized Capsule Neural Network (CapsNet) to detect and categorize 10 different tomato leaf diseases according to the shape, color, and location of spots. Using preprocessing and data augmentation approaches to avoid overfitting, this method was able to retain minimal loss and effectively achieve 96.39% accuracy on the test set. The approach uses CNN to extract features first, and CapsNet to classify the data. The absence of information in the article about the datasets utilized for testing, validation, and training, however, raises questions about the model's generalizability to new datasets and could cause problems with transparency for readers.

- **Corn Leaf Diseases Recognition Based on Convolutional Neural Network [19]**
  With an accuracy score of 93% on the test set, a CNN-based model for identifying corn leaf diseases has been developed. The model's potential to generalize to a greater range of corn leaf diseases in real-world circumstances may be limited by the relatively small size of the testing dataset, which only included 200 images. The architecture of the CNN includes several convolution and pooling layers, followed by two fully connected layers. The small dataset size raises questions regarding the reliability of the results, even though the suggested method's average precision on the testing set is 93.24%.
- **Deep Learning- Based Segmentation and Classification of leaf images for detection of tomato plant disease [20]**
  In this study, deep learning algorithms are used to segment and classify tomato leaf diseases into groups that are healthy and diseased. More specifically, leaf images are split from their backgrounds using U-Net and a modified U-Net, and the segmented leaves are then classified using different versions of InceptionNet (InceptionNet1, InceptionNet2, and InceptionNet 3). An image segmentation test dataset yielded a 97.1% accuracy rate for the model. With an F1-score of 99.3% in multi-class classification across nine tomato diseases and a healthy class, this method produces outstanding results. However, the computational needs of these complex CNN

architectures may affect power consumption, memory efficiency, and processing speed.

- **A lightweight Convolutional Neural Network Model for Identification of Grape Leaf Diseases [21]**

  To particularly identify grape leaf diseases, the study presents GrapeNet, a lightweight CNN model. Convolution Block Attention Modules (CBAM), Residual Blocks, and Residual Feature Fusion Blocks (RFFBs) are all included in this model to help in feature extraction and increase model efficiency. In identifying different phases of grape leaf disease symptoms on the test set, GrapeNet obtained an accuracy of 86.25%. Nonetheless, this degree of accuracy suggests that additional improvement is required for practical implementations.

- **Plant disease detection using deep learning [22]**

  The project focuses on leveraging deep learning, specifically the MobileNet architecture and transfer learning, to identify plant diseases. The dataset it made use of included 70,295 images of both healthy and unhealthy plants in 38 different categories, such as tomato, apple, blueberry, cherry, corn, grape, orange, peach, pepper, potato, raspberry, soybean, squash, and strawberry. After 25 training epochs, the suggested approach yielded 95.05% accuracy with a loss of 14.72. Even with these findings, further work is required to increase accuracy and reduce the high loss rate, indicating that more adjustments may be necessary to maximize the model's effectiveness in real-world applications.

- **Enhancing plant disease detection: A novel CNN-based approach with tensor subspace learning and HOWSVD-MD [23]**

  This work presents a complex technique that employs pre-trained Convolutional Neural Networks (CNNs) and cutting-edge algorithms to categorize tomato leaf diseases. The proposed approach employs Higher-Order Whitened Singular Value Decomposition (HOWSVD) for enhanced discriminatory capability, followed by Multilinear Discriminant Analysis (MDA). Experiments conducted using PlantVillage and Taiwan datasets were used to evaluate the effectiveness of this technique, yielding accuracy scores of 98.36% and 89.39%, respectively. Further optimization is necessary because

the computational complexity of the method may provide difficulties for real-time applications.

- **Class-specific data augmentation for plant stress classification [24]**

  This study describes an automated genetics algorithm-based method for class-specific data augmentation that improves deep learning-based image classifiers for identifying and categorizing plant stress, particularly in soybean leaves. On the soybean leaf stress dataset, the approach achieves an overall accuracy of 98% and a mean-per-class accuracy of 97.61%, addressing the issue of imbalanced and confounding datasets. The method reduces the computational overhead of training classifiers from scratch for each augmentation policy by fine-tuning only the baseline model's linear layer. However, additional improvements in accuracy are required to optimize its usefulness in real-life circumstances.

  As discussed above, a quick overview of the related works are summarized below in a table format.

Table 1: Summary of Related Works

No	Research Title	Algorithm	Author	Critics	Efficiency
1	Unhealthy Region of Citrus Leaf Detection Using Image Processing Techniques	Image enhancement, K-means segmentation, GLCM feature extraction, and SVM classification.	Gavhale, Kiran & Gawande, Ujwalla & Hajari, Kamal. (2015).	**Computational complexity**: this restricts the method's efficiency and scalability in real world application. **Handcrafted feature extraction:** accuracy of the method depends on the selection of appropriate offset (distance and direction) and this requires prior knowledge which can introduce bias and generalization problem.	SVM with Radial Basis Function (RBF) kernel achieved a Genuine Acceptance Rate (GAR) of 96%.
2	Using Deep Learning for Image-Based Plant Disease Detection	The algorithm used involves: googlenet as CNN architecture, transfer learning to adapt pre-trained model to a specific task, colored images as input data and 80% - 20% training - testing split for evaluating the model	Sharada P. Mohanty, David P. Hughes, Marcel Salathé (2016).	When the model is tested on a set of images taken under conditions different from the images used for training, the model's accuracy is reduced substantially, to just above 31%. Which strongly indicates overfitting.	The trained model achieves an accuracy of 99.35% held-out on test set.
3	A Smartphone Application to Detection and Classification of Coffee Leaf Miner	Uses methods like ostu method and color space analysis for image segmentation. It utilizes GLCM and statistical color feature extraction for extracting texture and color	Giuliano L. Manso, Helder Knidel, Renato A. Krohling, Jose A.	**Computational complexity:** It can take a lot of time and computing power to calculate glcms and the associated statistical measures for large images or datasets. **Handcrafted feature extraction:** The primary focus of the study is on	Study's overall accuracy ranged from 91.458% to 99.095%

	and Coffee Leaf Rust	attributes respectively and these extracted features are given as an input to ANN classifier for further analysis and classification.	Ventura (2019).	extracting predefined features, such as homogeneity, contrast, energy, mean, deviation, etc. Which might not capture full detail of image data.	
4	A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks	Presents an enhanced Deep Residual–Inception Attention Convolutional Neural Network as a grape leaf disease detector (DR-IACNN).	Xiaoyue Xie, Yuan Ma, Bin Liu, Jinrong He, Shuqin Li, Hongyan Wang (2020).	study exhibits accuracy problem. (**lower accuracy**)	DR-IACNN, had a detection performance of 81.1% map (mean Average Precision). The detector also showed a speed of 15.01 frames per second (FPS).
5	Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks	A memory-efficient Simple two stage CNN architecture was created with only 0.8 million parameters, inspired by VGG16 to identify rice disease and pests.	Chowdhury Rafeed Rahman, Preetom Saha Arko, Mohammed Eunus Ali, Mohammad Ashik Iqbal Khan, Sajid Hasan Apon, Farzana Nowrin, and Abu Wasif (2020).	A major limitation of two stage training is that the entire dataset must be divided manually into symptom classes. In a large dataset, detecting all the major intra-class variations is a labor-intensive process. There is a great chance of missing some symptom variations.	Achieved an accuracy of 93.3% on the rice datasets with a significantly reduced model size.
6	ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network	Uses CNN based model to detect and classify tomato leaf disease, consists of 3 convolution layers and 3 max pooling layers followed by 2 fully connected layer.	Mohit Agarwal, Abhishek Singh, Siddhartha Arjaria, Amit Sinha, Suneet Gupta (2020).	Further research is required to improve the accuracy and computational complexity of the model (in which this built model consists of above 2 million trainable parameters).	Experiment result shows the average testing accuracy of the model is 91.2%

7	Plant disease detection using deep learning	Use transfer learning with one of the state-of-the-art pre-trained CNN architectures called mobilenet model.	Hariharan K, Kamai D, Nandhini S, Dr. S. Saravanan (2021).	The model is evaluated using one of the metrics called accuracy. However, this metric does not exactly show the complete picture of the model's performance. Therefore, additional metrics such as precision, recall and f1 - score are required for providing a more comprehensive evaluation of model's performance.	After 25 epochs the proposed model scores an accuracy of 95.05% and 14.72% loss.
8	Deep Learning-Based Segmentation and Classification of leaf images for detection of tomato plant disease	This system employs two state-of-art semantic segmentation models, U - Net and modified U - Net for segmenting leaf image from the background. Additionally, various versions of inceptionnet(inceptionnet1, inceptionnet2, and inceptionnet3) are utilized to classify the segmented tomato leaves to various classes.	Muhammad Shoaib, Tariq Hussain, Babar Shah, Ihsan Ullah, Sayyed Mudassar Shah, Farman Ali, Sang Hyun Park (2022).	The proposed study involves employment of two computationally complex CNN architectures for performing single tasks. This could potentially lead to delays in processing input and generating output due to increased computational requirements, which may impact time efficiency, memory usage and processing power demand.	The proposed study performs various experiments by using different numbers of classes, achieving F1 - score of 99.3% for multi-classification with 9 tomato diseased classes and healthy class.
9	A lightweight Convolutional Neural Network Model for Identification of Grape Leaf Diseases	A lightweight CNN named grapenet architecture was developed for the identification of grape leaf disease. The two more advantageous building blocks of this architecture are Residual Feature Fusion Block (RFFB) and Convolutional Block Attention Module (CBAM).	Jianwu Lin, Xiaoyulong Chen, Renyong Pan, Tengbao Cao, Jitong Cai, Yang Chen, Xishun Peng, Tomislav Cernava, Xin Zhang (2022).	study exhibits accuracy problem. (**lower accuracy**)	The model was able to detect different stages of symptoms associated with grape leaf disease with an accuracy score of 86.25% on the test set.

10	Deep Learning for Plant Identification and Disease Classification from Leaf Images	Use a Generalized Stacking Multi-output cnns (gsmo-cnns) approach	Jianping Yao, Son N. Tran, Saurabh Garg, Samantha Sawyer (2023).	More research is needed to determine the approach's flexibility and real-world applicability in a variety of agricultural circumstances and imaging conditions.	Exhibits competitive accuracy of 99.31% on plant village dataset, predicting plant species and disease presence at the same time.
11	An Efficient Transfer Learning-based Approach for Apple Leaf Disease Classification	Uses pre-trained CNN architecture called efficientnetv2s as a feature extractor and passes the extracted feature to a classifier block for effective prediction.	Md. Hamjajul Ashmafee, Tasnim Ahmed, Sabbir Ahmed, Md. Bakhtiar Hasan, Mst Nura Jahan, A.B.M. Ashikur Rahman (2023).	The research highlights the limitations of the lab-generated dataset and recommends additional validation in real-world situations to ensure the correctness and consistency of the model.	The proposed approach achieved a 99.21% accuracy rate on a portion of the plantvillage dataset.
12	Corn Leaf Diseases Recognition Based on Convolutional Neural Network	CNN based approach has been used for corn leaf recognition. The CNN architecture consists of some convolution layers and pooling layers followed by two fully connected layers.	Mutia Fadhilla, Des Suryani, Ause Labellapansa, Hendra Gunawan (2023)	The dataset size used for testing was 200 images, which is quite small and limits the potential for generalizing the results to a broad range of corn leaf diseases in real-world situations.	The average precision of the proposed method on the testing set is 93.24%.
13	Transfer learning with Densenet201 architecture model for potato leaf disease classification	Densenet201 architecture, utilizing transfer learning was used to identify potato leaf disease.	Rifqi Alfinnur Charisma, Faisal Dharma Adhinata (2024).	**Generalization problem:** there is a significant difference between the training and test accuracies, suggesting an overfitting issues.	The accuracy of the training set was 99.5%, the validation set was 95.2%, and the test set was 92.5%.

14	An optimized capsule neural networks for tomato leaf disease classification	Combines both CNN for initial feature extraction and capsnet for further classification to detect and classify tomato leaf disease.	Lobna M. Abouelmagd , Mahmoud Y. Shams, Hanaa Salem Marie and Aboul Ella Hassanien (2024)	Insufficient detail about datasets used for training, validating and testing. This could raise concern regarding the model's ability to generalize on unseen datasets and transparency issues for readers.	Using an improved Capsule Neural Network (capsnet) methodology, the study achieved an accuracy of 96.39% in classifying ten tomato leaf diseases.
15	Enhancing plant disease detection: A novel CNN-based approach with tensor subspace learning and HOWSVD-MD	**Pre-trained Convolutional Neural Networks (CNNs):** These are used to extract features from tomato leaf images. **Higher-Order Whitened Singular Value Decomposition (HOWSVD):** A tensor subspace learning technique that enhances the discriminatory capability of the model. **Multilinear Discriminant Analysis (MDA):** This follows HOWSVD and is used for further feature extraction and classification.	Abdelmalik Ouamane, Ammar Chouchane, Yassine Himeur, Abderrazak Debilou, Abbes Amira, Shadi Atalla, Wathiq Mansoor, Hussain Al Ahmad (2024).	Real-time applications may be restricted by high computing complexity. Advanced algorithms could require powerful hardware for deployment.	Achieved high accuracy of 98.36% on the PlantVillage dataset and 89.39% on the Taiwan dataset.
16	Class-specific data augmentation for plant stress classification	The study utilizes a genetic algorithm to automate the selection of class-specific data augmentation strategies for improving plant stress classification.	Nasla Saleem, Aditya Balu, Talukder Zaki Jubery, Arti Singh, Asheesh K. Singh, Soumik Sarkar, Baskar Ganapathysubramanian (2024).	The method, though effective, still requires **further improvement in accuracy** to enhance its applicability in real-life situations.	The study achieved a mean-per-class accuracy of 97.61% and an overall accuracy of 98% on the soybean leaf stress dataset.

2.3 Gap Analysis

Our review on existing researches reveals several challenges and are discussed below:

- **Computational complexity:** many of the approaches have increased computational complexity impacting time efficiency, memory usage and processing power making it difficult for practical use.
- **Handcrafted feature extraction:** accuracy of the method depends on the selection of the right features.
- **Overfitting:** one of the main issues with using deep learning technique is having a major difference between training and validation accuracies, which shows weaker generalization capability of the model.
- **Accuracy issues:** Some models are not accurate enough, making it challenging for practical application in the real world.
- **Evaluation metrics:** accuracy alone as a metric is insufficient for evaluation of model performance. Most of the approaches use overall accuracy as their primary metric to measure performance of their model. Which could be misleading as it ignores metrics like precision and recall which are very useful when errors have serious consequences.
- **Model transparency and dataset details:** transparency and replication are limited by inadequate dataset and training procedure statements.

This approach seeks to take advantage of these problems as chances to enhance existing methods. Compared to other methods, it addresses these problems to detect and classify plant diseases more accurately.

Chapter Three

3. Methodology

This study aims at building a more efficient and accurate model for detecting and classifying grape leaf disease. This section discusses the set of software and hardware tools used and steps followed to build our model.

3.1 Type of Research

This research is categorized as both quantitative and experimental:

- **Quantitative Research:** By classifying images, this study gathers numerical data that can be statistically analyzed to determine model performance measures including accuracy, precision, recall, F1 score, and loss. This quantitative method makes it easier to evaluate how well the model classifies diseases of the grape leaf.
- **Experimental Research**: The design of the experiment makes it easier to manipulate factors (such model architectures and hyperparameters) and assess how they affect classification performance. This makes it possible to validate and evaluate several approaches in a methodical way to get the best results.

3.2 Material

3.2.1 Software tools

Python is one of the most popular programming languages in research. The structure of the language and its object-oriented approach help programmers to write logical and clear code for small and large projects. Python libraries (packages) effectively simplify many important processes such as analyzing and visualizing data, retrieving unstructured data from the web, image processing, building machine learning models, and textual information [25].

The implementation of the proposed system utilizes the following software tools:

- **Jupyter notebook:** utilized for building, training and evaluating our model.
- **Spyder:** utilized for developing the final graphical user interface (GUI).
- **TensorFlow and Keras:** utilized for building, training and evaluating machine learning models.
- **Matplotlib:** utilized for data visualization and plotting.

- **NumPy:** utilized for numerical operation and array manipulation.
- **Tkinter:** utilized for developing the graphical user interface (GUI), providing a user-friendly interaction with the system.

Together, these and other useful tools provide a comprehensive and effective workflow for the development, training, and evaluation of the deep learning model for plant disease detection and classification.

3.2.2 Hardware tools

In order to run our experiments very smoothly we worked on a laptop with the following specifications:

- Processor: intel core i7-8565U
- RAM: 8 GB
- Operating System: window 11

3.3 Methodology

3.3.1 Dataset Collection

To develop accurate image classifiers for the purposes of plant disease diagnosis, we needed a large, verified dataset of images of diseased and healthy plants. Until very recently, such a dataset did not exist, and even smaller datasets were not freely available [26]. To tackle this problem Kaggle project has started collecting images of healthy and diseased crop plants which are available freely [27]. This study employs datasets collected from Kaggle website as the images are high resolution and professional.

Initially, the collected dataset consists of 4062 images with four classes, each having the following values:

- Grape Black rot has 1180 images.
- Grape Esca (Black Measles) has 1383 images.
- Grape healthy has 423 images.
- Grape Leaf blight (Isariopsis Leaf Spot) has 1076 images.

The following figure shows an image of grape leaf corresponding to each class:

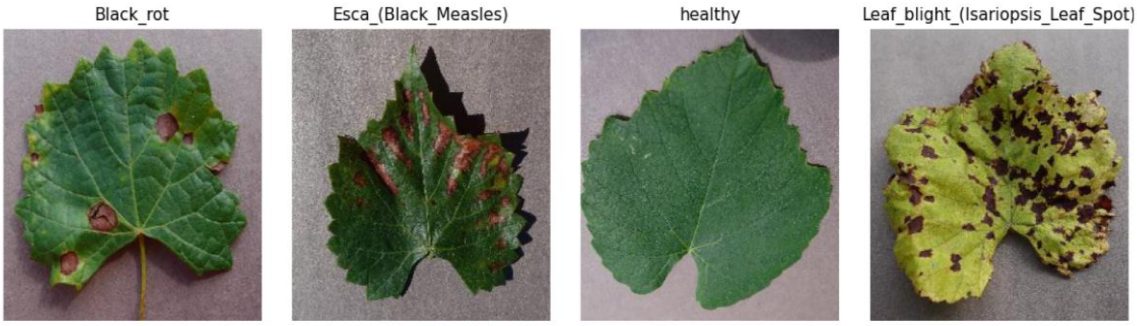


Figure 1:Sample Images from Each Class.

3.3.2 Dataset Preprocessing

Instead of feeding the dataset as it is to the neural network it will be very helpful to preprocess the dataset to create a cleaner, manageable, and abundant amount of data. This section involves several essential steps:

- **Dataset splitting:** the dataset was split into train, validation and test set using 70%-15%-15% split ration. This split ratio was chosen after performing various experiments on different split ratio and the use of this split ratio helps the model perform well compared to others.
- **Dataset balancing:** initially the number of images in each class was different exhibiting class imbalance. This can potentially bias the model towards a class with a larger number of images, affecting the overall performance and generalization. And to prevent this issue, oversampling technique was used to ensure that the number of images in each class was balanced. Beginning with the class that has the largest number of images as a reference, the method included augmenting the dataset. Subsequently, images were produced for more classes until each class's image count was almost identical or nearly equal. Table 2 shows the number of images for each class in each set before and after the datasets are being balanced. This oversampling technique was crucial in reducing overfitting and increases the generalization capability of the model.

Table 2 : Distribution of Datasets Before and After Oversampling for Train, Validation and Test Sets.

Classes	No images before oversampling			No of images after oversampling		
	Train set	Validation set	Test Set	Train set	Validation set	Test Set
Grape Black rot	826	177	177	967	926	941
Grape Esca (Black Measles)	968	207	208	968	942	949
Grape healthy	296	63	64	952	933	931
Grape Leaf blight (Isariopsis Leaf Spot)	753	161	162	963	945	934

- **Further data augmentation:** Following the dataset's balancing, additional augmentation was carried out with a focus on the training set to meet the model's requirement for wide-ranging data. This process increases the number of images from 3850 to 19,356 creating multiple augmented versions of each image in the training set using different image transformation techniques to create a richer and more varied dataset that could help in enhancing the model's capacity for generalization and reliability. Among the augmentation techniques used were image shearing, zooming, and horizontal flipping. In general, to build and evaluate the model a total of 19,356 images for training, 3746 images for validating, and 3755 images for testing were used.

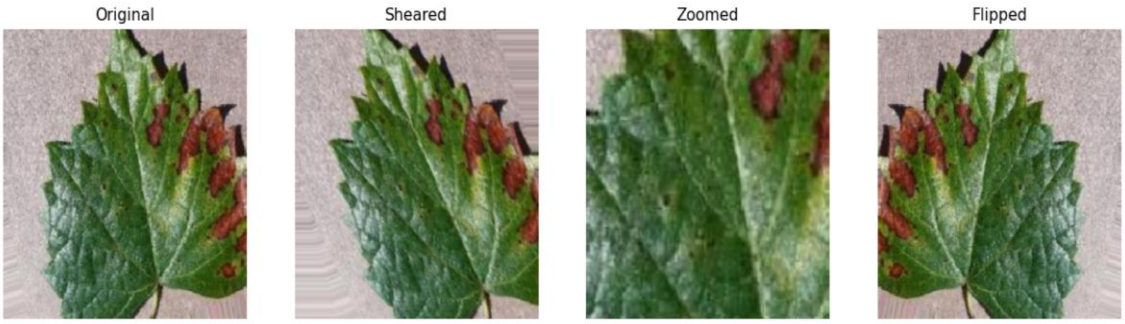


Figure 2 : Augmented Images of Black Measles Classes.

- **Preparing images for the pre-trained models:** Images were being normalized and resized to dimensions 299 x 299 x 3 to meet the requirements of the pre-trained models. By ensuring uniformity and compliance with the model architecture, this preprocessing phase improves the accuracy and efficiency of training.

3.3.3 Building a Model

3.3.3.1 Overview of CNN Architectures

This study, which was motivated by state-of-the-art deep learning techniques, uses CNNs to detect and classify grape leaf disease. CNNs work well for image classification tasks because they are capable of automatically extracting hierarchical features from images without requiring human feature engineering, which makes them an excellent choice for image classification tasks [28].

To select the best performing and computationally efficient model, three of the lately developed, computationally efficient and state-of-the-art pre-trained CNN architectures were evaluated.

- **Inception ResNet V2:** this architecture is hybrid Inception version 4 with residual block with significantly improved recognition performance [29]. This architecture introduces a very deep neural network with relatively reduced number of parameters reducing computation cost while providing a state-of-the-art performance outperforming resnet151

and inception v3 with top 1 error rate of 16.4% and top 5 error rate of 3.1% [29].

- **EfficientNet – B4:** belongs to the family of EfficientNets, which uses a compound scaling mechanism to systematically scale the network's width, depth, and resolution [30]. The accuracy and efficiency of EfficientNets are significantly higher than those of earlier ConvNets.
- **Xception:** stands for "Extreme Inception" and is based on inception architecture but it replaces the standard convolution with depth wise separable convolution. As a result, a more accurate and efficient model with fewer parameters is produced [31].

To address the research question "Which existing models are more suitable and how to enhance from existing one?" The performance of these three pre-trained models is carefully evaluated. The models are evaluated according to two criteria, F1 - score of the validation set, and computational efficiency.

3.3.3.2 Hyperparameter Tuning and Selection

Unlike model parameters learned during training, hyperparameters are predetermined and play a vital role in a model's performance [32]. To maximize model performance and generalization, key hyperparameters such as learning rate, regularization techniques, number of epochs and others must be adjusted.

The final set of hyperparameters used for the model following careful fine-tuning is summarized in the table below.

Table 3: List of Hyperparameters Selected for This Study

Hyperparameter	Description	Value
Learning rate	Controls how much the model changes in response to the estimated error each time the model weights are updated.	0.0001
Batch size	The number of training examples utilized in one iteration.	32
Number of epochs	The number of complete passes through the training dataset.	20
Dropout rate	The fraction of input units drops to prevent overfitting.	0.5

Optimizer	The algorithm used to update the weights of the model during training.	Adam
Loss	measure of the difference between the predicted and actual results, which directs the optimization of the model.	Categorical cross entropy
Early stopping	A technique to stop training when validation accuracy stops improving, to prevent overfitting.	Patience: 6
Model checkpoint	Saves the best model based on validation accuracy to avoid losing the best-performing model.	Save Best Only: True

These hyperparameters were chosen after much experimentation with the goal of improving the model's capacity to accurately detect and classify grape leaf diseases while maintaining a balance between computational efficiency and performance.

3.3.3.3 Learning Mechanisms

Deep learning has a hunger on a training dataset to achieve a well-behaved performance model. And in this case, we have a limited number of images or datasets. To properly address this problem one of the suggested methods is the employment of transfer learning [33]. We used three pre-trained models for this study: Xception, EfficientNet B4, and Inception ResNet V2. The ImageNet dataset, which has over a million images divided into 1000 classes, was used to train these models in the beginning. These pre-trained models allow us to apply the learned features to our grape leaf disease dataset.

To fit our classification objective, which includes detecting and classifying three common grape leaf diseases Black Rot, Black Measles, and Leaf Blight as well as one class for healthy grapes, we changed the last output layers of each pre-trained model.

3.3.3.4 Loss function

The loss function is also another important building block of the model that directly affects its performance. Loss function needs to fulfill the following

properties to achieve a very good performance. It must be a convex function avoiding a local minimum problem and it must be differentiable as the backward propagation does differentiation to update the model parameters [33]. According to [34] for multi - class classification categorical cross entropy is commonly used as a lost function and as a result this study have used this type of loss function.

3.3.3.5 Optimizer

This study used Adam optimizer. Adam optimizer stands for Adaptive Moment Estimation which is built by combining two of the most popular optimization methods SGDS with momentum and RMSprob optimizing techniques [35].

3.3.4 Model improvement

Modify the Inception ResNet V2 architecture of the pre-trained model to improve performance. This is done by making syntactic changes to the Inception ResNet V2 architecture and performing experiments with different hyperparameters such as learning rate, batch size, and number of layers.

In Inception ResNet V2, two of the inception ResNet blocks A and B are succeeded by a reduction modules A and B respectively responsible for reducing spatial size of input image while increasing number of feature maps enabling the network to learn faster [29]. This study introduces a new reduction module called reduction module C after the final Inception ResNet V2 block C, which did improve the performance of the model.

3.3.4.1 Reduction module C

Taking principles explained by [36] on how to build efficient grid size reduction module and previously built reduction modules A and B [29] as an input, this study introduces a new reduction module called reduction module C to the Inception ResNet V2 architecture.

This reduction module reduces the grid size of the input generated by the final Inception ResNet V2 block C from 8 X 8 to 3 X 3 while increasing the number of feature maps, enabling the network to focus only on features that are relevant for the model to make an accurate detection and classification.

This improved the accuracy level of the architecture. And the module has less computational cost compared to the existing reduction block A and reduction block B.

This study introduces four different versions of reduction module C namely, reduction module C using total of 256 filters,128 filters, 64 filters and factorized filter using total of 64 filters. The details of the design of these different versions of the reduction module C are clearly described under the result section of this study.

To provide a complete understanding of the Methodology, the corresponding diagram displays the overall system architecture.

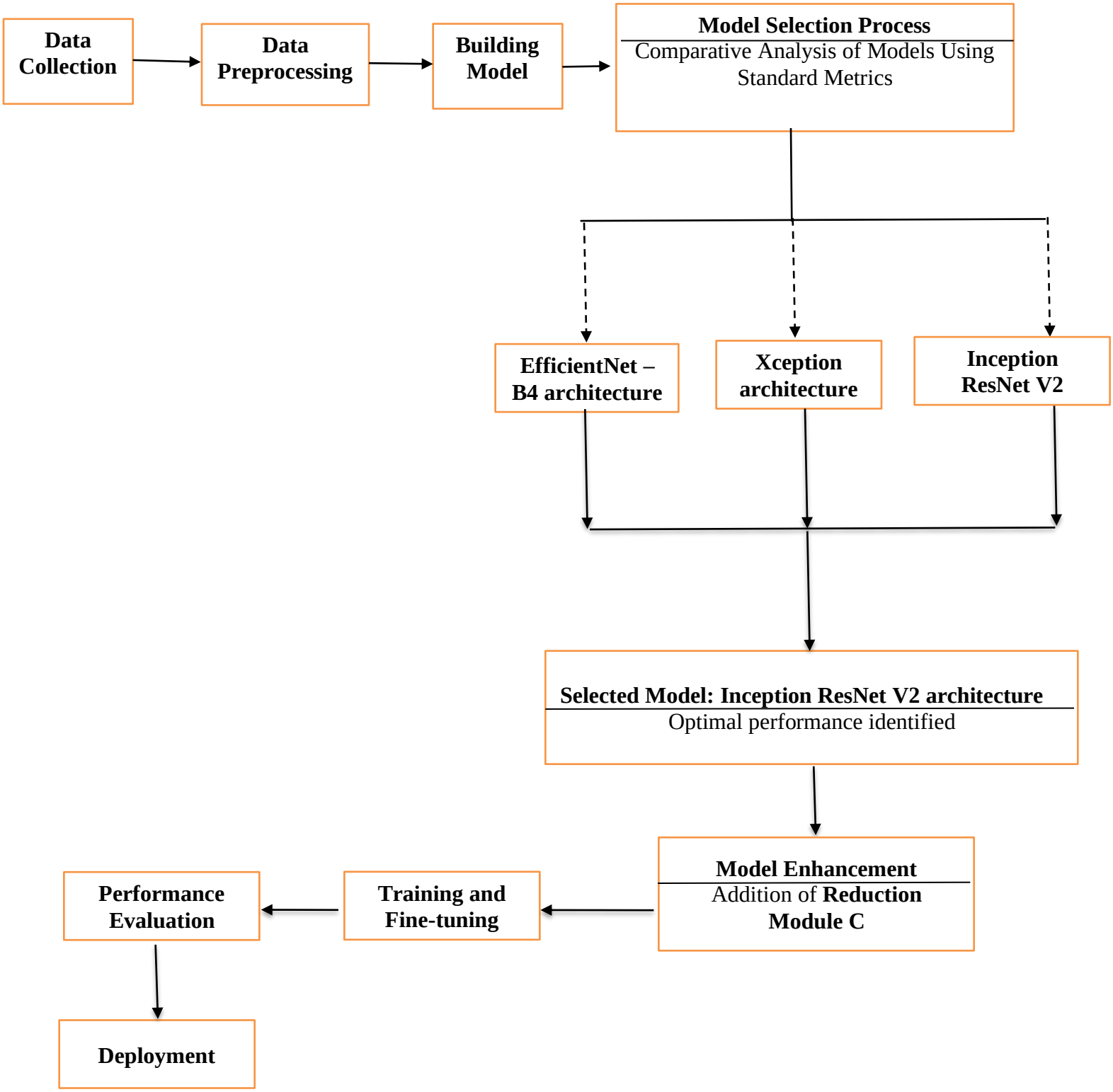


Figure 3: Proposed System Architecture

Chapter Four

4. Results and Discussion

This section looks at the results of our experiments that were conducted to build an accurate model for the detection and classification of grape leaf disease. To accomplish accurate classification with the least number of computational resources, our study carefully evaluated a variety of model architectures and methodologies.

4.1 Dataset Description

We experimented with a large dataset of over 26,857 images that were divided into four classes based on the types of grape leaf diseases and carefully separated into training, validation, and test sets.

4.2 Model Architecture

In this study, we tested three pre-trained CNN architectures, Inception ResNet V2, EfficientNet B4, and Xception. These architectures were chosen because they offer high image classification performance and computational resource efficiency. In order to determine which model performed the best, we compared these models using various hyperparameter settings. The F1 score on the validation set was the primary criteria, followed by computational complexity.

During the implementation of the model, the top classification layer was removed from all model architectures. And subsequently, a dense layer with different number of neurons (1024, 512, and 256) using RELU activation was added, followed by final classification layer with four neurons using SoftMax activation.

The dense layer employs various types of regularization in different experiments to address overfitting, along with batch normalization to normalize activations and improve training stability and efficiency.

We also introduce a new Inception ResNet V2 version, each integrated with a custom Reduction Module C, to significantly improve classification performance. This modular

augmentation was carefully designed to minimize computing resources and improve the model's capacity to accurately detect and classify grape leaf diseases.

4.3 Experimental Results

4.3.1 Confusion Matrices

In this study, the confusion matrix is used as an important tool to assess the performance of the classification models. It provides a detailed breakdown of the model's predictions by displaying the number of **true positives (TP)**, **true negatives (TN)**, **false positives (FP)**, and **false negatives (FN)** [34].

- **True Positives (TP)**: These are cases where the model correctly predicts the disease when it is present.
- **True Negatives (TN)**: The model correctly identifies healthy leaves (no disease present).
- **False Positives (FP)**: These represent instances where the model incorrectly predicts a disease when the leaf is actually healthy.
- **False Negatives (FN)**: This occurs when the model fails to detect the disease even though it is present.

The confusion matrix is crucial to this study because, in addition to overall accuracy, it shows how well the model predicts each grape leaf disease. To provide a better understanding of the model's performance, it also provides extensive information into important measures including precision, recall, and F1 score. We provide the confusion matrices for the training, and validation sets below so that we can further assess the model's performance.

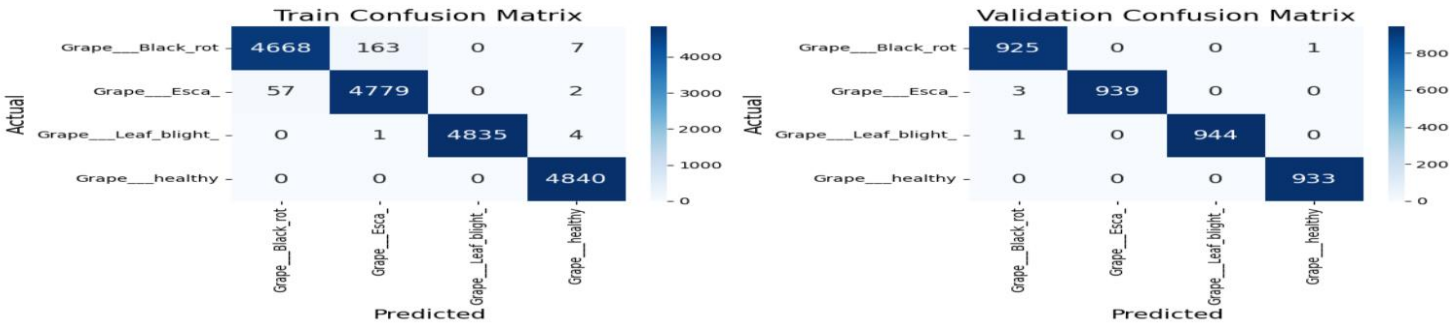


Figure 4 : Confusion Matrix of Inception ResNet V2

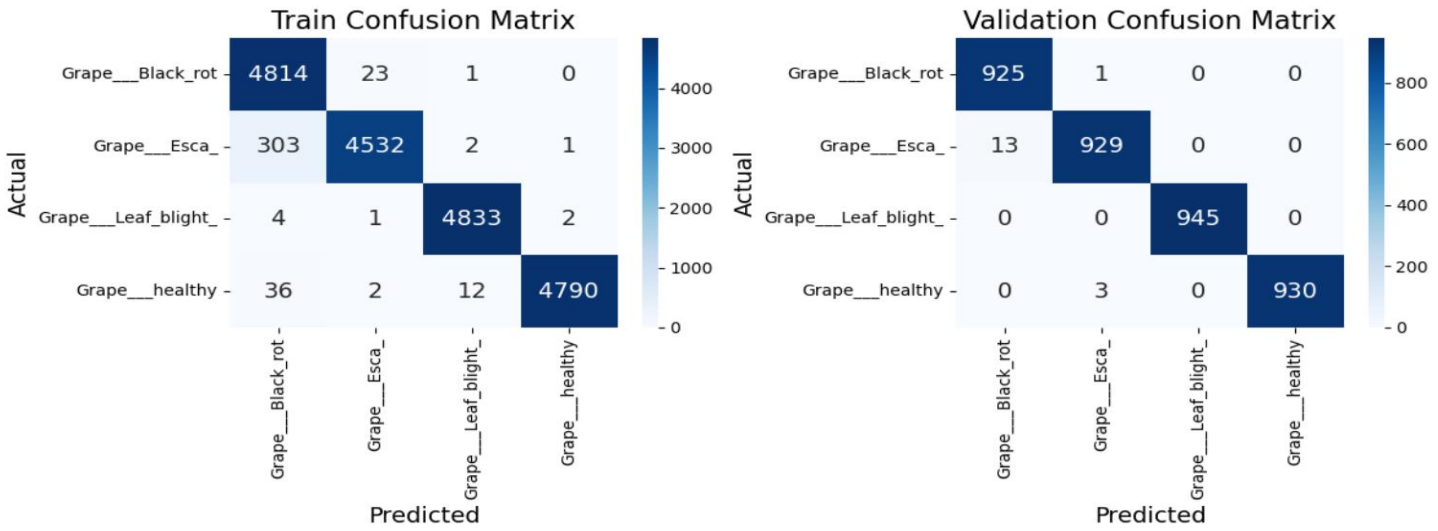


Figure 5 : Confusion Matrix of Efficient Net – B4

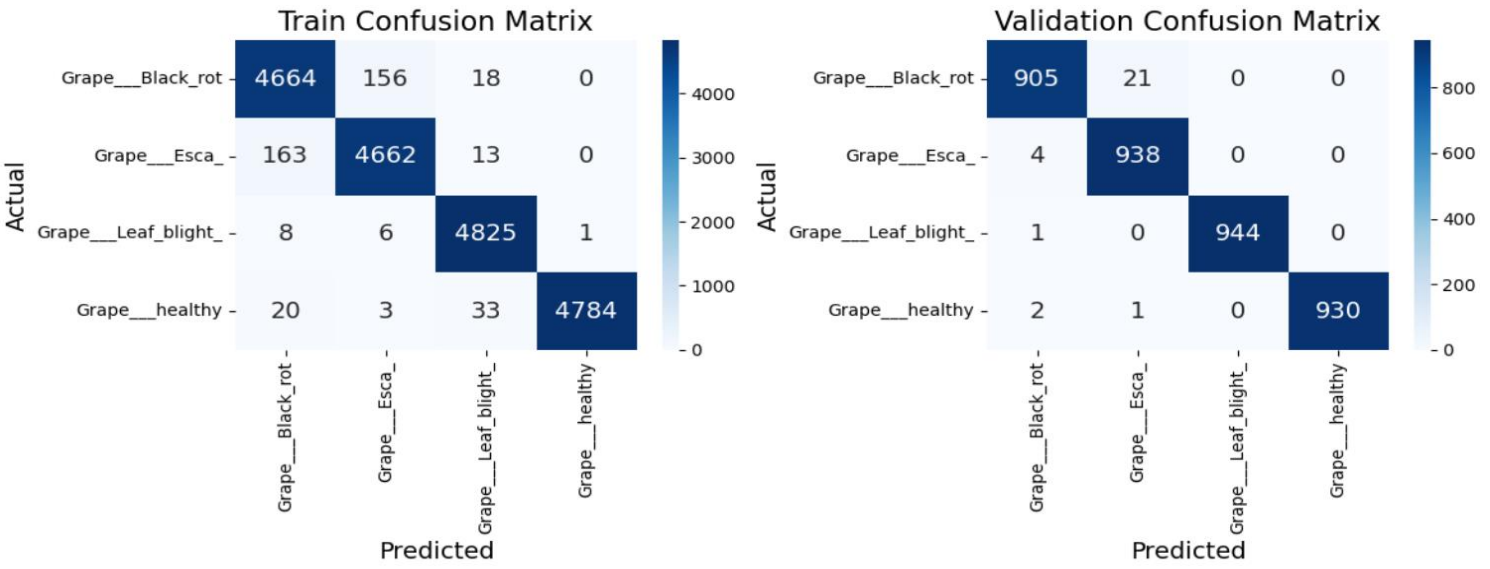


Figure 6 : Confusion Matrix of Xception

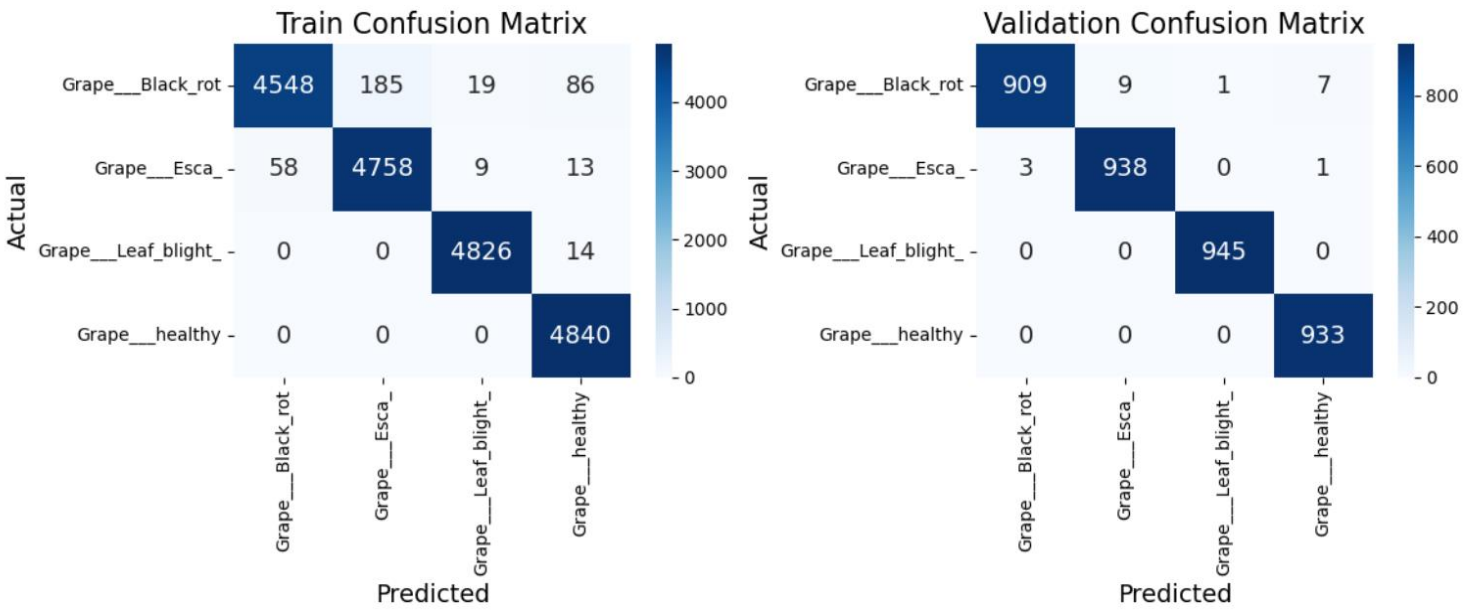


Figure 7 : Confusion Matrix of Inception ResNet V2 with Reduction Module 64 Filters

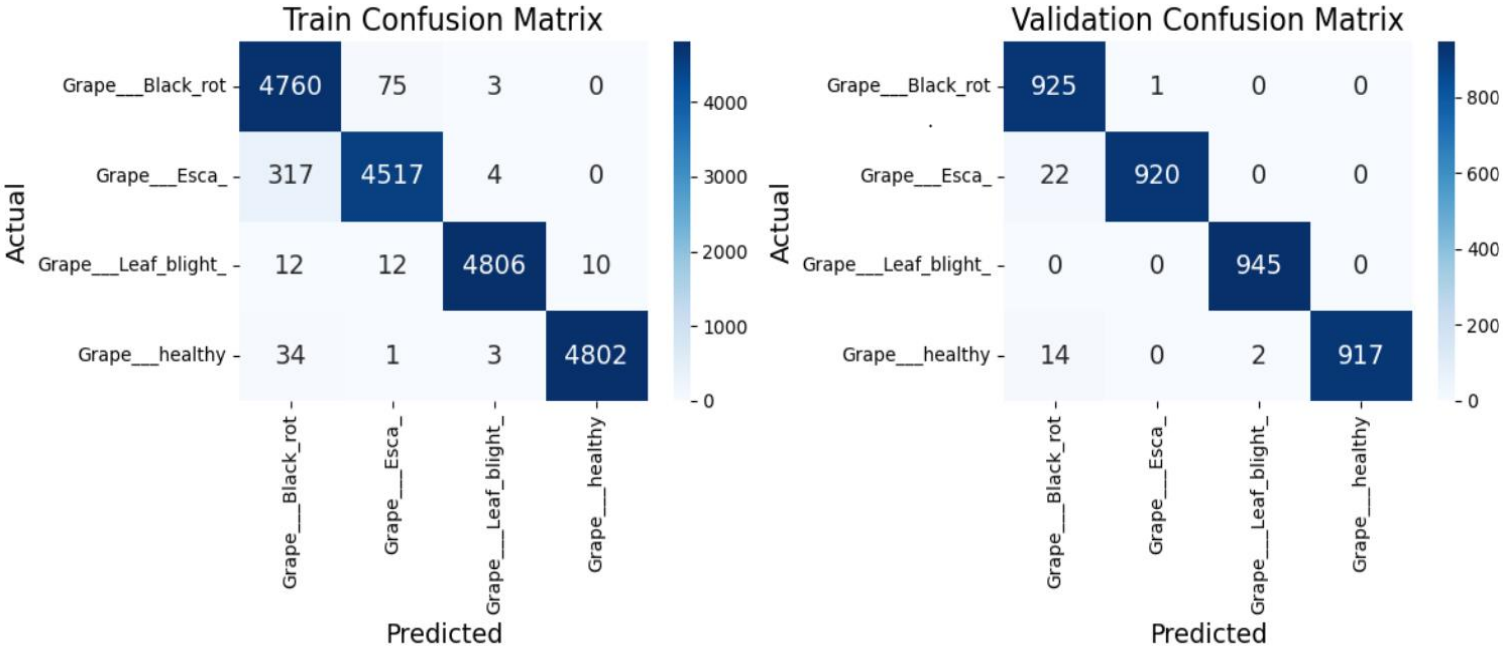


Figure 8 : Confusion Matrix of Inception ResNet V2 with Reduction Module 128 filters

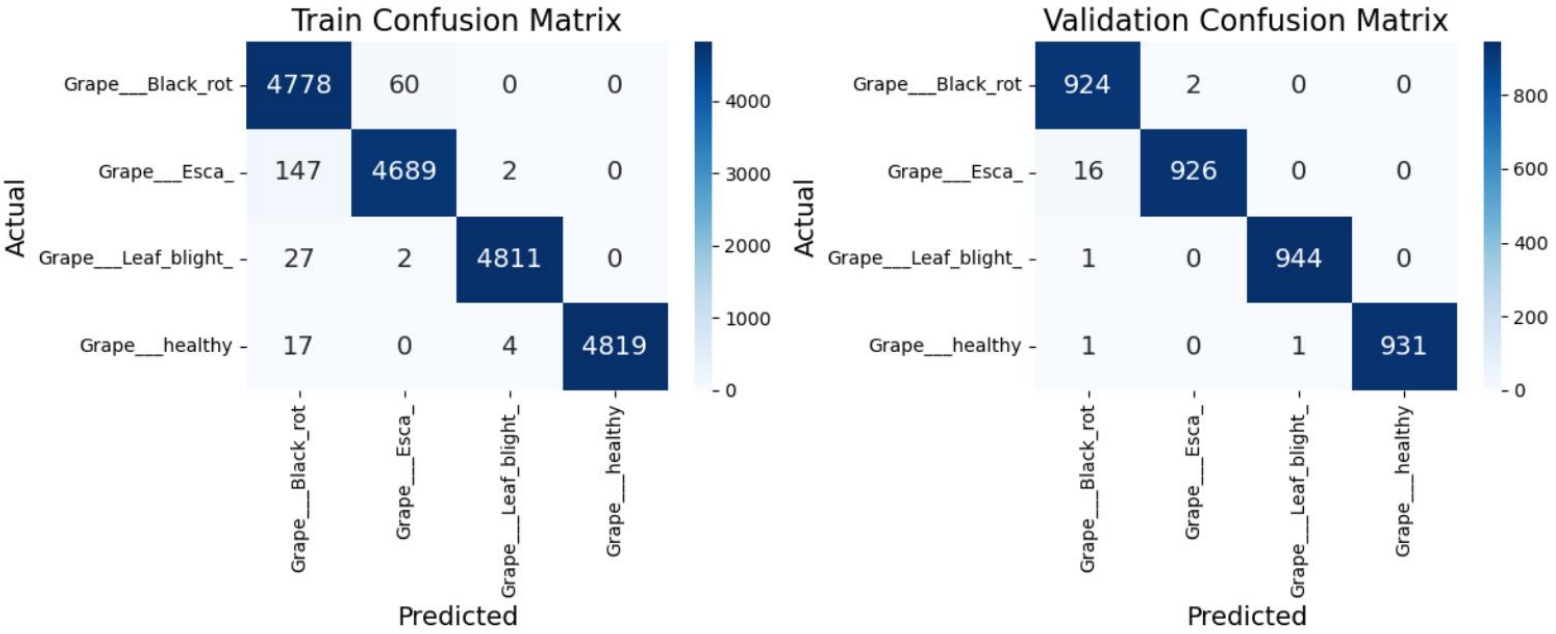


Figure 9 : Confusion Matrix of Inception ResNet V2 with Reduction Module 256 filters

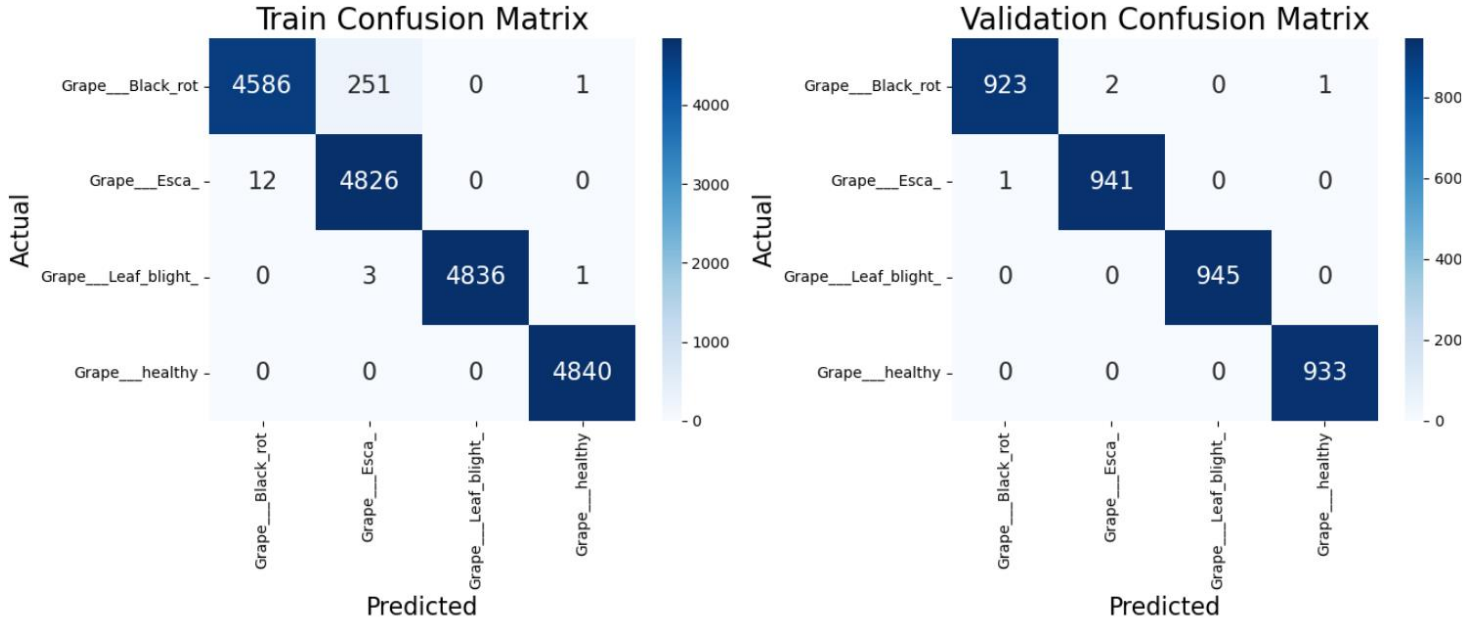


Figure 10 : Confusion Matrix of the final enhanced Inception ResNet V2 with Factorized Reduction Module 64 filters

4.3.2 Performance Metrics and Model Complexity of Pretrained Models

Each pretrained CNN architecture was subjected to a series of six experiments involving hyperparameter tuning. The goal was to select the best-performing architecture for further enhancements.

In machine learning, hyperparameter tuning is essential for enhancing model generalizability and accuracy. Although each hyperparameter affects a model's efficiency and performance, it might take a lot of time and processing resources to tune them all. As a result, we chose the three most important hyperparameters to tune for our study, regularization technique, number of neurons in the dense layer located before the classification layer, and learning rate. We think they will have major effects on attaining higher performance with comparatively fewer trainable parameters.

Each hyperparameter in the study was modified in accordance with the outcomes of earlier experiments using an iterative hyperparameter tuning strategy. For example, 512 neurons became the baseline for additional tuning if they performed better than 1024. Similarly, when Dropout performed better than L2 regularization, it will be chosen as the fixed regularizer in future iterations.

A batch size of 32, 15 epochs, 1024 neurons in the dense layer, a learning rate of 0.001, a regularizer called Dropout with a rate of 0.5, the Adam optimizer, categorical cross-entropy as the loss function, and early stopping with a patience value of 4 were all parts of the initial baseline hyperparameter setup used to train the model. In addition, the top-performing model was saved based on validation accuracy using a model checkpoint.

A detailed evaluation of each model's performance was conducted using five primary metrics **loss**, **accuracy**, **precision**, **recall**, and **F1 score [34]** were used. These metrics provide a comprehensive understanding of how well each model detects and classifies grape leaf diseases.

- **Loss:** is a metric that measures the difference between the model's predictions and the actual results during training.
- **Accuracy:** reflects the overall correctness of the model’s predictions.

- **Precision:** to make sure the model doesn't produce too many false positives; precision assesses the percentage of predicted disease instances that are actually true.
- **Recall** evaluates the model's ability to capture all actual disease cases, reducing the chances of missing any.
- **F1 score** balances both precision and recall, offering a single metric that accounts for both false positives and false negatives.

In addition to the performance metrics, the number of trainable parameters for each experiment was recorded in order to assess the models' computational complexity. This study enables a better understanding of the trade-off between model performance and resource requirements, which is crucial in deploying these models in real-world applications.

The table below summarizes the results, providing a clearer comparison of the model performances and computational complexity.

Table 4 : Comparing Performance and Complexity of Pretrained Models for Grape Leaf Disease Classification Under Multiple Hyperparameter Configurations

No	Model	Experiment ID	Tuned Hyperparameters			Dataset	Loss	Accuracy	Precision	Recall	F1 – Score	Number of Trainable Parameters
			Neorons	Regularizer	Learning rate							
1	**Inception ResNet V2**	Exp – 1	1024	Dropout /0.5/	0.001	Training Set	0.05	0.9817	0.9824	0.9824	0.9824	1.58 million
						Validation Set	0.011	0.9967	0.9968	0.9968	0.9968	
		Exp – 2	512	Dropout /0.5/	0.001	Training Set	0.052	0.9815	0.9880	0.9879	0.9879	0.790 million
						Validation Set	0.0075	0.9987	0.9987	0.9987	0.9987	
		Exp – 3	512	L2 /0.001/	0.001	Training Set	0.2034	0.9608	0.9611	0.9610	0.9610	
						Validation Set	0.1197	0.9936	0.9937	0.9936	0.9936	
		Exp – 4	512	Dropout /0.5/	0.01	Training Set	0.1015	0.9660	0.9660	0.9660	0.9660	
						Validation Set	0.0063	0.9984	0.9984	0.9984	0.9984	
		Exp – 5	512	Dropout /0.5/	0.0001	Training Set	0.1082	0.9616	0.9617	0.9616	0.9616	
						Validation Set	0.0158	0.9976	0.9977	0.9976	0.9976	
		Exp – 6	256	Dropout /0.5/	0.001	Training Set	0.1634	0.9382	0.9387	0.9386	0.9886	0.395 million
						Validation Set	0.0155	0.9960	0.9965	0.9964	0.9964	

No	Model	Experiment ID	Tuned Hyperparameters			Dataset	Loss	Accuracy	Precision	Recall	F1 – Score	Number of Trainable Parameters
			Nerons	Regularizer	Learning rate							
2	**Efficient NetB4**	Exp – 1	1024	Drop out /0.5/	0.001	Training Set	0.8550	0.9582	0.9815	0.9807	0.9807	1.84 million
						Validation Set	0.3722	0.9955	0.9924	0.9922	0.9922	
		Exp – 2	1024	L2 /0.001/	0.001	Training Set	0.2353	0.9673	0.9889	0.9889	0.9889	
						Validation Set	0.1440	0.9979	0.9979	0.9979	0.9979	
		Exp – 3	512	L2 /0.001/	0.001	Training Set	0.1516	0.9761	0.9760	0.9760	0.9760	0.920 million
						Validation Set	0.0906	0.9981	0.9981	0.9981	0.9981	
		Exp – 4	256	L2 /0.001/	0.001	Training Set	0.1024	0.9608	0.9828	0.9827	0.9827	0.460 million
						Validation Set	0.0633	0.9936	0.9981	0.9981	0.9981	
		Exp – 5	256	L2 /0.001/	0.01	Training Set	0.2239	0.9605	0.9601	0.9601	0.9601	
						Validation Set	0.1205	0.9968	0.9968	0.9968	0.9968	
		Exp – 6	256	L2 /0.001/	0.0001	Training Set	0.1338	0.9950	0.9950	0.9949	0.9949	
						Validation Set	0.1115	0.9968	0.9968	0.9967	0.9967	

No	Model	Experiment ID	Tuned Hyperparameters			Dataset	Loss	Accuracy	Precision	Recall	F1 – Score	Number of Trainable Parameters
			Neurons	Regularizer	Learning rate							
3	**Xception**	Exp – 1	1024	Dropout /0.5/	0.001	Training Set	0.1830	0.9706	0.9700	0.9693	0.9690	2.10 million
						Validation Set	0.1269	0.9944	0.9944	0.9944	0.9944	
		Exp – 2	1024	L2 /0.001/	0.001	Training Set	0.1140	0.9777	0.9783	0.9782	0.9782	
						Validation Set	0.0819	0.9922	0.9923	0.9922	0.9922	
		Exp – 3	512	Dropout /0.5/	0.001	Training Set	0.0501	0.9819	0.9820	0.9819	0.9819	1.05 million
						Validation Set	0.0120	0.9973	0.9974	0.9973	0.9973	
		Exp – 4	512	Dropout /0.5/	0.01	Training Set	0.1991	0.9328	0.9400	0.9389	0.9389	
						Validation Set	0.0143	0.9952	0.9952	0.9951	0.9951	
		Exp – 5	512	Dropout /0.5/	0.0001	Training Set	0.0511	0.9824	0.9824	0.9823	0.9823	
						Validation Set	0.0122	0.9971	0.9971	0.9971	0.9971	
		Exp – 6	256	Dropout /0.5/	0.001	Training Set	0.0632	0.9769	0.9769	0.9769	0.9769	0.525 million
						Validation Set	0.0135	0.9971	0.9971	0.9971	0.9971	

4.3.3 Performance Analysis and CNN Architecture Selection

Let's take a look at the performance of the models built using pretrained Inception ResNet V2 CNN architecture. Out of the six models the model from Experiment ID 2 (number of neurons = 512, dropout regularization with 0.5 dropout ratio and learning rate = 0.001) outperformed all other models achieving 0.9987 F1 score on validation set with 0.79 million trainable parameters.

Next out of the six models built using pretrained EfficientNet B4 CNN architecture the model from Experiment ID 4 (number of neurons = 256, L2 regularization, and learning rate = 0.001) outperformed all other models achieving 0.9981 F1 score on the validation set with 0.46 million trainable parameters.

Finally, out of the six models built using pretrained Xception CNN architecture the model from Experiment ID 3 (number of neurons = 512, dropout regularization with 0.5 dropout ratio and learning rate = 0.001) outperformed all other models achieving 0.9973 F1 score on validation set with 1.05 million trainable parameters. Thus, based on the finding model built using pretrained Inception ResNet V2 CNN architecture outperformed all other pretrained models. And it was selected as a baseline model for further enhancement.

4.3.4 Enhancing Inception ResNet V2 architecture

Following the performance comparison and model selection, the Inception ResNet V2 architecture was identified as the best choice for further improvement because of its balanced performance across multiple metrics and improved F1 score on the validation set. Then to further improve this pretrained model a lightweight Reduction Module C was inserted into the pretrained Inception ResNet V2 CNN architecture. This improves the feature extraction capability of the baseline model by enabling the model to capture more complex features while ignoring irrelevant information.

4.3.4.1 Reduction Module C and Its Variants

To determine the most optimized design of Reduction Module C, four versions of Reduction Module C were designed and tested, which are shown below. The first three versions share the same design, varying only on the number of filters employed in the convolution layer. However, the fourth version differs both in design and on the number of filters employed in the convolution layer, this version is called the factorized version of Reduction Module C.

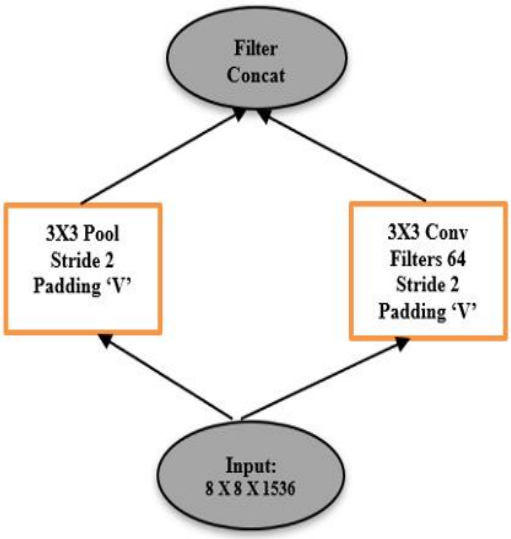


Figure 11(A): Reduction Module C with 64 Filters

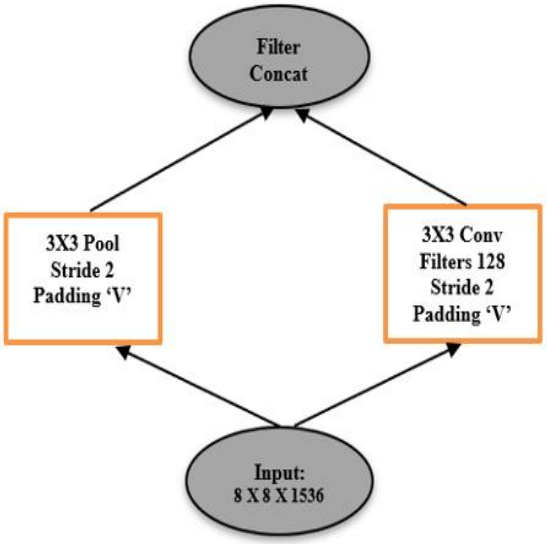


Figure 11(B): Reduction Module C with 128 Filters

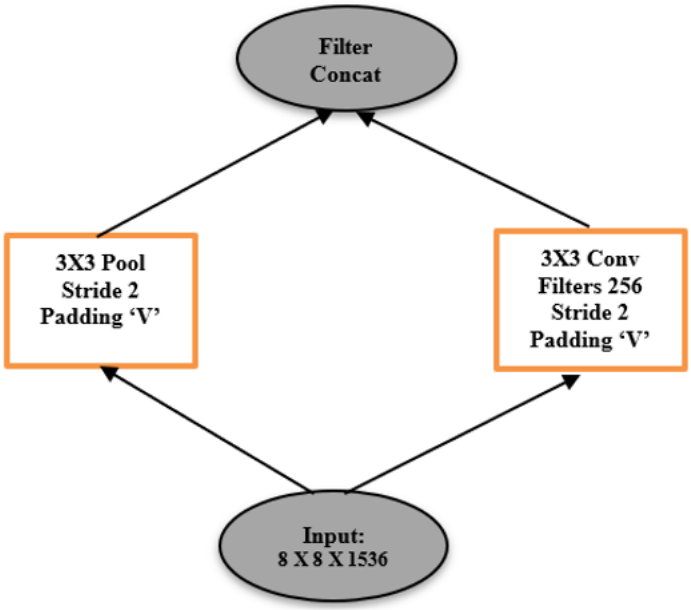


Figure 11(C): Reduction Module C with 64 Filters

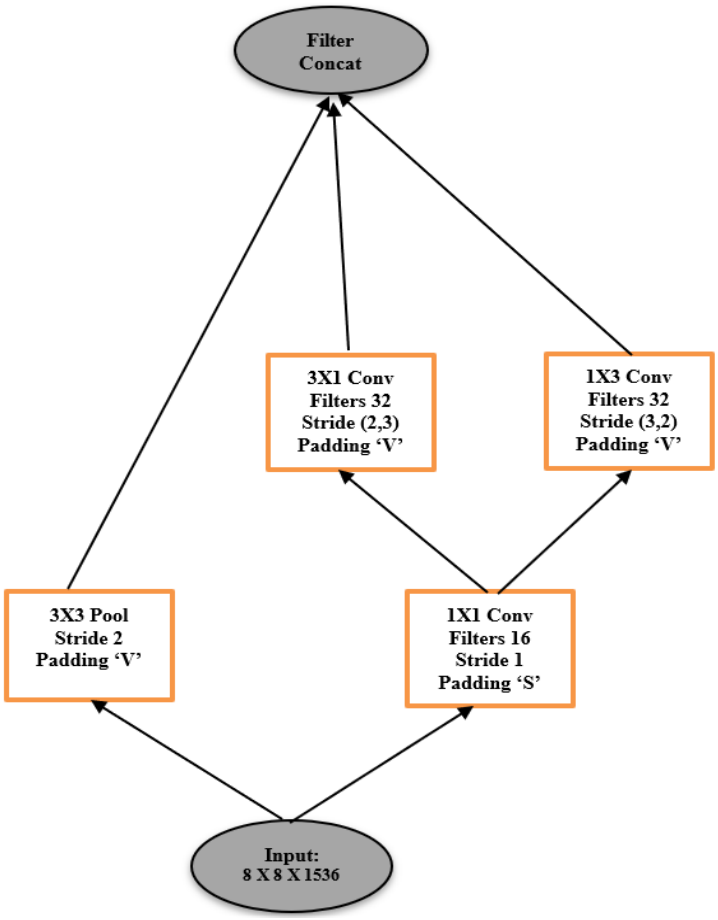


Figure 11(D): Reduction Module C Factorized Version

This module is a factorized version, replacing a single 3 * 3 convolution with 64 filters with two smaller convolutions. This improved version uses smaller trainable parameters while achieving an exceptional performance compared to the other three varieties of reduction module c. The key to this success is the use of 1 * 1 convolution with 16 filters and the breakdown of 3 * 3 convolution layers into 1 * 3 and 3 * 1 convolution layers, both with 32 filters.

Figure 11: Design of Reduction Module C with Various Configurations

4.3.4.2 Performance Metrics and Model Complexity Comparison of Enhanced Inception ResNet V2 Models

To identify the most optimized version of Reduction Module C, we built four models using each modified pretrained Inception ResNet V2 CNN architectures (pretrained Inception ResNet V2 with Reduction Module C). The experiments were conducted by using baseline hyperparameter configuration (batch size = 32, number of neurons = 1024, number of epochs = 15, drop out regularization with 0.5 dropout ratio, early stopping with patience value = 4, model checkpoint based on validation accuracy, optimizer = Adam and loss = categorical cross entropy).

The performance and computational complexity of each model is summarized in the following table.

Table 5 : Performance Metrics and Model Complexity Comparison of Inception ResNet V2 Models with Different Reduction Module C Designs

Number	Inception ResNet V2 With Reduction Module C	Dataset	Loss	Accuracy	Precision	Recall	F1 - Score	Number of Trainable Parameters
1	Reduction Module C with 64 filters using parallel 3 * 3 conv and pooling layer	Training Set	0.1398	0.9787	0.9804	0.9802	0.9800	2.53 million
		Validation Set	0.0990	0.9963	0.9944	0.9944	0.9944	
2	Reduction Module C with 128 filters using parallel 3 * 3 conv and pooling layer	Training Set	0.1737	0.9749	0.9765	0.9757	0.9757	3.48 million
		Validation Set	0.1659	0.9834	0.9899	0.9896	0.9896	
3	Reduction Module C with 256 filters using parallel 3 * 3 conv and pooling layer	Training Set	0.0956	0.9872	0.9868	0.9866	0.9866	5.38 million
		Validation Set	0.0830	0.9968	0.9945	0.9944	0.9944	
4	Reduction Module C where 3 * 3 Conv layer is factorized into 1 *3 and 3 *1 Conv layers each with 32 filters.	Training Set	0.1111	0.9867	0.9856	0.9855	0.9855	1.67 million
		Validation Set	0.0914	0.9955	0.9971	0.9971	**0.9971**	

4.3.4.3 Performance Analysis and Optimized Reduction Module C Selection

From the table we can observe that the factorized version of Reduction Module C outperformed the other three versions of Reduction Module C by achieving 0.9971 F1 score on the validation set with 1.67 million trainable parameters. Therefore, the factorized version of Reduction Module C will

be used as the basis for developing the final high-performing enhanced model due to its superior performance.

4.3.4.4 Enhancing Inception ResNet V2 with Optimized Reduction Module C

To further improve the performance of the modified pretrained Inception ResNet V2 Model (model with the optimized Reduction Module C), several models were built by configuring several key hyperparameters like learning rate, regularization and number of neurons in the dense layer, just like we did for the pretrained CNN architectures. Consequently, total of 10 experiments were conducted.

The following table summarizes the results from the 10 experiments:

Table 6: Performance Metrics and Computational Complexity of Models Using Inception ResNet V2 Architecture Incorporating Optimized Reduction Module C

Model	Experiment ID	Tuned Hyperparameters			Dataset	Loss	Accuracy	Precision	Recall	F1 – Score	Number of Trainable Parameters
		Neurons	Regularize	Learning rate							
Inception ResNet V2 with the optimized Reduction Module C	Exp – 1	1024	Dropout /0.5/	0.001	Training Set	0.1111	0.9867	0.9856	0.9855	0.9855	1.67 million
					Validation Set	0.0914	0.9955	0.9971	0.9971	0.9971	
	Exp – 2	512	Dropout /0.5/	0.001	Training Set	0.0646	0.9755	0.9778	0.9778	0.9778	0.835 million
					Validation Set	0.0147	0.9976	0.9976	0.9976	0.9976	
	Exp – 3	512	L2 /0.001/	0.001	Training Set	0.0582	0.9869	0.9870	0.9869	0.9869	
					Validation Set	0.0379	0.9939	0.9940	0.9939	0.9939	
	Exp – 4	512	Dropout /0.3/	0.001	Training Set	0.0426	0.9847	0.9840	0.9839	0.9839	
					Validation Set	0.0111	0.9965	0.9958	0.9957	0.9957	
	Exp – 5	512	Dropout /0.1/	0.001	Training Set	0.0846	0.9826	0.9822	0.9822	0.9822	
					Validation Set	0.0486	0.9968	0.9964	0.9964	0.9964	
	Exp – 6	512	Dropout /0.5/	0.0001	Training Set	0.0373	0.9850	0.9902	0.9902	0.9902	
					Validation Set	0.0060	0.9982	0.9982	0.9982	0.9982	
	Exp – 7	512	Dropout /0.5/	0.01	Training Set	0.0434	0.9869	0.9908	0.9908	0.9908	
					Validation Set	0.0061	0.9981	0.9981	0.9981	0.9981	
	Exp – 8	256	Dropout /0.5/	0.0001	Training Set	0.0396	0.9855	0.9910	0.9909	0.9909	0.417 million
					Validation Set	0.0045	0.9984	0.9984	0.9984	0.9984	
	Exp – 9	128	Dropout /0.5/	0.0001	Training Set	0.0387	0.9870	0.9922	0.9922	0.9922	0.208 million
					Validation Set	0.0056	0.9987	0.9987	0.9987	0.9987	

	Exp – 10	64	Drop out /0.5/	0.0001	Training Set	0.0543	0.9800	0.9855	0.9855	0.9855	0.104 million
					Validation Set	0.0143	0.9965	0.9969	0.9968	0.9968	

Out of all the experiments discussed above, **Experiment ID 9** had the best performance, with the fewest trainable parameters, 0.21 million parameters and an F1 score of 0.9987 on the validation set.

Despite having the same F1 score as the baseline model (0.9987), the final model is more computationally efficient since it has a substantially smaller number of trainable parameters (0.21 million compared to 0.79 million). Since our objective is to build a model that performs better than baseline model while being computationally efficient, we kept improving the model. To further improve its performance, we increased early stopping patience value to 6 and increased the number of epochs to 20, preventing overfitting and allowing more training time. And accordingly, these modifications resulted in a final F1 score of 0.9989, surpassing the baseline model performance.

The final comparison of the baseline model with the final enhanced model based on F1 score on validation set and number of trainable parameters is shown below in the table.

Table 7: performance and computational complexity comparison between baseline model and final enhanced Inception ResNet V2 model

Model	**F1 score (on validation)**	**Trainable Parameters**
Baseline Model from Experiment ID 2	0.9987	0.79 million
Final Enhanced Inception ResNet V2	0.9989	0.21 million

Thus, our enhanced model demonstrated a superior performance both in accuracy and efficiency, making it suitable for resource constrained devices and for practical use in the agricultural area.

4.3.4.5 Performance Analysis: Training Insights and Test Set Evaluation

By examining the training and validation processes as well as the test set performance, this section assesses the final enhanced Inception ResNet V2 model's performance. The objective is to make sure the model is computationally efficient, maintains high accuracy, and generalizes well to unseen dataset.

Training and Validation Insights

The final enhanced model's accuracy and loss during training and validation are shown in the following graph. With both training and validation accuracy continuously increasing across epochs, it demonstrates that the model converges well.

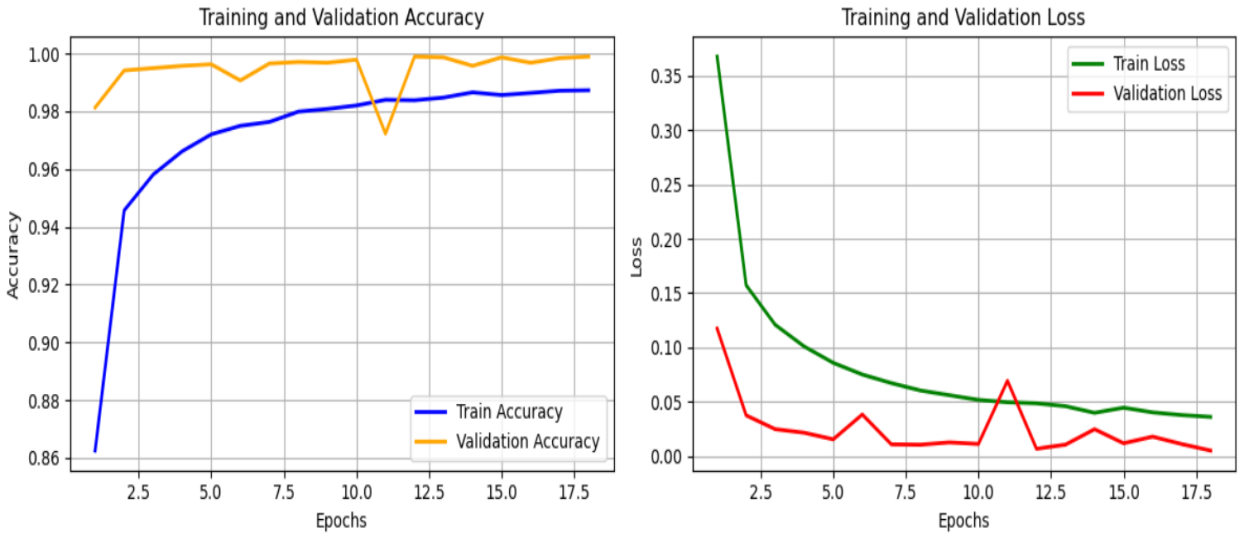


Figure 12:Training and Validation Accuracy and Loss Curves for the Enhanced Inception ResNet V2 Model

Here are the detailed explanations analyzed for the graphs displayed above:

a. **Training and Validation Accuracy**:

- The **training accuracy** (blue line) : here the training accuracy of the model is increasing gradually from epoch to epoch showing the model's ability to capture the most useful features from the training data.
- The **validation accuracy** (yellow line) : here the validation accuracy of the model falls in the range between 98 and 99. Showing a very few differences with the train accuracy of the model indicating the model's excellent generalization capability.
- The small fluctuations in the validation accuracy indicate slight differences in performance but aren't indicators to any serious instability or overfitting.

b. Training and Validation Loss:

- The **training loss** (green line) demonstrates a clear and smooth decline, indicating that the model's classification on the training set improve consistently over epochs.
- The **validation loss** (red line) declines generally but shows minor oscillations, which could be a sign of the dataset's random fluctuations or sensitivity to challenging validation samples.

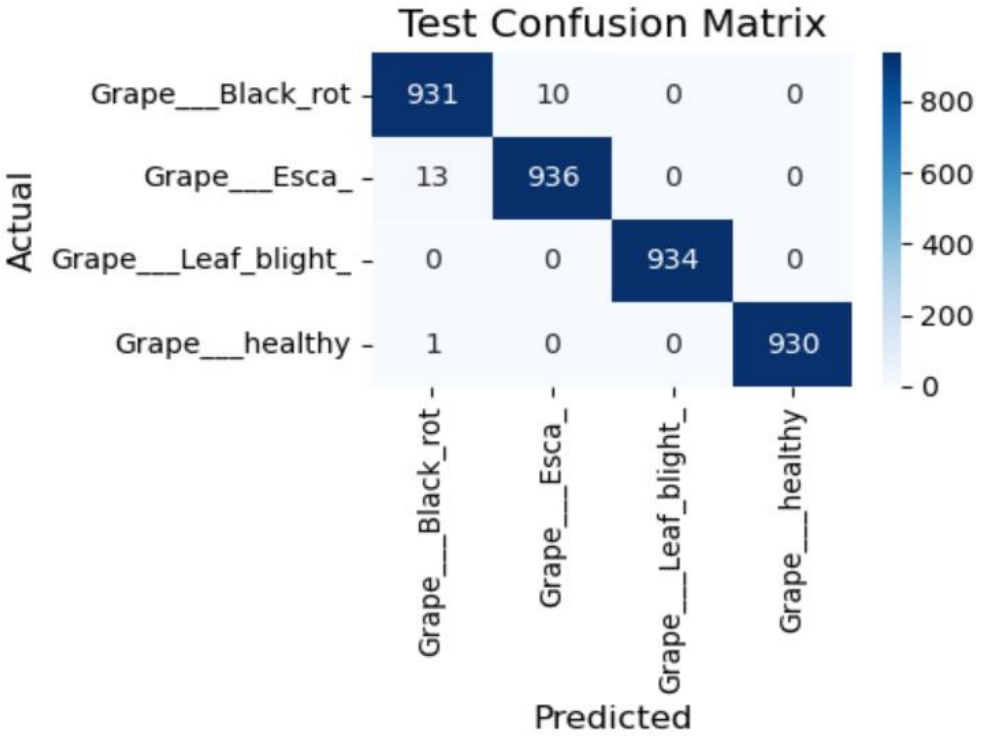
From the graph we can analyze that the enhanced model is generally well-optimized, with low loss, high accuracy, and no significant signs of overfitting. The minor oscillation that occurred in the validation metrics do not indicate any serious issue.

Test Set Evaluation

To observe how well our model is doing with unseen data, we evaluated the confusion matrix of our model on the test set. And accordingly, we observed that our model correctly classified 3731 and incorrectly classified 24 sets of grape leaf images. This shows our model's strong generalization capability to unseen data, making it the right model for practical use in the real world.

The confusion matrix of our model on the test set is shown below:

Figure 13: Final Enhanced Model's Confusion Matrix on Test Set.



Following an analysis of the confusion matrix, the following table displays the final enhanced model's overall performance on multiple metrics for the test set:

Table 8: Performance Metrics of The Final Enhanced Model on Test Set

Loss	Accuracy	Precision	Recall	F1 Score
0.0194	0.9936	0.9939	0.9937	0.9937

As we can see from the table our model achieved F1 score of 99.37 on the test set which simulates real world condition showing the model generalize well to unseen data. This makes it suitable for practical use in the agricultural environment.

4.4 Discussion

4.4.1 Model Selection

From the above evaluations, the final enhanced model, incorporating Reduction Module C with factorized convolution layers and the fine-tuned adjustments performs better than other variants regarding computation efficiency and performance. Not only this but also when compared with the other pre-trained models such as Inception ResNet v2, Efficient Net - B4, and Xception this enhanced architecture stands out as a top performer, achieving an outstanding performance. Thus, this final enhanced model is chosen as the most appropriate grape leaf disease detection and classification model.

4.4.2 Performance Summary

This part offers an overview of the model complexity insights and a performance evaluation of all built models using the F1 score on the validation set.

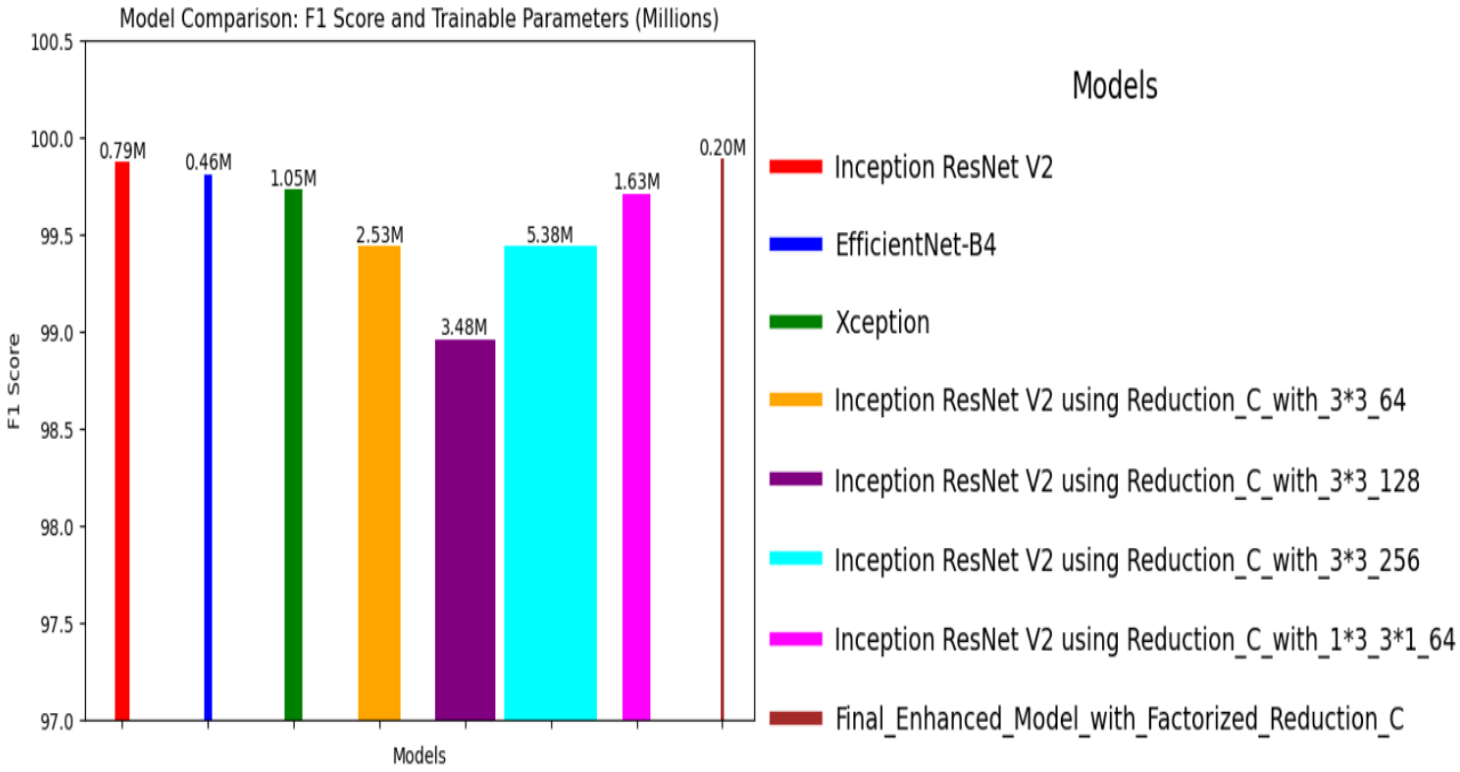


Figure 14: Performance Evaluation of the Best Performing Models Using the F1 Score on the Validation Set

The bar graph shows the top 8 best performing out of the 30 models that are built throughout the study. Where the height of the bar represents F1 score on validation set and the thickness represents the number of trainable parameters in millions.

From the bar graph we can observe that our enhanced model (Final_Enhanced_Model_with_Factorized_Reduction_C) outperformed all the models achieving 99.89% F1 score on validation set with 0.21 million trainable parameters.

This higher F1 score indicates higher precision and higher recall which is useful for this study. Since, the goal of building this system is to detect diseases early, preventing the spread of disease (higher recall) and avoid false alarm, preventing the use of unnecessary pesticide (higher precision).

Now that we have a better idea of the model's performance in several areas, let's take a closer look at the F1 score for every class in the validation set.

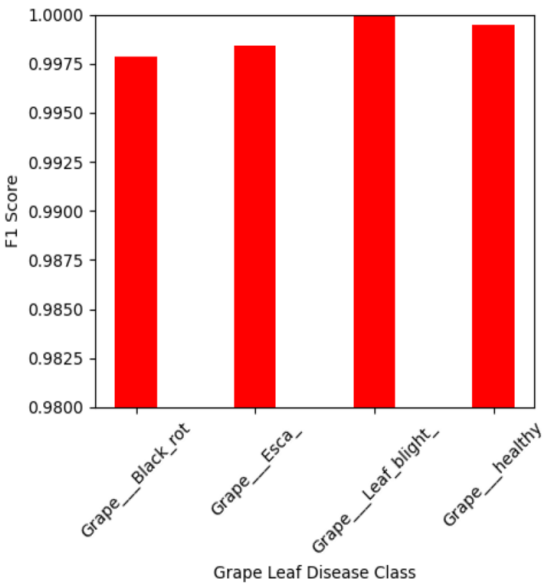
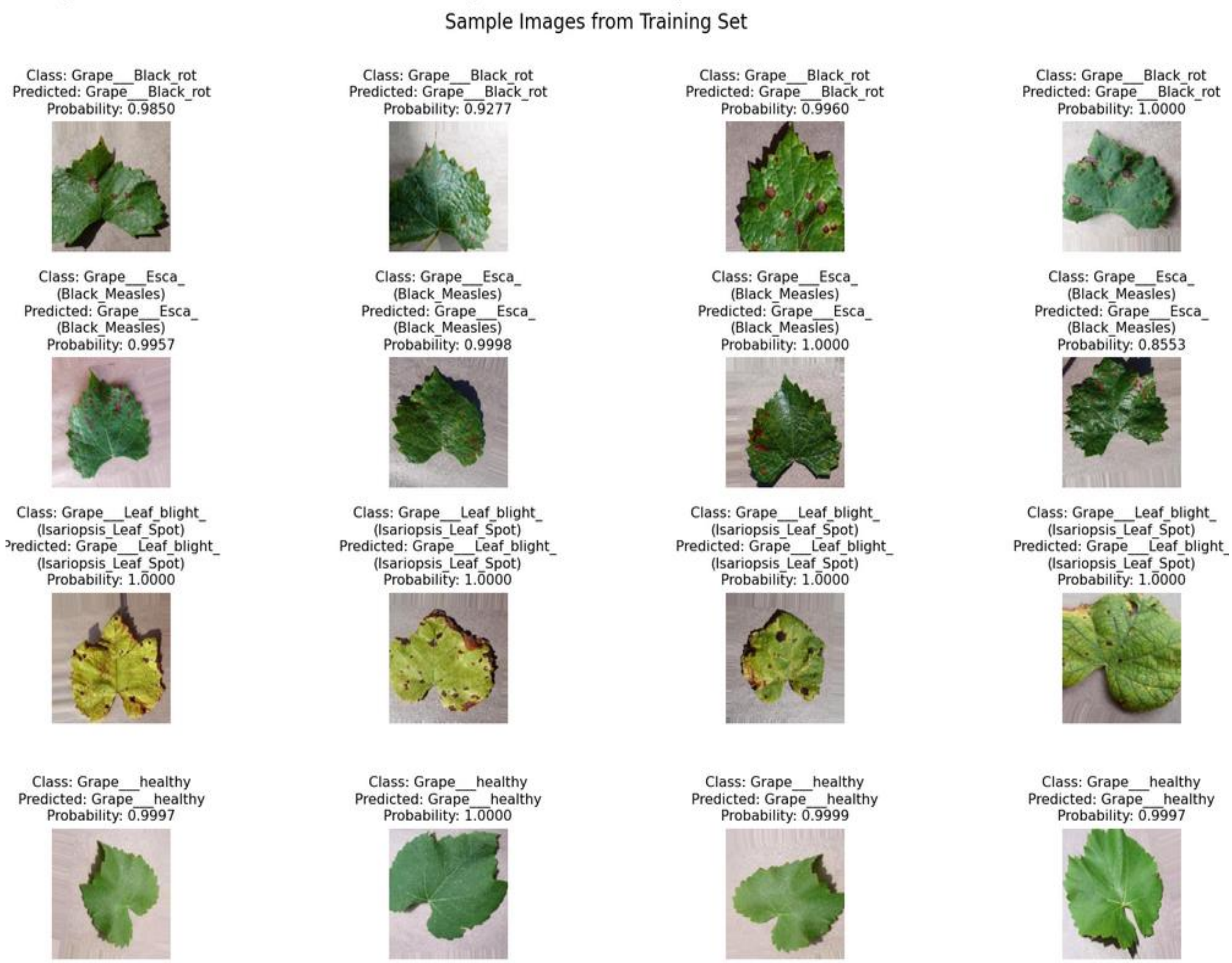


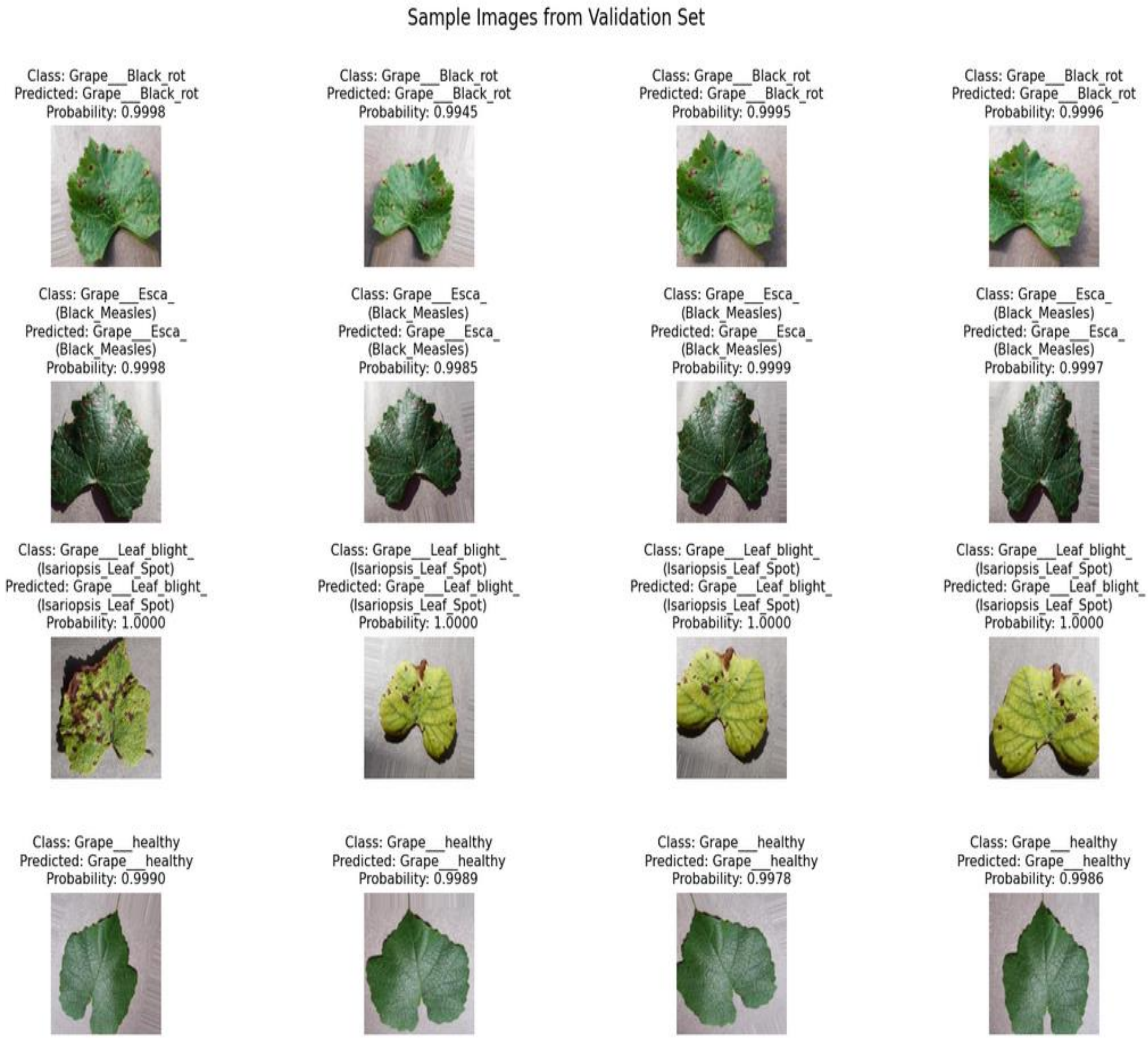
Figure 15:F1 score of each class achieved by final enhanced model on validation set.

The bar graph shows F1 core of the enhanced model on the validation set for each class. The model achieved a satisfactory score in all classes. Out of the four classes the model does exceptionally well on the grape leaf blight disease.

4.4.3 Sample Image Detection and Classification

Let us see some sample image detections and classifications from each set to better demonstrate the model's performance. We will observe the related probabilities assigned by the model in addition to the detected class labels. This will offer a more advanced viewpoint on the degree of confidence the model has in detecting and classifying different grape leaf disease categories.




Sample Images from Validation Set
Class: Grape___Black_rot
Predicted: Grape___Black_rot
Probability: 0.9998
Class: Grape___Black_rot
Predicted: Grape___Black_rot
Probability: 0.9945
Class: Grape___Black_rot
Predicted: Grape___Black_rot
Probability: 0.9995
Class: Grape___Black_rot
Predicted: Grape___Black_rot
Probability: 0.9996
Class: Grape___Esca_(Black_Measles)
Predicted: Grape___Esca_(Black_Measles)
Probability: 0.9998
Class: Grape___Esca_(Black_Measles)
Predicted: Grape___Esca_(Black_Measles)
Probability: 0.9985
Class: Grape___Esca_(Black_Measles)
Predicted: Grape___Esca_(Black_Measles)
Probability: 0.9999
Class: Grape___Esca_(Black_Measles)
Predicted: Grape___Esca_(Black_Measles)
Probability: 0.9997
Class: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Predicted: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Probability: 1.0000
Class: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Predicted: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Probability: 1.0000
Class: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Predicted: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Probability: 1.0000
Class: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Predicted: Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Probability: 1.0000
Class: Grape___healthy
Predicted: Grape___healthy
Probability: 0.9990
Class: Grape___healthy
Predicted: Grape___healthy
Probability: 0.9989
Class: Grape___healthy
Predicted: Grape___healthy
Probability: 0.9978
Class: Grape___healthy
Predicted: Grape___healthy
Probability: 0.9986

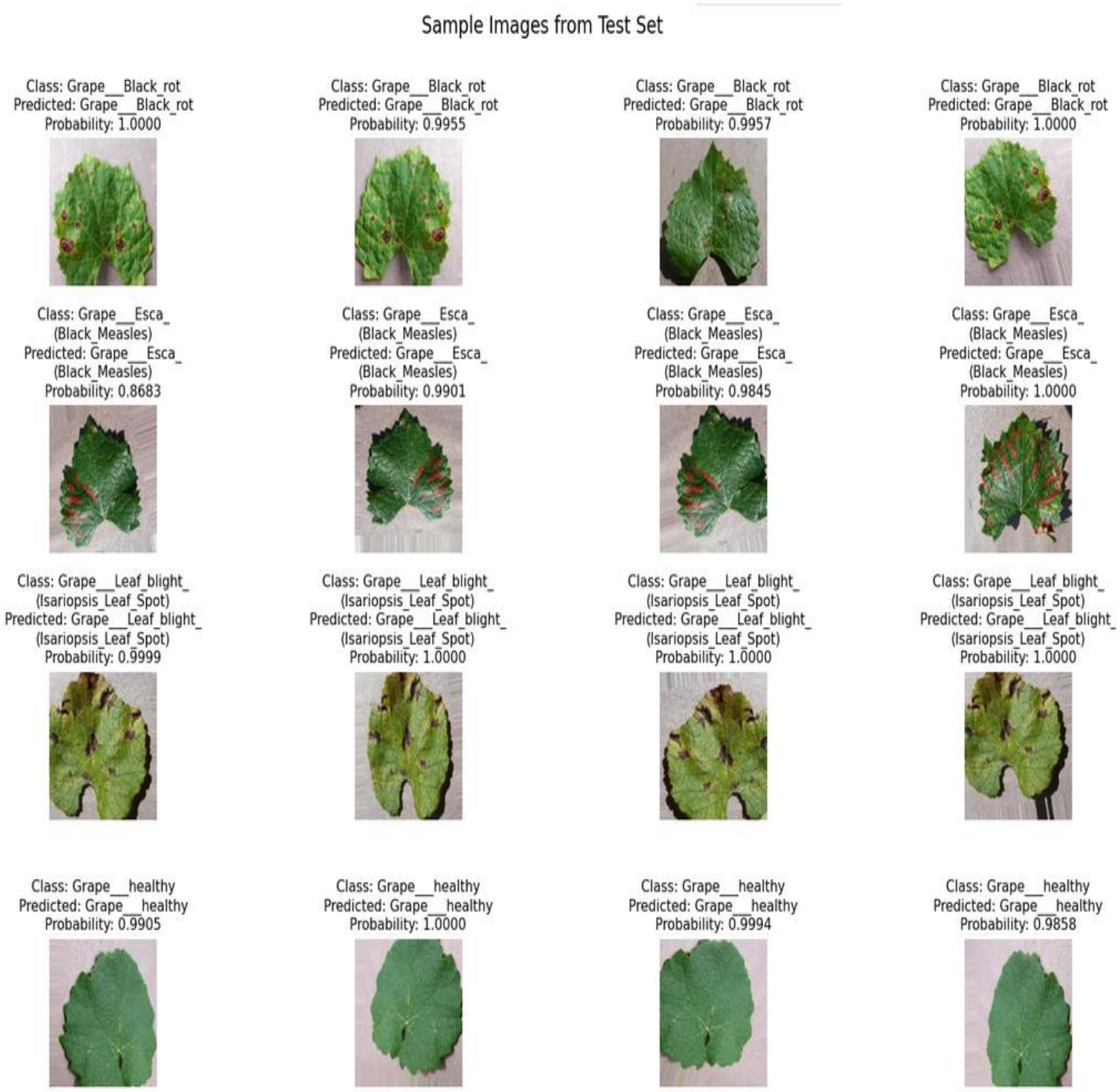


Figure 16: Detected and Classified Classes and Confidence Level for Grape Leaf Disease Images

4.4.4 Feature Visualization

4.4.4.1 Studying and comparing the final enhanced model's and the baseline model's visual features

This part compares the activation maps produced by two models for a particular grape leaf image affected by leaf blight disease. To determine the effect of improvement on feature representation, the baseline model and the final enhanced model are compared. This feature visualization aims to evaluate the impact of Reduction Module C's insertion on the model's feature representation.

This visualization of the activation map is done by starting to select a particular image of a grape leaf that has the leaf blight disease as a sample for comparison. To create activation maps, the image shown below will be fed into the baseline and final enhanced models as input.

Figure 17: Sample Grape Leaf Image Affected by Leaf Blight Disease

a. **Activation map from baseline model:** the activation map of the baseline model was generated from the final convolutional layer (Conv_7b_ac) using class activation mapping. This layer was selected because the output of this directly fed to the custom lightweight Reduction Module C after enhancing the baseline model. This will help us understand the feature extraction capability before and after adding the module.

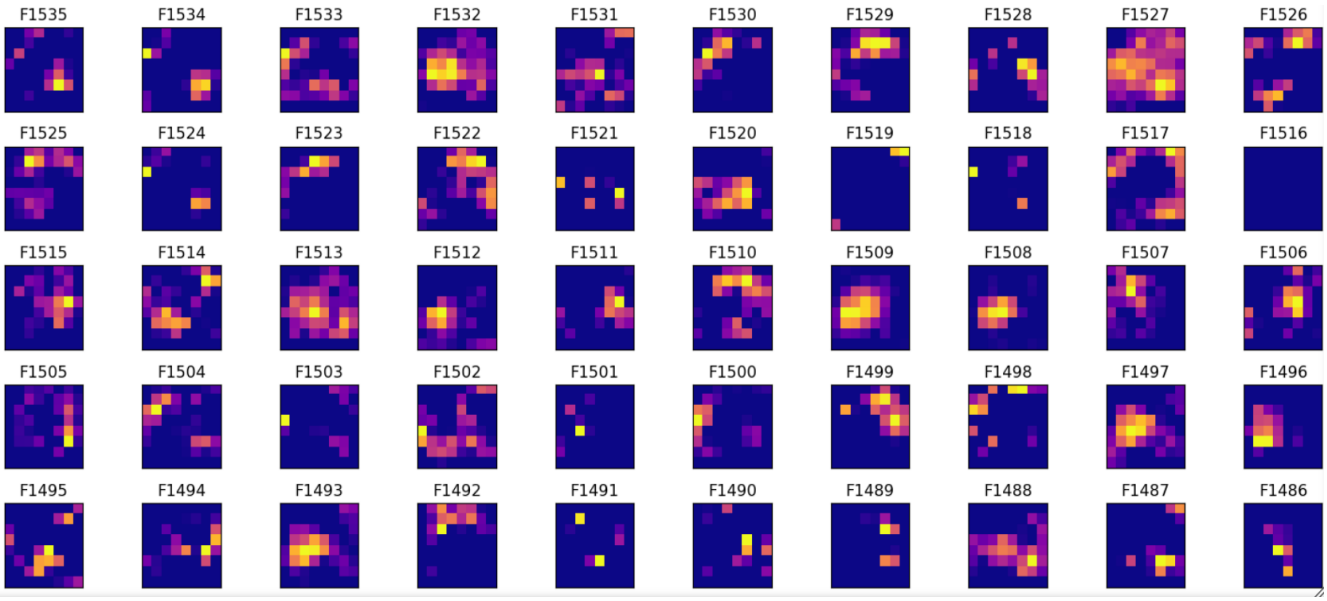


Figure 18:Activation Map of conv_7b_ac Layer from Baseline Model/Inception ResNet V2/

b. **Activation map from the final enhanced model:** similarly, we generated the activation map of the enhanced model using class activation mapping from the custom lightweight Reduction Module C.

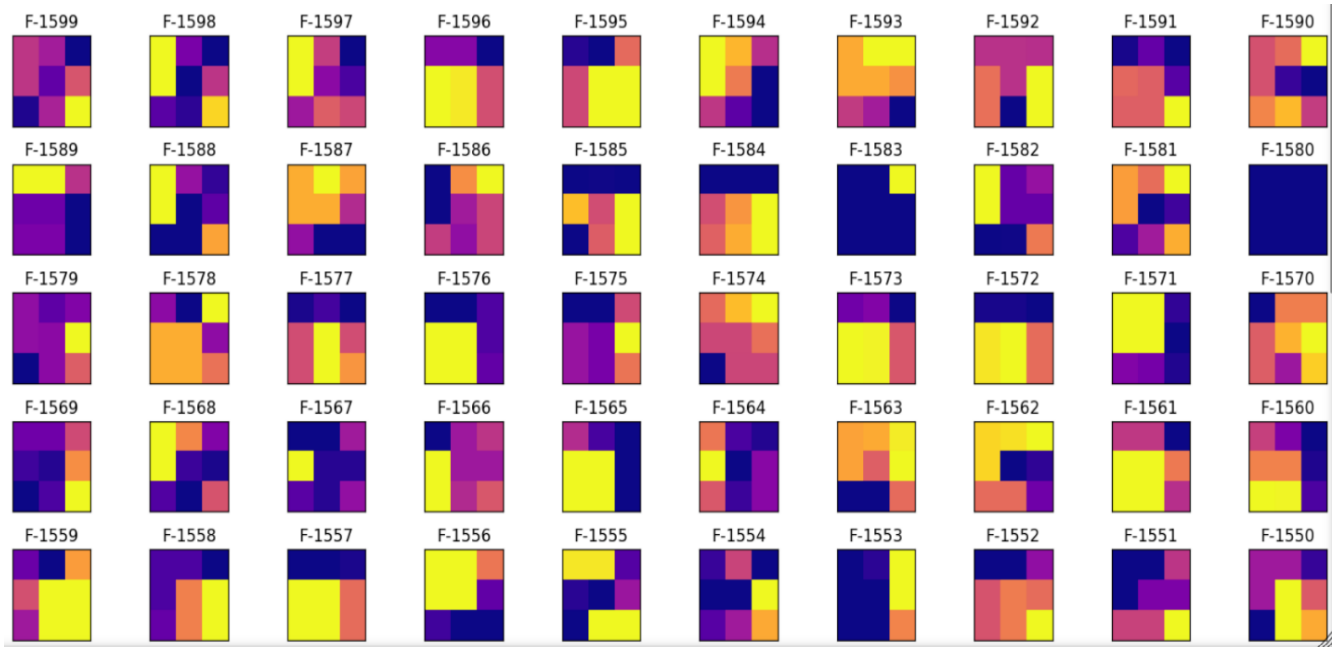


Figure 19: Activation Map of reduction_module_c Layer from Final Enhanced Model.

In order to understand the feature extraction capability of both baseline model and enhanced model, let's discuss the two main features (color distribution and activation intensity) that are analyzed from the activation maps generated from each model.

I. Color distribution

The number of colors in the activation map generated from the baseline model are more dominated by darker blue color indicating very few features are being extracted by the model. This shows that the model is less accurate.

And the activation maps generated from the enhanced model are dominated by several different colors indicating an increased number of different features are being extracted by the enhanced model. Increasing the model's accuracy to detect and classify grape leaf diseases.

Therefore, it can be concluded that the final enhanced model does well at detecting and classifying grape leaf diseases because it can extract a broader range of features from the input image and thus understand it more.

II. Activation intensity

To start with the activation intensity of the activation map generated by the baseline model, this activation map shows a weaker intensity meaning the model is focusing on features that are not useful. While the activation map generated by the enhanced model shows stronger intensity, meaning the model focuses on features that are useful for detecting and classifying grape leaf disease (shape of the diseased spots, surface texture of the leaf and color variation).

In general, the addition of reduction module C improves the model's sensitivity to important features by enhancing feature representation.

Compared to the baseline model, this enables accurate grape leaf disease detection and classification.

4.4.4.2 Visualizing the most specific and influential features where the final enhanced model focuses on using gradient weighted class activation mapping (Grad-CAM)

To better understand in which part of the leaf image our model is focusing, we use an effective visualization technique called Gradient-weighted Class Activation Mapping (Grad-CAM). Grad-CAM identifies which areas of the leaf image influences the most in the model's decision to detect and classify grape leaf disease. Grad-CAM allows for understanding the inner workings of deep learning models by clarifying the areas of interest inside an image.

To understand these concepts in practical terms, we have carried out Grad-CAM examinations on black rot disease. Now let's start by examining the Grad-CAM representation of the final convolutional layer for black rot, which is Reduction Module C. The image seen below is divided into three sections: the original image, the Grad-CAM heatmap and superimposed image. Here is the detailed interpretation of the images:

a. **Original image:** this section shows a grape leaf with visible symptoms of black rot disease. Most of the areas of the image are dominated by green color, but several areas are covered with a circular black spot with a yellow edge. which are the characteristics of the disease.
b. **Grad – CAM heatmap:** The Grad-CAM heatmap shows different varieties of colors indicating which region of the image contributes the most to the model's decision. The following colors were observed by the Grad-CAM red, yellow, orange, blue and light green. Here is their interpretation:
    i. **Red areas:** this color is the hottest color indicating higher activation intensity being detected. The model is most actively focusing its attention on these areas covered by the red colors. These colors are mostly seen on the edge of the leaf and on the

darker spots of the leaf. As a result, we can say that our model is mostly focusing on the disease specific features of the leaf image.

ii. **Yellow and orange areas:** These colors indicate that the model paid moderate attention to the edges of the darker spots, which frequently had yellow borders surrounding the black spots. This suggests that even though these areas are not as important as the red zones, they are still significant.

iii. **Blue and light green areas:** These are the areas of the heatmap that are the coolest and show the model's least level of attention. The healthy portions of the leaf are frequently represented by the blue and light green sections, indicating that these regions have little effect on the way the model makes decisions.

c. **Superimposed image:** This image makes it simple to observe where the model is focusing by combining the original leaf image with a heatmap. As we can see from the image, the model's focus changes across the leaf, as indicated by the combined colors. The heatmap's red and yellow colors highlight the darker dots and the areas that are surrounding them as important areas of interest.

In general, the Grad-CAM visualization shows that the model is successfully focusing on regions that are indicative of a disease affecting the leaf.

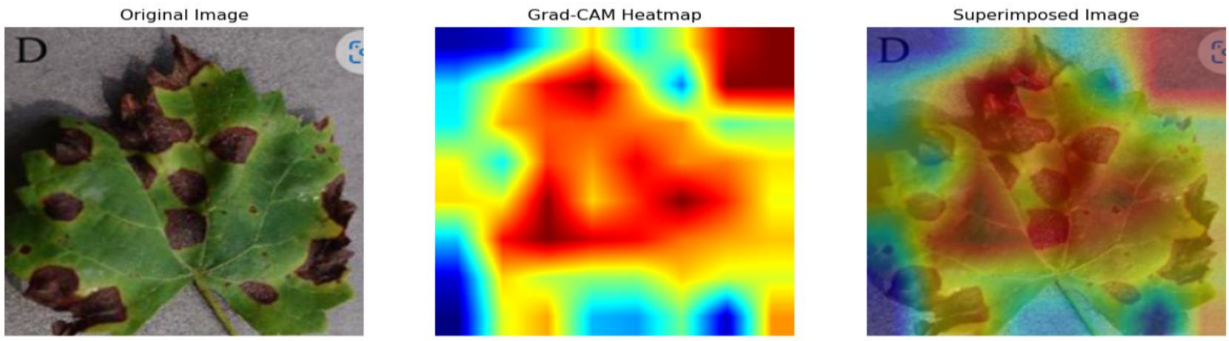


Figure 20: Grad-CAM Visualization for Black Rot Disease in Grape Leaf

4.4.5 Proposed Model Performance Comparison with Existing Models

Even though previous research in this area has done an average job of detecting and classifying grape leaf diseases. Yet, there are still gaps in their works, such as issues with generalization and reduced accuracy, that need to be filled with more studies. To tackle the first problem, we have employed three key strategies:

- **Data augmentation:** which creates diversified variations of pre-existing images.
- **Regularization techniques:** drop-out regularization technique was used. Which enables the model to concentrate only on relevant features, and
- **Oversampling technique**: to avoid bias towards any particular class, we also ensured that all grape leaf disease varieties were fairly represented in the dataset (balancing of classes in each dataset).

To address the second issue of accuracy, **Reduction Module C** plays a crucial role in improving the model's accuracy in detecting and classifying diseases in grape leaves, allowing the model to capture more feature maps while focusing on important features in the images which will help it make an accurate decision.

Proposed model performance summary: With a 99.89% accuracy rate on the validation set, the suggested model performs exceptionally well in detecting and classifying grape leaf disease. Below is a summary comparison of our model against existing models:

Table 9:Performance Comparsion of Existing Models with the Proposed Model

Research Title	Algorithm	Accuracy
Unhealthy Region of Citrus Leaf Detection Using Image Processing Techniques [9]	K-means clustering, GLCM, SVM for classification	96%
A Smartphone Application to Detection and Classification of Coffee Leaf Miner and Coffee Leaf Rust [10]	Color space analysis, Otsu method, GLCM, ANN	91.5%-99.1%
A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks [11]	Faster DR-IACNN with ResNet34, SE blocks, Inception-v1	81.1%

Transfer learning with Densenet201 architecture model for potato leaf disease classification [12]	DenseNet201 with various optimizers and dropout rates	95.2%
Deep Learning for Plant Identification and Disease Classification from Leaf Images [13]	Generalized Stacking Multi-output CNNs (GSMo-CNNs)	99.31%
Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks [14]	Memory-efficient CNN inspired by VGG16	93.3%
Using Deep Learning for Image-Based Plant Disease Detection [15]	GoogleNet with transfer learning	99.35%
ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network [16]	CNN architecture with fully connected and convolutional layers	91.2%
An Efficient Transfer Learning-based Approach for Apple Leaf Disease Classification [17]	Pre-trained EfficientNetV2S used as feature extractor	99.21
An optimized capsule neural networks for tomato leaf disease classification [18]	Optimized CapsNet to classify tomato leaf diseases	96.39
Corn Leaf Diseases Recognition Based on Convolutional Neural Network [19]	CNN architecture with several convolution and pooling layers	93.24
Deep Learning-Based Segmentation and Classification of Leaf Images for Detection of Tomato Plant Disease [20]	U-Net for segmentation and various InceptionNets for classification	99.3%
A lightweight Convolutional Neural Network Model for Identification of Grape Leaf Diseases [21]	GrapeNet with CBAM, Residual Blocks for feature extraction	86.25%
Plant disease detection using deep learning [22]	MobileNet architecture with transfer learning	95.05%
Enhancing Plant Disease Detection: A Novel CNN-Based Approach with Tensor Subspace Learning and HOWSVD-MD [23]	Pre-trained CNNs with HOWSVD and MDA for categorization	98.36%

Class-specific data augmentation for plant stress classification [24]	Automated genetics algorithm-based method for data augmentation	98%
Proposed Model	**Pre-trained Inception ResNet V2 with an integrated Reduction Module C**	**99.89%**

In plant disease detection research, a wide variety of approaches and classification accuracy are shown by comparative analysis. Many models use advanced techniques like transfer learning and convolutional neural networks (CNNs) to attain high accuracy rates, some researches have shown accuracies of over 99% (e.g., the works of [15], [17], and [20]). However, the suggested model, which makes use of the pre-trained Inception ResNet V2 with an integrated Reduction Module C, attains a remarkable accuracy of 99.89%, demonstrating its effectiveness in accurately detecting and classifying grape leaf diseases.

Chapter Five

5. Conclusion and Recommendations

5.1 Conclusion

The addition of lightweight Reduction Module C to the pretrained Inception ResNet V2 model provided three key benefits improved feature representation, improved computational complexity and improved accuracy, advancing the detection and classification of grape leaf disease. The enhanced model's highest F1 score of 99.89% showed an excellent balance between precision and recall. This advancement is crucial to sustainable agriculture because it guarantees precise disease detection and effective pesticide application to crops.

The enhanced model was able to demonstrate a superior feature extraction capability when compared with the baseline model. This shows the the model's improved capacity to concentrate on relevant features, such as color, texture, and lesion form, which are displayed through activation maps. Making it preferable for practical use in the agricultural sector.

One of the key achievements of this study is that, the enhanced model achieved a higher F1 score with only 0.21 million trainable parameters, showing our model is both accurate and efficient using less memory and processing power while maintaining higher performance. This makes it preferable for resource constrained devices including mobile-based platforms.

Here are detailed explanations showing how this study addressed each research question:

5.1.1 Research question 1: Which existing models are more suitable and how to enhance from existing one?

After examining existing pre-trained models for grape leaf disease detection, we concluded that it was appropriate to begin with Inception ResNet V2 as our baseline model, as it performs better than the other examined models, such as Xception and EfficientNet - B4. We then improved the model's accuracy by adding Reduction Module C which is capable of increasing the feature maps allowing the

network to capture more complex features, while reducing the spatial dimension of the input image.

5.1.2 Research question 2: what are the techniques used to determine the principal features that have significant impact on the detection and classification of grape leaf diseases?

A technique called gradient-weighted class activation mapping, or Grad-CAM, was used to understand the features that the model had learned. The areas in the grape leaf images that are essential to the model's decision-making process are highlighted in the heatmaps created by Grad-CAM. The regions of the leaf that the model concentrates on are shown on these maps, especially those that show signs of disease. It is clear from looking at these highlighted areas that the model is finding signals relevant to different grape leaf diseases.

5.1.3 Research question 3: What are the principal features that have a significant impact on the detection and classification of grape leaf disease?

The model mainly learns to recognize the following features based on the examination of the CAM and Grad-CAM.

a. **Color variations:** Color changes, such as yellowing, browning, dark streaks, or dark circular spots, are common symptoms of many grape leaf diseases. These discolored spots are often highlighted in the activation maps, indicating that color variation is a crucial characteristic used by the model to differentiate between healthy and diseased leaves.
b. **Texture:** The surface texture of leaves can be changed by diseases, giving them a rough, uneven, or dying appearance. The activation maps' focus on these areas suggests that texture is another crucial component the model takes into account.
c. **Shape of lesions:** Certain diseases are indicated by specific patterns and shapes of lesions. For example, the existence of circular spots, streak-like structures of spots, or irregular spots. The fact that the model focuses on certain shapes

suggests that it is aware of these patterns and applies them to its categorization procedure.

Although activation maps are a useful tool, they do not provide a full understanding of every characteristic that the model has learned. Because of their complexity, deep learning models can identify advanced and abstract patterns that extend beyond the basic visual characteristics like color, texture, and shape. As a result, the internal workings of the model continue to somewhat hide the exact nature of the features learned.

5.1.4 Research question 4: what are the techniques used to measure the performance of the model?

In order to assess the performance of our model and compare it with other models built throughout this study we applied a number of metrics like loss, accuracy, precision, recall , F1 score and computational complexity. Since we are dealing with disease classification both preventing the spread of disease (higher recall) and preventing resource (pesticide) wastage (higher precision) are greatly required. Therefore, F1 score will be used as a primary metric as it balances both recall and precision.

The final enhanced model's balanced F1 score of 0.9989 showed how well it addressed both kinds of errors, reducing false positives while guaranteeing the precise identification of true positives. This state of balance is essential for maintaining sustainable farming methods and efficient disease control, confirming the model's applicability for the job.

5.2 Recommendations

Even though the current research has been successful in addressing some of the gaps in the detection and classification of grape leaf diseases. There is still work that needs to be done in the future. The following are important areas to focus on in the future:

- Increasing Coverage of Plant Varieties and Diseases: the current model is only limited to detect and classify only four types of grape leaf diseases. And the future work should focus on expanding this capability to cover more types of diseases in various plant species. To achieve this, we will recommend performing the following methods:

- Collecting Diverse Datasets: collecting large data sets with images of various plant diseases affecting various plants.
- Model Adaptation: adapting and modifying this model to correctly detect and classify these extra diseases.

- Improving Model Usability and Deployment: to make the system more accessible from anywhere a transition of the application from desktop based to mobile based platform is greatly needed. And the main amendment that needed to be done to perform this transition is to reduce memory and CPU complexity by optimizing the model to be lightweight without affecting accuracy.

References

[1] S. Parihar and D. Sharma, "A brief overview on Vitis vinifera," Saudi Journal of Pathology and Microbiology, vol. 10, no. 12, pp. 231-239, 2021. Available: https://doi.org/10.36347/sajp.2021.v10i12.005.

[2] M. Szabó et al., "Black rot of grapes (Guignardia bidwellii) - A comprehensive overview," Horticulturae, vol. 9, no. 2, p. 130, 2023. Available: https://doi.org/10.3390/horticulturae9020130.

[3] L. Mugnai et al., "Esca (Black Measles) and Brown Wood-Streaking: Two old and elusive diseases of grapevines." Available: https://www.burgundy-report.com/wp/wp-content/uploads/2005/09/esca_report.pdf.

[4] A. J. Maia et al., "The control of isariopsis leaf spot and downy mildew in grapevine cv. Isabel with the essential oil of lemon grass and the activity of defensive enzymes in response to the essential oil," Crop Protection, vol. 63, pp. 57-67, 2014, doi: 10.1016/j.cropro.2014.05.005.

[5] S. Su et al., "A Review on Machine Learning Classification Techniques for Plant Disease Detection," in 2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS), Coimbatore, India, 2019, pp. 1-6, doi: 10.1109/ICACCS.2019.872841.

[6] A. Krizhevsky et al., "ImageNet classification with deep convolutional neural networks," in Advances in Neural Information Processing Systems 25, 2012. Available: https://proceedings.neurips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[7] J. Gu et al., "Recent advances in convolutional neural networks," arXiv, 2017. Available: https://doi.org/10.48550/arXiv.1512.07108.

[8] Leonida et al., "Survey of diseases affecting tropical fruit trees in Central Panay Island, Philippines," IOP Conference Series: Earth and Environmental Science, vol. 1208, p. 012024, 2023. Available: https://doi.org/10.1088/1755-1315/1208/1/012024.

[9] Gavhale et al., "Unhealthy region of citrus leaf detection using image processing techniques," 2014 International Conference for Convergence of Technology (I2CT), 2014. Available: https://doi.org/10.1109/I2CT.2014.7092035.

[10] G. L. Manso et al., "A smartphone application to detection and classification of coffee leaf miner and coffee leaf rust," Submitted on 19 Mar. 2019. Available: https://doi.org/10.48550/arXiv.1904.00742.

[11] X. Xie et al., "A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks," Frontiers in Plant Science, vol. 11, Article 751, Jun. 2020. Available: https://doi.org/10.3389/fpls.2020.00751.

[12] R. A. Charisma and F. D. Adhinata, "Transfer learning with Densenet201 architecture model for potato leaf disease classification," arXiv preprint arXiv:2402.03347, 2024. Available: https://doi.org/10.48550/arXiv.2402.03347.

[13] J. Yao et al., "Deep Learning for Plant Identification and Disease Classification from Leaf Images: Multi-prediction Approaches," Submitted on 25 Oct 2023. DOI: 10.48550/arXiv.2310.16273. Available: https://doi.org/10.48550/arXiv.2310.16273.

[14] C. R. Rahman et al., "Identification and Recognition of Rice Diseases and Pests Using Convolutional Neural Networks," Submitted on 3 Dec 2018. Available: https://doi.org/10.48550/arXiv.1812.01043.

[15] S. P. Mohanty et al., "Using Deep Learning for Image-Based Plant Disease Detection," Frontiers in Plant Science, vol. 7, 2016. Available: https://doi.org/10.3389/fpls.2016.01419.

[16] M. Agarwal et al., "ToLeD: Tomato Leaf Disease Detection using Convolution Neural Network," Procedia Computer Science, vol. 167, pp. 293–301, 2020. Available: https://doi.org/10.1016/j.procs.2020.03.225.

[17] M. H. Ashmafee et al., "An Efficient Transfer Learning-based Approach for Apple Leaf Disease Classification," arXiv, 2023. Available: https://doi.org/10.48550/arXiv.2304.06520.

[18] L. M. Abouelmagd et al., "An Optimized Capsule Neural Network for Tomato Leaf Disease Classification," J. Image Video Proc., vol. 2, 2024. Published 08 January 2024. Available: https://doi.org/10.1186/s13640-023-00618-9.

[19] M. Fadhilla et al., "Corn Leaf Diseases Recognition Based on Convolutional Neural Network," IT J. Research and Development, vol. 8, 2023. Available: https://doi.org/10.25299/itjrd.2023.13904.

[20] M. Shoaib et al., "Deep Learning-Based Segmentation and Classification of Leaf Images for Detection of Tomato Plant Disease," Frontiers in Plant Science, vol. 13, 2022. Available: https://doi.org/10.3389/fpls.2022.1031748.

[21] J. Lin et al., "GrapeNet: A Lightweight Convolutional Neural Network Model for Identification of Grape Leaf Diseases," Agriculture, vol. 12, no. 6, p. 887, 2022. Available: https://doi.org/10.3390/agriculture12060887.

[22] K. Hariharan et al., "Plant Disease Detection Using Deep Learning," International Journal of Innovative Research in Technology, vol. 7, no. 10, p. 278, 2021. Available: https://www.academia.edu/64696457/Plant_Disease_Detection_Using_Deep_Learning.

[23] A. Ouamane et al., "Enhancing Plant Disease Detection: A Novel CNN-Based Approach with Tensor Subspace Learning and HOWSVD-MD," arXiv, 2024. Available: https://doi.org/10.48550/arXiv.2405.20058.

[24] N. Saleem et al., "Class-Specific Data Augmentation for Plant Stress Classification," arXiv, 2024. Available: https://doi.org/10.48550/arXiv.2406.13081.

[25] S. Gholizadeh, "Top Popular Python Libraries in Research," JRAR, vol. 3, pp. 142-145, 2022. Available: https://doi.org/10.33140/JRAR.03.02.02.

[26] S. P. Mohanty et al., "Using Deep Learning for Image-Based Plant Disease Detection," Frontiers in Plant Science, vol. 7, Art. 1419, 2016. Available: https://doi.org/10.3389/fpls.2016.01419.

[27] Kaggle. (n.d.). Your machine learning and data science community. Available: http://www.kaggle.com. Accessed January 20, 2024.

[28] I.Goodfellow et al., Deep Learning. MIT Press, 2016. Available: http://www.deeplearningbook.org. Accessed: Jul. 5, 2024.

[29] C. Szegedy et al., "Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning," arXiv, 2016. Available: https://doi.org/10.48550/arXiv.1602.07261.

[30] M. Tan and Q. V. Le, "EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks," in International Conference on Machine Learning (ICML), 2019. Available: https://doi.org/10.48550/arXiv.1905.11946.

[31] F. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2016. Available: https://doi.org/10.48550/arXiv.1610.02357.

[32] J. Ilemobayo et al., "Hyperparameter Tuning in Machine Learning: A Comprehensive Review," Journal of Engineering Research and Reports, vol. 26, pp. 388-395, 2024. doi: 10.9734/jerr/2024/v26i61188.

[33] L. Alzubaidi et al., "Review of Deep Learning: Concepts, CNN Architectures, Challenges, Applications, Future Directions," Journal of Big Data, Published online March 31, 2021. Available: https://doi.org/10.1186/s40537-021-00444-8.

[34] J. R. Terven et al., "Loss Functions and Metrics in Deep Learning," Submitted on July 5, 2023. Available: https://doi.org/10.48550/arXiv.2307.02694.

[35] D. P. Kingma and J. L. Ba, "Adam: A Method for Stochastic Optimization," International Conference on Learning Representations, Submitted on December 22, 2014. Available: https://doi.org/10.48550/arXiv.1412.6980.

[36] C. Szegedy et al., "Rethinking the Inception Architecture for Computer Vision," Submitted on December 2, 2015. Available: https://doi.org/10.48550/arXiv.1512.00567.

Appendix

Appendix A: Questionnaires

The following questionnaires were conducted to assess the awareness of grape producers on detection and classification of grape leaf diseases.

I Demographic information
   a. What is your age?
   b. What is your level of education?
   c. How many years have you been involved in grape cultivation?

II Disease Awareness and Identification
   a. How familiar are you with grape leaf diseases like Black Rot, Black Measles, and Leaf Blight diseases of grape leaves?
      i. Very familiar
      ii. Somewhat familiar
      iii. Not familiar at all
   b. What symptoms do you observe when these diseases are present?
   c. In your grape harvests, how do you currently diagnose Black Rot, Black Measles and Leaf Blight?

III Detection and Diagnosis
   a. Which method do you use to diagnose diseases?
      i. visual inspection
      ii. laboratory testing
      iii. procedures assisted by technology
   b. How accurate do you think these techniques are in classifying these diseases?

IV Challenges Encountered
   a. What challenges do you encounter in identifying diseases of grape leaves?
   b. Has it been difficult for you to find reliable source of information for monitoring Black Rot, Black Measles and Leaf Blight?

V Technology usage
   a. How interested are you to use technologies for disease detection?

b. From the disease detection application that diagnose Black Rot, Black Measles ad Leaf Blight what essential characteristics would you look for?

i. Offline access

ii. Mobile compatibility

iii. Instant feedback

iv. Simple user interface

Appendix B: Sort of code used

This section shows some parts of the code that contribute to the development of the system.

I Reduction Module C

The code shown below refers to the implementation of the factorized version of Reduciton Module C.

```
import tensorflow as tf

class ReductionModuleC(tf.keras.layers.Layer):
    def __init__(self, **kwargs):
        super(ReductionModuleC, self).__init__(**kwargs)

        # Define layers in the reduction module C
        self.convolution_layer_1x1 = tf.keras.layers.Conv2D(16, kernel_size=(1, 1),
padding='same', strides=(1, 1))  # Add 1x1 convolution layer
        self.convolution_layer_1x3 = tf.keras.layers.Conv2D(32, kernel_size=(1, 3),
padding='valid', strides=(3, 2))
        self.convolution_layer_3x1 = tf.keras.layers.Conv2D(32, kernel_size=(3, 1),
padding='valid', strides=(2, 3))
        self.batch_normalization_layer_1x1 = tf.keras.layers.BatchNormalization()
        self.batch_normalization_layer_1x3 = tf.keras.layers.BatchNormalization()
        self.batch_normalization_layer_3x1 = tf.keras.layers.BatchNormalization()
        self.activation_layer = tf.keras.layers.ReLU()  # ReLU activation
        self.dropout_layer = tf.keras.layers.Dropout(0.5)
        self.pooling_layer = tf.keras.layers.MaxPooling2D(pool_size=(3, 3),
padding='valid', strides=(2, 2))
        self.concatenation_layer = tf.keras.layers.Concatenate()

    def call(self, inputs):
        # Apply 1x1 convolutional layer
        convolution_output_1x1 = self.convolution_layer_1x1(inputs)

        # Apply batch normalization
        batch_normalized_output_1x1 =
self.batch_normalization_layer_1x1(convolution_output_1x1)

        # Apply ReLU activation
        activated_output_1x1 = self.activation_layer(batch_normalized_output_1x1)

        # Apply 1x3 convolutional layer
```

```
        convolution_output_1x3 = self.convolution_layer_1x3(activated_output_1x1)

        # Apply batch normalization
        batch_normalized_output_1x3 =
self.batch_normalization_layer_1x3(convolution_output_1x3)

        # Apply ReLU activation
        activated_output_1x3 = self.activation_layer(batch_normalized_output_1x3)

        # Apply 3x1 convolutional layer
        convolution_output_3x1 = self.convolution_layer_3x1(activated_output_1x1)

        # Apply batch normalization
        batch_normalized_output_3x1 =
self.batch_normalization_layer_3x1(convolution_output_3x1)

        # Apply ReLU activation
        activated_output_3x1 = self.activation_layer(batch_normalized_output_3x1)

        # Concatenate outputs of 1x3 and 3x1 convolution layers
        concatenated_output = self.concatenation_layer([activated_output_1x3,
activated_output_3x1])

        # Apply dropout
        dropout_output = self.dropout_layer(concatenated_output)

        # Apply max pooling layer
        pooling_output = self.pooling_layer(inputs)

        # Concatenate outputs of dropout and pooling layers
        final_output = self.concatenation_layer([dropout_output, pooling_output])

        return final_output
```

II Loading the Dataset

```
import os
# Specify the directory path
base_path = r'C:\Users\dania\Desktop\original_dataset'
# List of classes (subdirectories)
classes = [
    'Grape___Black_rot',
    'Grape___Esca_(Black_Measles)',
    'Grape___healthy',
    'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)'
     ]
# Dictionary to store counts for each class
```

```
class_counts = {}
# Loop through each class directory and count the images
for class_name in classes:
    class_path = os.path.join(base_path, class_name)
    file_count = len(os.listdir(class_path))
    class_counts[class_name] = file_count

# Print the total number of images for each class
for class_name, count in class_counts.items():
   print(f"Class '{class_name}' has {count} images.")
```

output:

Class 'Grape___Black_rot' has 1180 images.
Class 'Grape___Esca_(Black_Measles)' has 1383 images.
Class 'Grape___healthy' has 423 images.
Class 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)' has 1076 images.

III Dataset preprocessing

a. Dataset splitting

```
import os
import random
import shutil
# Set the paths
dataset_path = r'C:\Users\dania\Desktop\original_dataset'
output_path = r'C:\Users\dania\Desktop\split_dataset'
# Define the ratios
train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15
# Create the output directory
os.makedirs(output_path, exist_ok=True)
# Define the directories for train, validation, and test sets
train_dir = os.path.join(output_path, "train")
val_dir = os.path.join(output_path, "val")
test_dir = os.path.join(output_path, "test")
os.makedirs(train_dir, exist_ok=True)
os.makedirs(val_dir, exist_ok=True)
```

```
os.makedirs(test_dir, exist_ok=True)
# Iterate through each class folder
for class_name in os.listdir(dataset_path):
    class_path = os.path.join(dataset_path, class_name)
    # Create corresponding directories in train, validation, and test sets
    os.makedirs(os.path.join(train_dir, class_name), exist_ok=True)
    os.makedirs(os.path.join(val_dir, class_name), exist_ok=True)
    os.makedirs(os.path.join(test_dir, class_name), exist_ok=True)
    # Get a list of all images in the class folder
    images = os.listdir(class_path)
    # Shuffle the images
    random.shuffle(images)
    # Split the images into train, validation, and test sets
    train_split = int(len(images) * train_ratio)
    val_split = int(len(images) * val_ratio)
    train_images = images[:train_split]
    val_images = images[train_split:train_split + val_split]
    test_images = images[train_split + val_split:]
    # Copy images to their respective directories
    for img in train_images:
        src = os.path.join(class_path, img)
        dest = os.path.join(train_dir, class_name, img)
        shutil.copy(src, dest)
    for img in val_images:
        src = os.path.join(class_path, img)
        dest = os.path.join(val_dir, class_name, img)
        shutil.copy(src, dest)
    for img in test_images:
        src = os.path.join(class_path, img)
        dest = os.path.join(test_dir, class_name, img)
        shutil.copy(src, dest)
print("Dataset split completed successfully.")
```

b. Dataset balancing using oversampling techniques

```
import os
import numpy as np
from keras.preprocessing.image import ImageDataGenerator
# Function to count files in a directory
def count_files(directory):
    count = 0
    for root, dirs, files in os.walk(directory):
        for file in files:
            count += 1
    return count
# Define the paths to the split dataset
train_dir = r"C:\Users\dania\Desktop\split_dataset\train"
val_dir = r"C:\Users\dania\Desktop\split_dataset\val"
test_dir = r"C:\Users\dania\Desktop\split_dataset\test"
# Calculate the target number of samples for each class
max_samples     =     max(max(train_counts.values()),     max(val_counts.values()),
max(test_counts.values()))
# Create an ImageDataGenerator for augmentation
datagen = ImageDataGenerator(
    rotation_range=20,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest')
# Augment images in the train set
for class_name in os.listdir(train_dir):
    class_path = os.path.join(train_dir, class_name)
    count = count_files(class_path)
    while count < max_samples:
        # Randomly select an image
        img_name = np.random.choice(os.listdir(class_path))
        img_path = os.path.join(class_path, img_name)
        img = np.expand_dims(plt.imread(img_path), axis=0)
        # Generate augmented images
```

```
        for batch in datagen.flow(img, batch_size=1, save_to_dir=class_path,
save_prefix='aug', save_format='jpg'):
            count += 1
            if count >= max_samples:
                break
# Augment images in the validation set
for class_name in os.listdir(val_dir):
    class_path = os.path.join(val_dir, class_name)
    count = count_files(class_path)
    while count < max_samples:
        # Randomly select an image
        img_name = np.random.choice(os.listdir(class_path))
        img_path = os.path.join(class_path, img_name)
        img = np.expand_dims(plt.imread(img_path), axis=0)
        # Generate augmented images
        for batch in datagen.flow(img, batch_size=1, save_to_dir=class_path,
save_prefix='aug', save_format='jpg'):
            count += 1
            if count >= max_samples:
                break
# Augment images in the test set
for class_name in os.listdir(test_dir):
    class_path = os.path.join(test_dir, class_name)
    count = count_files(class_path)
    while count < max_samples:
        # Randomly select an image
        img_name = np.random.choice(os.listdir(class_path))
        img_path = os.path.join(class_path, img_name)
        img = np.expand_dims(plt.imread(img_path), axis=0)
        # Generate augmented images
        for batch in datagen.flow(img, batch_size=1, save_to_dir=class_path,
save_prefix='aug', save_format='jpg'):
            count += 1
            if count >= max_samples:
                break
print("Dataset balancing completed successfully.")
```

IV Building the Model with Checkpointing and Early Stopping

```
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping
# Define the filepath for saving the model
filepath5 = ' updated_factorized_reduction_c_model64_lr_01_dense_128REG0.5.h5'
# Define the ModelCheckpoint callback to save the best model based on validation accuracy
checkpoint = ModelCheckpoint(filepath5, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')
# Define the EarlyStopping callback to stop training when validation accuracy stops improving
early_stopping = EarlyStopping(monitor='val_accuracy', patience=6, verbose=1, mode='max', restore_best_weights=True)
# Define the number of steps per epoch for both training and validation
train_steps_per_epoch = train_generator.n // train_generator.batch_size
valid_steps_per_epoch = valid_generator.n // valid_generator.batch_size
# Fit the model with validation data
history5 = model5.fit(train_generator,
          epochs=20,
          steps_per_epoch=train_steps_per_epoch,
          validation_data=valid_generator,
          validation_steps=valid_steps_per_epoch,
          callbacks=[checkpoint, early_stopping])
```

Appendix C: Algorithm Grape Leaf Disease Detection and Classification

Steps of The Algorithm

I Step 1: Dataset collection and preparation

- Source: collect sets of grape leaf images from Kaggle, covering four classes.
- Class distribution: Initial classes include 4062 images, with distributions as follows:
    - Grape Black Rot: 1180 images
    - Grape Esca (Black Measles): 1383 images
    - Grape Healthy: 423 images
    - Grape Leaf Blight: 1076 images

II Step 2: Data Preprocessing

- Split the Dataset: Divide data into training (70%), validation (15%), and test (15%) sets.
- Balance the Dataset: we applied an oversampling method to avoid class imbalance.
- Augment the Data: this increases the number of samples in training set from 3746 to 19356 preventing overfitting issue.
- Normalization and Resizing: this resize and normalizes all sets of images to match the requirements of pretrained models.

III Step 3: Model selection

- After building several models using each pretrained CNN architecture (Inception ResNet v2, EfficientNet B4 and Xception), we then selected the best performing pretrained model based on F1 score on validation set as primary metric. Based on the finding pretrained Inception ResNet V2 was selected as the baseline model as it outperforms all other pretrained models.

V Step 4: Model enhancement

- Given the baseline model we improved the performance of this model by inserting a lightweight Reduction Module C. This module enables the model to capture complex

features while ignoring irrelevant information, leading to improved feature extraction capability.

Hyperparameter Configuration

- Learning Rate: 0.0001
- Dropout Rate: 0.5
- Optimizer: Adam
- Loss Function: Categorical Cross-Entropy
- Early Stopping Patience: 6 epochs
- Model Checkpointing: Based on validation accuracy
- Epochs: 20