Crop Disease Detection Using Deep Learning Techniques on Images

By

Kinjal Vijaybhai Deputy

A thesis submitted in partial fulfillment

Of the requirements for the degree of

Master of Science (M.Sc.) Computational Sciences

The Office of Graduate Studies

Laurentian University

Sudbury, Ontario, Canada

THESIS DEFENCE COMMITTEE/COMITÉ DE SOUTENANCE DE THÈSE
Laurentian Université/Université Laurentienne
Office of Graduate Studies/Bureau des études supérieures

Title of Thesis
Titre de la thèse Crop Disease Detection Using Deep Learning Techniques on Images

Name of Candidate
Nom du candidat Deputy, Kinjal

Degree

Diplôme Master of Science

Program
Programme Computational Sciences

Date of Defence
Date de la soutenance May 23, 2023

APPROVED/APPROUVÉ

Thesis Examiners/Examinateurs de thèse:

Dr. Kalpdrum Passi
(Supervisor/Directeur(trice) de thèse)

Dr. Ratvinder Grewal
(Committee member/Membre du comité)

Dr. Abdel Omri
(Committee member/Membre du comité)

Dr. Jyoti Singh Kirar
(External Examiner/Examinateur externe)

Approved for the Office of Graduate Studies
Approuvé pour le Bureau des études supérieures
Tammy Eger, PhD
Vice-President Research (Office of Graduate Studies)
Vice-rectrice à la recherche (Bureau des études supérieures)
Laurentian University / Université Laurentienne

ACCESSIBILITY CLAUSE AND PERMISSION TO USE

I, **Kinjal Deputy**, hereby grant to Laurentian University and/or its agents the non-exclusive license to archive and make accessible my thesis, dissertation, or project report in whole or in part in all forms of media, now or for the duration of my copyright ownership. I retain all other ownership rights to the copyright of the thesis, dissertation or project report. I also reserve the right to use in future works (such as articles or books) all or part of this thesis, dissertation, or project report. I further agree that permission for copying of this thesis in any manner, in whole or in part, for scholarly purposes may be granted by the professor or professors who supervised my thesis work or, in their absence, by the Head of the Department in which my thesis work was done. It is understood that any copying or publication or use of this thesis or parts thereof for financial gain shall not be allowed without my written permission. It is also understood that this copy is being made available in this form by the authority of the copyright owner solely for the purpose of private study and research and may not be copied or reproduced except as permitted by the copyright laws without written authority from the copyright owner.

Abstract

Agriculture is a field which is referred to as the main sector for the development of the economy in various countries, and it is also providing food to the large population of the world despite various limitations and boundaries. Food security is threatened by several factors including climate change, the decline in pollinators, plant diseases and others. Different efforts have been developed to prevent crop loss due to infections in the plants. The advancement in technology is helping farmers in developing different systems that can help in reducing the problem. Smartphones specifically offer very novel ways to identify diseases because of their computing power, high resolution displays, and extensive built-in sets of accessories, such as advanced HD cameras. This leads to a situation where disease diagnosis based on automated image recognition is needed. Image recognition is made possible by applying a deep learning approach. So the research is aimed to analyze deep learning-based image detection techniques to identify the various diseases in the plants. The "PlantVillage" dataset has been used to train models. Deep learning Architectures such as AlexNet and GoogleNet, ResNet50 and InceptionV3 are used. Two approaches are used to train the model: 'training from scratch' and 'transfer learning'. It was found from the results of the primary analysis that the GoogleNet leaves behind the AlexNet, ResNet50 and InceptionV3 in training from scratch approach. And ResNet50 performed best in transfer learning.

Keywords: Machine Learning, Deep Learning, crop disease, agriculture, Image detection

Acknowledgements

I am grateful to my professor, Dr. Kalpdrum Passi for his significant support and insights that helped me during my studies. It was golden opportunity to do the research in his guidance.

I'd like to thank my friends for encouraging me. I would like to thank my loving parents, who supported me during my studies financially. It would not have been possible without their support. Thank you very much.

Abbreviation

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
SGD	Stochastic gradient descent
CNN	convolutional neural network

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Background

Agriculture is crucial in developing countries where food security is becoming a major problem. A technology that shows promise for detecting plant diseases is hyperspectral imaging. Hyperspectral data cubes include redundant information, which makes deep learning identification of plant illnesses more believable. The purpose of this study is to describe a deep learning-based method for detecting crop diseases from images. Thus, the main objectives will be focusing on the technology that is based on crop disease detection and its types. Further, it eventually focuses on the predictions of agriculture and research applications that are using the automated phenotyping-based platform.

Panchal et al. [1], stated that as a result of transportation issues, plant diseases, and a lack of storage facilities, the crops were lost. The disease is a big problem that needs to be handled because it causes more than 15% of the world's crops to be lost. It is necessary to have an automated system that can detect these diseases and guide farmers in taking the proper action to control crop loss. Because timely detection is challenging in different parts of the infrastructure, crop diseases are also a significant component of food security. The farmers relied on a variety of age-old methods, but not all of them were equally successful in identifying plant illnesses with their unaided eyes. Computer vision can help to detect the product defects and sort the products by color, size, weight, rapines and other factors. Farmers can use computer vision to analyze their crops, it can help them

work faster and better. Chen et al. [2] asserted that deep learning, because of its extensive library, is now the greatest developer-friendly and user-friendly environment for applying computer vision techniques in the field of agriculture. Thus, deep learning provides useful techniques that helps computers to understand what naturally comes to humans.

The main barrier inhibiting the spread of agriculture internationally is plant diseases, which cause yearly enormous losses. According to Jain et al. [3] The management of plant diseases has garnered a lot of interest. For the purpose of putting into place efficient prevention measures, plant diseases must be discovered and treated as soon as feasible. Even though the impacts of plant diseases are extremely complicated and variable, the majority of forest producers in conventional agricultural and forestry production can identify the species and severity of a disease based on their prior experiences with plant illnesses. For forest farmers to do this, they must be able to identify disease indications. Nagasubramanian et al. [4] Asserted that ignorance will result in uneven plant disease identification, improper treatment, and ultimately a delay in the treatment time, which will result in unneeded economic losses. Even with the aid of experts, a disease's diagnosis will take time. Therefore, developing an automated system for categorizing and identifying plant diseases is crucial. As a result of recent developments in computational systems, computer vision technologies are increasingly being employed to detect plant diseases.

Plant or leaf disease costs money and jeopardizes the development of numerous agricultural goods globally [5]. The inappropriate use of pesticides and fungicides is due to the failure to recognize illnesses, bacteria, and viruses in plants. Since biological characteristics of diseases are of particular interest to scientists, they have become quite interested in plant diseases. The use of

modern technology in precision farming results in improved decisions. Expert visual inspections and biological investigations are frequently used in plant diagnosis. This tactic often costs money and time. Identification of plant diseases using complex and robust methods is essential to resolving these problems. To improve the effectiveness of disease identification, conventional machine learning (ML) approaches have been implemented in agricultural operations. Recent examples of deep learning (DL), a type of machine learning, have shown its outstanding capacity to find, identify, and categorize things in the real world. Conversely, this shift in agricultural research has resulted in DL-based fixes [6]. State-of-the-art outcomes employing DL techniques have been reached for agricultural tasks like harvesting fruit, identifying plants, and differentiating between crops and weeds. The present research emphasis has been on identifying crop diseases, which is a major agricultural concern.

The illness severity is essential to forecast the production and to suggest treatment plan for the plants. Where the illness's severity is accurately and rapidly diagnosed, it helps to reduce the cost of treatment. According to [7], the severity of a plant disease is determined by trained experts by evaluating plant tissues visually. The costly and inadequate study of human illness is a contributing factor in the slow development of modern agriculture. Precision agriculture, high-throughput plant phenotyping, smart greenhouses, and other industries are highly interested in automated disease diagnostic models as a result of the growing use of digital cameras and the development of computer vision. In this study, deep learning models for autonomous image-based diagnosis of plant disease severity are presented. It was motivated by the advancement in deep learning for image-based plant disease detection. The severity of each illness was highlighted in annotations made on photographs of healthy and black rot apples from the public PlantVillage collection [8].

To assess the most effective educational program and network architecture, we start from scratch and build shallow networks of different depths to enhance the pre-trained, state-of-the-art deep networks.

Additionally, great progress has been made recently in computer vision development with the use of deep learning technology, which has paved the way for the widespread use of smartphones. These technologies aid in the proper management of the circumstances even though climate change, pollinators, and other factors continue to pose a threat to food safety [9]. Plant diseases are seriously affecting smallholder farmers whose livelihoods depend on growing healthy crops, in addition to endangering global food security. It has been found that deep learning is considerably superior to the conventional approach in these problems in terms of the advancements in digital picture processes [9]. However, because it uses machine vision technology to examine photographs and determine whether or not disease and pests can be seen in plant images, plant disease represents a very important research area in machine vision. In the beginning, machine vision-based plant disease detection equipment were used in agriculture, taking the role of the conventional naked eye method of identification as redundant. Using deep learning for plant disease detection can help in treating the plants early on to reduce the negative impact of plant diseases on agricultural production. Ashqar and Abu-Naser [10] stated that plant diseases represent a major threat to smallholder farmers that depend on healthy crops to survive with 80% of the global production of agriculture being affected. Thus it is found that disease identification is a crucial step for disease management and traditional approaches that help to identify those diseases are done by visiting local plant clinics.

1.2 Research Aim

The aim of this work is to analyse deep learning-based image detection techniques to identify crop diseases.

1.2.1 Research Objectives

- To analyse the Image-based plant disease detection technique.
- To describe the feature extraction for disease identification using deep learning.
- To analyse the types of crop diseases and relevant areas of sensors used during crop disease detection
- To identify the challenges that occur during crop disease detection.
- To recommend advanced image detection techniques for improving crop disease detection.

1.2.2 Research Question

- What is Image-based plant disease detection?
- How deep learning helps in crop disease detection?
- What are the issues that occur during crop disease detection?
- What is feature extraction in disease identification using deep learning techniques?

1.2.3 Problem Domain

The crop disease alludes to a major issue of low agricultural productivity. Guo et al. [11] stated that farmers face issues in identifying the crop diseases for controlling and detecting them. Thus, fast detection is beneficial for those farmers to avoid further losses. In the modern times if agriculture gets affected by crop diseases, it can be harmful for a country. The food demand which

is growing exponentially for the production of agronomy requires control of plant diseases. To boost agricultural production and meet food demand, scientists, farmers, analysts, experts, and the government all work together and employ various tactics [12]. The precise detection of plant diseases is a significant problem that has an impact on crop production. Another problem that arises during the detection is the appropriate application of the detection methods. The variables that can alter the global climate and plant diseases are at the disposal of farmers who are dealing with these issues. Crop production is decreasing, and farmers are thinking about committing suicide, for a variety of possible reasons. Because it requires a lot of time, accuracy, money, and good crop quality, visually inspecting crops is a challenging task. The majority of recent studies have mostly concentrated on classifying plant diseases [5]. The difficult task of diagnosing plant diseases, which necessitates both the location and classification of the disease in the plant, has not received nearly enough attention. For the purpose of detecting or identifying crop diseases using high-level DL structures, none of the aforementioned methods has been thoroughly studied. Deep learning (DL) is a branch of machine learning that has been proven to be very successful in real-world item identification, recognition, and classification tasks [13]. As a result, more and more DL-based approaches are being used in agricultural research. For agricultural tasks including crop/weed discrimination, fruit harvesting, and plant recognition, state-of-the-art outcomes using DL approaches have been attained.

1.2.4 Overview of Research Methodology

The explained research is based on the aim of analyzing crop disease with the help of deep learning-built image detection techniques. Further, to accomplish its results positively, the research will be focusing on some keywords such as image-based crop disease detection technology, types

of plant disease, relevant areas of sensors in crop disease detection, feature extractions for disease identification in deep learning, and more. Thus, the study will effectively help to describe the analysis of image-based crop disease detection with the help of deep learning. Also, it will help to describe the challenges, problems, types, and feature extraction, which is more helpful in plant disease detection.

1.3 Outline of the Thesis

The thesis is organized in six chapters, namely the introduction, literature review, Data Preparation, Deep Learning architectures, results and conclusion. The problem statement and the objectives are stated in the opening section of chapter 1 along with the study's background. The research questions, objectives, and significance of the study are all included in this chapter. The literature review is discussed in Chapter 2 which covers a variety of prior research. The data is discussed in Chapter 3. The deep learning architectures that were used to implement the crop disease detection are discussed in Chapter 4. The results and findings of methods are presented in chapter 5. The main conclusions of the study are presented in Chapter 6. Chapter 6 also highlights any limitations and suggestions for further research.

Chapter 2

Literature review

2.1 Image-Based Plant Disease Detection

The image-based plant disease detection technology is recently represented in various areas. Crop waste is representing enhanced disease, which becomes a critical identification method of disease [1]. Currently in developing countries, most of the population is based on agriculture in the form of direct and indirect energy. It represents the significant usage of application-based plant disease detection that helps the farmers to understand the reason behind the disease based on the plant's size, the colour of the leaf, the size of the leaf, and the growth pattern. In the current generation with the usage of smartphones all over the world, it is easy to click pictures of plants. Where various peoples have internet access also across the globe. Currently, for their convenience and to use a variety of applications, more than 300 million individuals have access to the internet. Although the government has access to a variety of tools, including a 24-hour helpline number for farmers to place orders and obtain answers to their queries, it can be difficult to effectively assist those who live in rural areas when they are having issues finding solutions to their problems. Self-paced image-based disease identification is a simple answer to this issue. Furthermore, according to van Bruggen et al. [14] crop disease is expanding globally and needs to be addressed if pesticides and insecticides are to be used to offer a short-term but ultimately beneficial answer. The chemicals have side effects on the crop which can ultimately hurt the health of the citizens. Currently, AI has spread in various domain areas and can be helpful in agriculture problems as well. The crops and leaves are also crucial in order to provide details regarding the amount and nature of horticultural

yield. Some of the factors affecting food production includes soil sterility, the presence of weeds, and climate change. However, in addition to these losses, leaf and plant diseases pose a global threat to the development of numerous agricultural goods [15]. Following the diagnosis of failure infection in plants, insufficient pesticide and fungicide usage occurs. The disease plan has therefore heavily taken into consideration the scientific community, with an emphasis on biological illness aspects. In precision farming, decision-making is optimized using the most cutting-edge technology. Expert visual inspection and biological assessments are used to complete the plant diagnosis when necessary. The method is well-known, labor-intensive, and cost-effective. It is essential to identify the plant disease using sophisticated and clever techniques in order to treat these issues.

Inexperienced farmers often encounter challenges due to their limited knowledge and lack of practical experience. These challenges become particularly pronounced as crops become increasingly vulnerable to the effects of climate change, requiring expert management that experienced farmers are better equipped to handle [16]. In agriculture, it is essential to recognize plant diseases since they have a significant impact on crop output and product quality. Viral illnesses that go untreated can have catastrophic implications for the nation's economy and food supply [17]. To create new and precise methods for identifying plant diseases, researchers from several academic disciplines, including microbiology, agronomy, and plant science, are collaborating. The utilization of subject-matter experts and specialized tools is necessary for strategies that exploit domain expertise. It is now possible to diagnose diseases solely using picture data as a result of developments in the computational processing of high-dimensional data, such as images. Spotting diseases in image data can be considered as a visual anomaly detection task

[18]. Identifying or putting strange observations in data is the work of anomaly detection. It is critical to pinpoint these issues or abnormal occurrences, which can include electricity theft, dishonest business dealings, strange illnesses, product flaws, etc., because these anomalous data points may be connected to a variety of issues or abnormal happenings. The different data sets that can be used for anomaly identification are due to the rarity of anomalous events. Plant disease data sets are not an exception and frequently display a size imbalance [19]. To categorize and identify plant diseases, deep learning technology is also used. A significant issue now is the detection of plant diseases. Both the quantity and the quality of agricultural production can be affected by plant diseases. The crop-growing sector depends on early disease detection. Enhancing agricultural production is the primary goal of identifying and categorizing plant diseases. As a result of a shift in consumer behaviour and attitudes toward processed foods, as well as the accessibility of smart devices, internet connectivity, and the most recent technologies, a modern subset of agriculture known as "precision agriculture" or "smart farming" has, however, exploded in popularity. For smart farming and sustainable agriculture, early identification of any plant stress has proven to be a big challenge.

Furthermore, Deep learning (DL) and machine learning (ML) algorithms are used to carry out agricultural tasks. Deep learning is the subset of machine learning that is exceptionally effective for real-world objectives based on rearrangement, detection, and classification reasons [20]. DL-based treatments are becoming the main focus of agricultural research. In agricultural tasks like crop and weed discrimination, plant restructuring, and fruit harvesting, the adoption of DL methods has led to state-of-the-art results. The DL models have also been used to categorize plant diseases using a well-known DL architecture. The study described that the modified version of

algorithms helps to enhance the performance of classification in diseases of various plant species. Additionally, Saleem et al. [5] stated that the convolutional neural network (CNN) and DL optimizer attain better results in plant disease classification. The CNN model is utilised to categorize the results of the enhanced plant disease categorization. The deep learning architecture has fixed input sizes and acts as a broker in the case where the input size is not fixed. This helps to make the input data consistent and easier to handle, which can improve the accuracy and efficiency of the deep learning model. Ultimately, this can make the model more effective and useful for various applications. For categorizing various plant diseases, MobileNet models are used. Similar to this, other research has mostly concentrated on advanced training strategies that analyse ways that aid in assessing the effectiveness of AlexNet and GoogleNet, which were both entirely new systems, and in transferring the learning methodology. The value of the fine-tuning method can be seen by contrasting the most complex DL structures for plant disease categorization. To address the problem of task object categorization, meta architectures are utilised to categorise and localise the objectives on a single platform. Complex agricultural practices were found to be engaged in the reorganisation of plant diseases by DL methods [5]. Deep learning has been used to produce models for identifying and evaluating plant diseases. Two distinct types of methodologies are designed and compared for analysis in order to achieve automated pest identification using deep learning techniques. Results showed how recognising deep learning approaches helps in providing better results than those of rival technologies [21]. When diagnosing the disease in cassava leaves, the single shot multibox detector (SSD), which is used in the DL technique, helps to produce results that are satisfactory. As a result, it is discovered that deep learning is the most precise and accurate paradigm for the identification of plant diseases.

2.2 Relevant areas of image sensors used in plant disease detection

A huge size of current development in path systems using various kinds of sensitive sensors and multiple data analysis pipelines helps to provide the various kind of sensor systems. It is classified as an optical sensor along with RGB, multi- and hyperspectral reflectance, thermal, and fluorescence imaging sensors [21]. Digital photographic images are extremely important in plant pathology because they may be used to access plant health. Figure 1 below provides a detailed description of various types of sensors. Digital cameras can easily manage the straightforward source of RGB (Red, Green, and Blue) images for identifying, quantifying, and detecting diseases. The technological requirements of simple-handled devices include a photo sensor with light sensitivity, spatial resolution, and digital and optical focus, all of which contribute to the improvement that is seen every year. The latest and most powerful digital camera-based sensors that are available in mobile phones and tablets are being used by farmers and psychopathologists in the current age. Additionally the digital image of plant is organized from the roots to the inflorescences using video cameras and scanners as an alternative way [22]. During the growing season, the RGB sensors were used to monitor the health of the plants at every resolution scale. The detector type, spatial resolution, and spectral range of multi- and hyperspectral reflectance sensors are used to categorise them. The information on the objectives is typically accessed by the multispectral sensors in a variety of broad wavebands. The primary data source for the RGB wavebands is multispectral imaging cameras. Using this technique, the fruits and crops are also screened to avoid storage disease. The thermal sensors show the infrared thermography excess plant temperature, which is connected to the plant water status [23]. Thus, many pathogens can cause an increase in the transpiration rate of leaves and causes pathogens that can be negatively

impacting on the surface layer of the leaf and enhance the circular transportation in increased water loss rate.

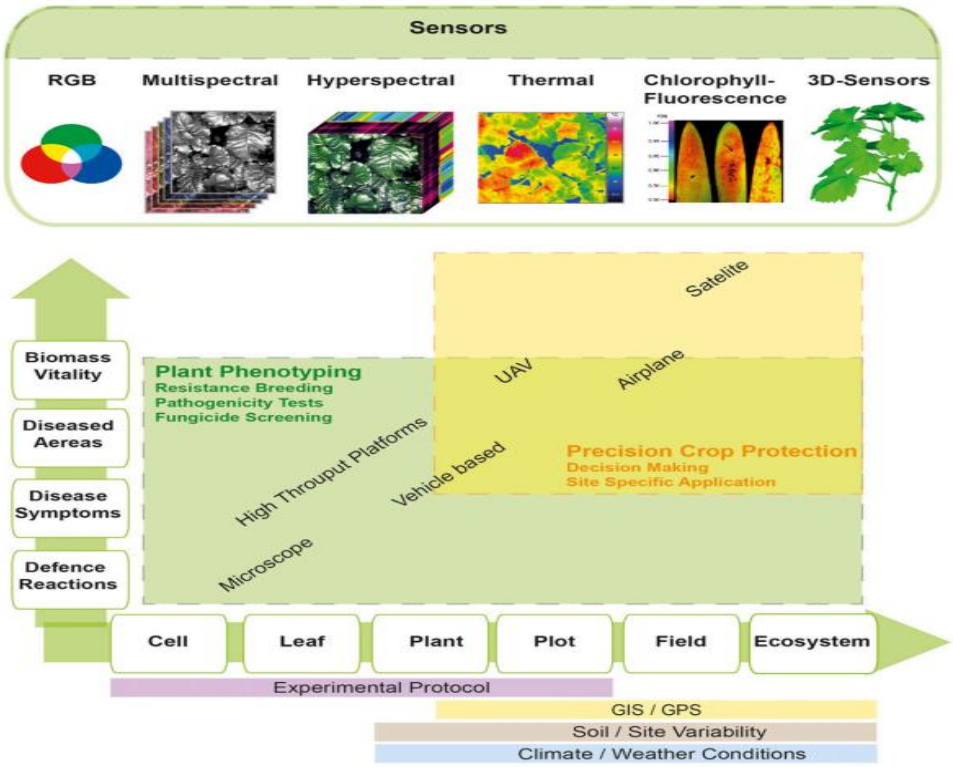


Figure 2.1. Current sensor technology used for the automated detection and identification of host plant interactions (Source: Mahlein, 2016) [24]

The variability between and within leaves can also be used to analyze Infrared thermography (IRT) images effectively. Individual leaves, crop stands, and plant interior temperatures can all be used to identify the emergence of plant diseases. A variety of chlorophyll fluorescence properties are used to evaluate changes in a plant's photosynthetic activity [25]. It provides a technique that actively makes use of LED and laser light source sensors to access photosynthetic electron transfer. It was possible to examine the behaviors brought on by biotic and abiotic pressures in the leafy areas using this method. As with the painstaking planting and difficult execution of typical agricultural greenhouse and field conditions, chlorophyll fluorescence imaging technologies have

limitations. By deriving fluorescence parameters from field-based sun-induced reflectance, it would be able to evaluate plant diseases at both the canopy and field levels. A variety of sensor technologies are accessible to plant pathologists to offer high-resolution data for a crop that is standing and can act as the basis for the early detection and identification of plant disease [26]. These technologies have been developed and put into use in agriculture and plant disease diagnostics, both of which have seen impressive advancements. Crop and plant research can now be approached in a new and concentrated way thanks to advancements in precision agriculture and plant phenotyping. The sensors used to non-intrusively analyse the nutritional status of crops in the field were the most significant and useful ones. So, using low-cost sensor technologies with acceptable market performance, future useful applications in agriculture can be created. However, Arsenovic et al. [13] claimed that no specific plant disease detection sensors are presently available on the market. The potential for sensor-based sickness detection is shown. The instruments as well as technological solutions represent the field of greenhouse and phenotyping that is available. According to Polder et al. [27] these specialised and unique prototypes weren't appropriate for widespread use. The field systems currently represent a sophisticated system that may be utilised to develop an imaging platform for identifying the tulip-breaking virus (TBV), which infected the tulip decorations from the prototype of the hyperspectral imaging platform for detecting yellow rust. This technique has enabled two advancements: the multispectral corneal robot and online machine-version analysis pipelines. These could serve as driving factors for the development expenses of a reliable optical sensor platform for the prompt and accurate detection of plant disease in crops.

Hyperspectral sensors operate on the same fundamental principles as RGB and multispectral cameras. As per Thomas *et al.*, [28] each of these gadgets records the data it collects by measuring the amount of light that reaches the sensor. A hyperspectral sensor can identify up to several hundred electromagnetic spectrum bands within its wavelength range, in contrast to RGB or multispectral cameras. The hyperspectral sensor has a high spectral resolution because each of these spectral regions only gauges the electromagnetic spectrum by a few nanometers. The two primary categories of sensors include both image sensors and non-imaging sensors [29]. Non-imaging sensors analyze the normal reflectance spectrum over a certain area of a surface without retaining spatial data. The average area is affected by several variables, including the focal length, angle of view, and proximity to the object. Most non-imaging sensors are portable and do not need elaborate measurement infrastructure. They are small and light, with a good spectral resolution (1-3 nm), and a broad spectral range (300-2500 nm). SVC, ImSpector, and ASD FieldSpec are the three most well-known spectrometers among them (Analytical Spectral Devices Inc., USA). To create a spectrum profile for each pixel, hyperspectral image sensors combine spectral and spatial resolution. With two spatial dimensions and an additional spectral dimension, the final image is a three-dimensional data array (hypercube) [30]. Depending on the type of sensors being used, there are four ways to acquire a hypercube of data, which are shown in Figure 2 below. Contrarily, due to their massive amount of data and high level of collinearity, hyperspectral images are a very challenging, emerging topic that necessitates non-trivial solutions. To successfully tackle this problem, machine learning, neural networks, and discriminant and cluster analysis techniques have been used.

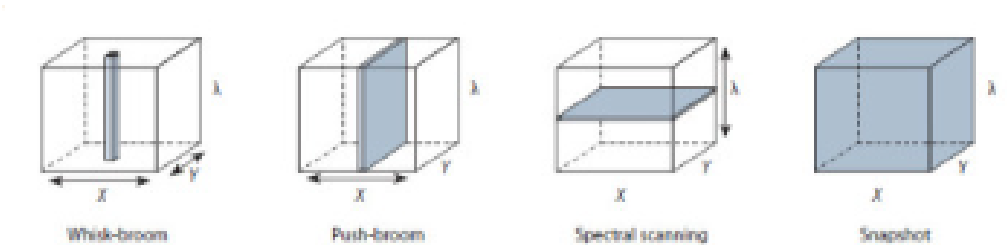


Figure 1.2 Hyperspectral images-based acquisition approach (Author: Cheshkova, 2022) [29]

Additionally, resistance screening represents the plant phenotyping that is used in numerous technological systems that have been developed [31]. The developments constituted an investigation into a single plant under carefully controlled conditions. It demonstrates how different genotypes differ in their susceptibility and resistance, and it demonstrates how this information aids in describing the specific disease that may be measured using optical sensors. Additionally, biological symptoms and presymptomatic physiological abnormalities were employed to identify plant illnesses. Depending on the plants modest deface reaction, pathogen development can be prevented during the resistance screening. The host resistance is a metaphor for a plant genotype's ability to prevent the establishment and spread of diseases [32]. Numerous genes influence complete resistance, although many genes have only minor effects on incomplete resistance. The terms polygenic effects and polygenic partial effects, respectively, refer to these two basic forms of resistance. Following the initial pathogen contact and subsequent compatible and incompatible interactions, which took into account the plant side for susceptibility or resistance of genotype, these alterations represented the distinctions in the tissues and cellular level. Because of this, the sensor-based and data-driven phenotyping approaches based on the small-scale host-pathogen interactions can be used to discriminate between barley genotypes with various susceptibilities to powdery mildew.

2.3 Issues/challenges in the image-based plant disease detection

Various techniques are used to identify the diagnosis of the disease that has been developed and are proven in the molecular biology delivery that aids in the accurate identification of the pathogenic factors. These techniques are used to analyze plant disease, and significant damage, and the development of new techniques for the accurate identification of pathogenic factors [3]. However, many farmers are not able to use the numerous methodologies used during analysis because they are expensive and require a lot of resources to be implemented. The use of cutting-edge technology is necessary for precision farming to improve the decision-making process. The best choices have reduced costs as a result of the deployment of machine learning technologies in decision-making. Additionally, the classification of issues involved the use of several technologies, such as decision trees, random forests, linear regression, K-nearest neighbours, logistic regression, support vector machines (SVM), Naive Bayesian, and clustering [12]. Because of the deep learning (DL) approaches' ground-breaking results, artificial intelligence and computer vision have advanced. These techniques, as opposed to the conventional approach, result in more accurate predictions, promoting better decision-making. DL techniques are currently being used to quickly resolve a range of complex problems due to developments in hardware. It is conceivable that for a brief period, CNN used DL frameworks as its main technology [33]. Predictions made with DL technology are based on instances that are not distributed equally across the training data. In addition, a variety of issues and applications might affect the plant disease diagnosis procedure. One problem is the way in which the data sets are set up to show the data collected at various levels. When collecting data from plants in villages, for instance, it could be difficult to explain the data uniformly because it was collected from a range of fields. The issue with data visualization

in tables and other exhibits originates from the fact that the bulk of datasets will have variable and inconsistent material.

Farmers are unable to protect the plants because they struggle to identify these illnesses and are unable to do so. One area for identifying plant diseases is biomedicine [34]. The most suitable, effective, and trustworthy technology available today for the aim of identifying illness using photos of plant leaves is image processing techniques. To save time and rapidly and precisely diagnose all plant ailments, farmers require quick and effective techniques [35]. Scientists have developed several hypotheses for estimating agricultural yields with the use of labs and equipment for quickly diagnosing plant leaf diseases. Data input from several sources is needed for automated disease identification. The use of image-based plant disease detection may result in low-quality images of plant leaves, which are considered by all the different research publications. Further, various kind of issues and challenges in detecting the disease is identified when extensive collection of data is affected by noise and background data. The input image's training and testing samples are utilized to precisely segment the data, starting with the leaves and then on to the meaningful illness that can be recognized [36]. The impact of image processing using a machine learning technique on the ability to detect disease, however, necessitates an improvement of the existing research. Diagnosing plant diseases is a critical task for the safety and security of food. The PlantVillage project was started to develop accurate image classifiers for identifying plant diseases [37].

There are thousands of labelled photos of both healthy and diseased crop plants that were taken under controlled conditions. With the aid of such a large dataset, deep learning difficulties for

developing a reliable image classifier for plant disease diagnosis have been identified. Numerous deep neural network-based anomaly detection techniques have recently been proposed in the fields of machine learning and computer vision. The deep anomaly is divided into three groups that is depending on machine learning such as supervised, unsupervised, and semi-supervised approaches that help to provide a comprehensive review of the approach.

In addition, the data sets that are difficult to explain graphically are related to additional concerns [33]. Leaders understand the content of the data sets better when solutions are presented as graphs. The graphs for some data sets, however, might need to be completely redone because they cannot be altered directly. Furthermore, the usage of deep learning technology is used to analyze plant disease detection, but the lack of data present the most difficulties [38]. Newhart et al. [39] stated that the statistical analysis of the data and the data sets may also be difficult to understand for persons without a background in mathematics or statistics by assessing some difficulties that include atmosphere, temperature, snow, and moisture. Therefore, there is no ideal approach to promote economic growth. The estimation of the K-nearest neighbour algorithm is studied using the bare minimum of climate data [40]. To collect the discriminating information, various forms of feature extraction are also carried out utilizing the IRT input photographs. Non-linear feature extraction methods make use of aspects including color, shape, and texture [41]. However, due to the lack of interpretability and transparency of the DL classifiers, the technology helps in providing a novel strategy using the random forest, CHAID, K-nearest neighbour, and Naïve Bayes for plant disease classification research. Thus, DL classifiers are usually considered to be mysterious, deep black boxes that lack any kind of justification or details regarding how they classify data and its high accuracy.

2.4 Feature extraction for disease identification

The automatic plant disease detection system receives the images of the diseased leaves as input and diagnoses the illness correctly. According to Panchal et al. [1] this system's effectiveness will depend on the feature extraction techniques used. Images of infected leaves that are input into the system are processed using image processing algorithms to extract features from the pictures. According to Sapkal and Kulkarni [42], there are two different varieties of feature extraction methods. First, the image processing methods are used to extract features from the infected leaf images input to the system. Colour, Shape, Texture, HOG, SURF and many more properties can be extracted from images using image processing methods. A Gray Level Covariance Matrix is used to obtain the texture features. The second method makes use of Alexnet's pre-trained deep learning model, which will automatically detect features from the input image. From the submitted image of the sick leaf, the Alexnet model will automatically extract the features. The pretrained Alexnet model doesn't need much time to recognize the features in the given image. Both methods apply the Backpropagation Neural Network (BPNN) algorithm to the gathered features. The issue of slow convergence affects the BPNN. However, it should be noted that in this case, deep learning hyper parameters rather than texture features support the BPNN's faster convergence. Further, the BPNN is especially used for the deep neural network that is working on error-prone projects that includes images and other speech recognition tasks. The BPNN algorithm can collect sensitive noisy data if it is not properly designed or trained. Neural networks learn from the data they are trained on, and if the training data includes sensitive or noisy information, the network may inadvertently learn to use that information in its predictions or classifications.

Diseases are identified using a deep learning methodology based on texture and colour extraction methods. As per Magsi et al. [43], the proposed approach to disease diagnosis was tested using a dataset of 1200 photos of date palm disease, and it had an overall accuracy of 89.4%. On a national and international scale, this application will benefit harvesters and other stakeholders locally and regionally. A more precise Histogram for the coloured image-based database can be created by using the image thresholding technique to remove the region of interest from the input image's background. The data from the histogram and additional values produced from the characteristics' data are then used for statistical analysis [44]. Feature extraction happens following preprocessing of the input image. Color, size, morphology, and textural characteristics are only a few of the characteristics that make up an extraction. These approaches still have poor detection performance in terms of feature extraction. Xie et al. [45] described that CNN has developed into a complete deep-learning approach in recent years. They fully exploit image big data and find the discriminative features from the original photographs themselves to do away with memory-intensive and time-consuming image processing. Due to CNNs' ability in pattern recognition, early plant leaf disease detection has become a new area of focus for smart agriculture.

Magsi et al. [43] used feature extraction to find faults in mango fruit. The sequential forward selection method was used to retrieve the most important parts of the image. An effective neural network design that considered textural information produced a recognition accuracy of 90.26%. Colour is a characteristic that visual systems value highly. The input image is transformed into the HSV colour space to retrieve the leaf's colour values. Also, Viana et al. [46] claims that this allows for the separation of the damaged region from the rest of the image. After that, the segmented image is used to determine the size values (area) of the diseased (yellowish/pale) and healthy

(green) areas of a leaf. The smoothness and roughness of the damaged portion compared to the healthy part of the leaf are perceptual characteristics that can be quantified using texture data. The segmented contaminated area of the leaf's values is then extracted for statistical analysis utilizing morphological techniques. Following the histogram segment, the system pulls each bit of colour information from an image using the built-in tool for photos of Matlab [43]. In this instance, colour information extraction differs from thresholding. Each image pixel's RGB colour values are computed during the extraction of colour characteristics, accounting for any differences in the primary or secondary colours. Each pixel's processing results in a unique determination of the RGB ratio. The texturing feature is then used in the approach. In order for texturing to work, grey scaling is required. The image is changed to greyscale prior to utilising the "grey level co-occurrence matrix technique" (GLCM) to look at the texture feature [43]. Utilizing the complete area of the selected image, the approach determines the size of the affected area. Expanding the system's already calculated colour feature, this feature increases accuracy and efficacy.

Chapter 3

Data and Processing

3.1 Data Selection

There are several well-known datasets available for plant disease detection that can provide a collection of images showcasing both diseased and healthy crops. Notable examples include the Plant Pathology Dataset, Fruit Disease Dataset, Tomato Leaf dataset, and The PlantVillage dataset. The Plant Pathology dataset, conveniently provided by Kaggle, offers a diverse range of images featuring plant leaves affected by various diseases. The dataset includes classes for diseases such as rust, scab, and multiple types of leaf spots. If the focus is specifically on diseases impacting fruits, the Fruit Disease Dataset, also available on Kaggle, provides a comprehensive resource. For those specifically interested in tomato diseases, the Tomato Leaf dataset offers a dedicated compilation of images displaying tomato leaves affected by bacterial spot, early blight, and late blight, among other ailments. Similarly, crop-specific datasets exist for other plants like wheat, rice, or soybeans, proving valuable for targeted research and analysis. These datasets serve as valuable resources, particularly when the objective is to focus on a specific crop and study its associated diseases.

To develop a computer vision algorithm for crop disease detection, we used the PlantVillage Dataset as a starting point for learning from scratch [41]. We used transfer learning with a pre-trained model that was originally trained on the ImageNet dataset. This pre-trained model was already proficient at identifying objects and features in images, so we fine-tuned it using the

PlantVillage dataset to learn the specific features of plant images and improve its ability to classify plant species and its diseases. By combining these two approaches, an effective and efficient computer vision algorithm was developed for crop disease detection.

3.1.1 The PlantVillage Dataset

The PlantVillage dataset is a large and comprehensive dataset of plant images that was created to aid in the development of computer vision algorithms for crop disease diagnosis. It was created by the PlantVillage team at Penn State University, led by Dr. David Hughes, with funding from the National Science Foundation and the Bill and Melinda Gates Foundation. PlantVillage project provides the dataset available openly and freely.

The dataset contains over 54,000 high-quality images of 26 different crop diseases and 14 plant species along with images of healthy plants, which have a spread of 38 class labels. The images were collected from multiple sources, including professional photographers, citizen scientists, and farmers. Figure 3.1 shows example of leaf images from PlantVillage dataset.

Figure 3.2. Example of leaf images from the PlantVillage Dataset representing crop-disease pair used.

The dataset includes images of both leaves and fruits from the various plant species. The plant species included in the dataset are: apple, blueberry, cherry, corn, grape, orange, peach, pepper, potato, raspberry, soybean, squash, strawberry, and tomato. The diseases included in the dataset are: apple scab, bacterial spot, cedar apple rust, common rust, early blight, late blight, leaf curl, mosaic virus, powdery mildew, septoria leaf spot, spider mites, target spot, tomato yellow leaf curl virus, and two-spotted spider mites.

The dataset is available in two formats: as a collection of individual images and as a compressed file containing all the images in a standardized format (JPEG) with a resolution (256x256 pixels). There are three different versions for the whole PlantVillage dataset.

1. Color – 54,305 images
2. Gray scale - 54,305 images
3. Leaf Segmented – 54,306 images

Figure 3.2 shows sample images from the three versions of the PlantVillage dataset.

Figure 3.2.Sample images from three different versions of the PlantVillage dataset.

The PlantVillage team has also created a website called PlantVillage.com, which provides a wealth of information on crop diseases, including diagnostic tools, treatment recommendations, and a community forum for plant enthusiasts and professionals.

3.1.2 The ImageNet Dataset

The ImageNet dataset [47] has several advantages in this research. It is one of the largest datasets currently available, which makes it a perfect source of data for creating machine learning models. By classifying the photos using the WorldNet hierarchy, the models created for identifying objects are accurate and efficient. The ImageNet dataset is a priceless tool for creating machine-learning models. This research's ideal data source contains more than 50,000 color photos of crop leaves, including both healthy and damaged plants. A high sample size, precise labelling, and the potential to identify agricultural diseases are just a few advantages of using this dataset.

The ImageNet dataset contains 14,197,122 annotated images according to WorldNet hierarchy. It has 1000 different object categories, with an average of 1000 images per category.

3.2 Data Processing

Data analysis is a crucial step in cleaning and modelling the data once it has been acquired. Extraction of pertinent data from data sources is a step in the data analysis process. To ensure that the information is correct and dependable, this procedure also aids in cleaning and deleting extraneous data from the obtained data [48]. To deliver precise findings for the completion of the work, Tensor Flow—a system for managing information and analysis—was used.

Several libraries built on the Python programming language were used to examine the data in the PlantVillage datasets. Tensor Flow, Numpy, and Keras for building Neural Network Architecture, as well as MatplotLib for plotting libraries were utilized for the analysis. These libraries all serve various purposes and are essential to the study. A crucial library for developing and deploying machine learning models is TensorFlow [49]. It offers a thorough and adaptable environment for creating and developing neural networks. It also makes computing effective and scalable, which makes it a perfect library for the study of massive datasets.

3.2.1 Tensor flow library

To verify that the information is trustworthy and correct, data analysis is essential. The data is effectively evaluated by the Tensor Flow technology. The study of PlantVillage datasets is made possible by the use of many Python-based tools, including TensorFlow, Numpy, Keras, and MatplotLib. These libraries work together to ensure that the data is examined properly and efficiently, producing reliable findings.

After Data preprocessing, dataset were trained on different models including: AlexNet, GoogLeNet, InceptionV3 and ResNet50. We perform modelling with various training-testing ratios of 90:10, 80:20, 70:30, and 60:40.

3.3 Hardware Setup and Specifications

The hardware specification for implementation of crop disease detection is as follows:

Table 3.1: Hardware Setup and Specifications

Processor	AMD Ryzen 7 5800H
Memory	16 GB
Storage	512 SSD
Operating System	Windows 10
Graphics card	NVIDIA GeForce RTX 3050ti (4 GB)
Development Environment	Jupyter Notebook

Chapter 4

Methods

4.1 Description

AI refers to the broad field of computer science focused on creating intelligent systems that can mimic human intelligence. Machine learning (ML) is a subset of AI that uses algorithms to enable computers to learn from data and make predictions or decisions without explicit programming. Deep learning (DL) is a specific type of ML that utilizes artificial neural networks with multiple layers to learn complex patterns and hierarchies in data, enabling it to excel in tasks such as image and speech recognition. In crop disease detection for image classification problem, we focus on some popular deep learning architectures namely AlexNet, GooLeNet, ResNet50 and InceptionV3. These architectures are used to determine healthy and diseased crops with their names.

4.2 Library

There are few Libraries which are required during training of deep learning models. The main libraries such as ‘TensorFlow’, ’keras’, ’matplotlib’, ‘OS’, and ‘time’ has been used. TensorFlow provides a range of tools, that we used for building and training deep neural networks. Keras provides a simple and intuitive interface that we used to build and experiment with deep learning models. Matplotlib is used to plot images. We have used OS library to use operating system functionality like reading or writing to the file system and creating new directories. Time is used

to generate a unique run ID for each run of the model. Figure 4.1 shows the code snippet to import the libraries.

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import os
import time
```

Figure 4.1 Importing Libraries

4.2.1 TensorFlow

TensorFlow is an open-source deep learning framework developed by Google that allows developers to build, train, and deploy machine learning models. With TensorFlow, we can design a graph of calculations that represent the different mathematical processes involved in the model. Overall, TensorFlow makes it easier for developers to build and experiment with deep learning models [50]. Currently, TensorFlow is used in a variety of applications, including voice search, picture recognition, and text-based ones.

4.2.2 Keras

Keras is a high-level deep learning library and API created by Google that can be used as a stand-alone framework or integrated into other deep learning frameworks like TensorFlow, Microsoft Cognitive Toolkit, or Theano. Keras provides a user-friendly and intuitive interface that allows developers to quickly build and experiment with deep learning models, without needing to have a deep understanding of the low-level details of neural networks. Keras was originally developed by François Chollet and is now a part of the TensorFlow project. It is utilized to simplify the development of neural networks using Python. Because it offers a high degree of abstraction

python frontend and the choice of many back-ends for computation, Keras is comparatively simple to understand and use. As a result, Keras is considerably more beginner-friendly yet a bit slower than other deep-learning frameworks [51].

4.2.3 Matplotlib

Matplotlib is a data visualization library that works with the popular numerical computing library NumPy in Python. It provides a cross-platform toolkit for creating high-quality graphs and charts that can be used for data analysis, scientific research, and engineering. Matplotlib is an open-source alternative to proprietary tools like MATLAB and can be used to create visualizations that are both informative and aesthetically pleasing. Developers can also use the interfaces provided by Matplotlib to include graphs in GUI applications, making it a versatile and powerful tool for data visualization [52].

4.2.4 OS

The OS module in Python provides a set of functions that allow us to interact with our operating system. It provides a variety of practical OS features that may be utilised to carry out OS-based operations and obtain OS-related data. Python's basic utility modules cover the OS [53].

4.2.5 Sklearn

Scikit-learn is Python's most practical and reliable machine-learning library. It provides powerful tools and algorithms that can be used for statistical modeling and machine learning tasks. Sklearn is designed to be easy to use and integrates well with other Python libraries such as NumPy and Pandas. It also includes useful features such as data preprocessing, model selection, and

performance evaluation tools. With the help of scikit-learn we can employ various machine learning techniques such as clustering, classification, and regression.

4.3 Parameters

4.3.1 Loss function

A loss function quantifies the error between a machine learning model's predicted output and the true output. It is a parameter that is specified when the model is constructed and helps determine the model's performance during training. The choice of loss function depends on the problem being solved and the type of data being used. We have used sparse_categorical_crossentropy given by equation 4.1.

$$- \sum_{i=1}^{N} \log t_i \log(p_i) \tag{4.1}$$

For n classes, where t_i is the truth label and p_i is the Softmax probability for the i^{th}class [54]. Sparse categorical crossentropy is a loss function used in neural network training, particularly for multiclass classification problems where the number of classes is large. It calculates the cross-entropy between the predicted probability distribution and the true probability distribution of the classes, but only considers the true class label as a single value, rather than a one-hot encoded vector. This makes it useful for cases where the number of classes is very large and the one-hot encoding of the labels would require a lot of memory. The sparse categorical crossentropy loss function encourages the model to assign a high probability to the correct class label, and penalizes the model for assigning high probabilities to incorrect labels.

4.3.2 Optimizer

In machine learning, an optimizer is an algorithm used to adjust the parameters of a model during training to minimize the loss function. The goal of the optimizer is to find the set of parameters that result in the lowest possible loss value, thus improving the model's ability to make accurate predictions.

Stochastic Gradient Descent (SGD) was used in this study, Stochastic Gradient Descent (SGD) is a popular optimization algorithm used in machine learning for training deep neural networks. It is a gradient-based optimization algorithm that updates the model parameters in small steps based on the gradients of the loss function with respect to the parameters. SGD works by randomly selecting a small batch of data from the training set and computing the gradients of the loss function with respect to the parameters using that mini-batch. The model parameters are then updated in the direction of the negative gradient, scaled by a learning rate, to minimize the loss function. This process is repeated for multiple mini-batches until the model converges to a satisfactory solution. SGD is computationally efficient and can work well for large datasets and simple models. SGD performs a parameter update for each training example $x^{(i)}$ and label $y^{(i)}$:

$$\theta = \theta - \eta \cdot \nabla_\theta J(\theta; x^{(i)}; y^{(i)}) \tag{4.2}$$

Where θ represents the current values of the model parameters that the algorithm is trying to optimize. η is the learning rate, $\nabla_\theta J(\theta; x^{(i)}; y^{(i)})$ represents the gradient of the loss function. SGD eliminates redundancy by computing the gradient and updating the model parameters for each example one at a time [55].

4.3.3 Learning rate

The learning rate determines the step size at which a model's parameters are updated during training. It is critical in determining the model's ability to converge to the optimal parameters, and selecting an appropriate learning rate is an important part of training machine learning models. During the experimentation process, various learning rates were tested for the deep learning networks. However, it was observed that some of the tested learning rates led to overfitting of the model. On the other hand, a learning rate of 0.005 consistently resulted in the best accuracy for the deep learning networks. Therefore, 0.005 was chosen as the optimal learning rate for the final model.

4.3.4 Epochs

In machine learning, an epoch is a term used to describe one complete pass through the training dataset during the training of a model. During each epoch, the model makes a prediction on each training sample and updates its parameters to minimize the loss function. The number of epochs was **30** and it is a hyperparameter that is specified before training and determines the number of times the entire dataset will be used to train the model. Different number of epochs were used to test the model and the best accuracy was achieved with 30 epochs.

4.3.5 Batch_size

The batch size is a hyperparameter that determines the number of training samples used in each iteration of the model's training during an epoch, which was **30** for GoogLeNet and ResNet50, **20** for AlexNet and InceptionV3. Different batch sizes were used to test on different architectures and the best accuracy was achieved with the given batch sizes. The batch size is used in conjunction

with the number of epochs and the learning rate to optimize the model's parameters. A larger batch size can result in faster training times and more stable updates to the model's parameters, while a smaller batch size can help the model generalize better and avoid getting stuck in local minima.

4.4 Deep Learning Architectures

Deep learning architectures are neural networks with multiple layers that can learn hierarchical representations of data. These architectures have revolutionized the field of artificial intelligence and have been successfully applied to various applications such as image recognition, speech recognition, natural language processing, and more. Some common deep learning architectures include: Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Long Short-Term Memory Networks (LSTMs), Generative Adversarial Networks (GANs), and Autoencoders. These are just a few examples of the many deep learning architectures that exist, and each has its own strengths and weaknesses depending on the specific task at hand. We have used Convolutional Neural Networks for the crop disease detection.

4.5 Convolutional Neural Networks (CNNs)

Convolutional Neural Network is a type of deep learning architecture that is primarily used for image and video analysis. A CNN consists of several layers that are designed to extract meaningful features from the input images or videos. The first layer in a CNN is a convolutional layer, which applies a set of filters to the input image to extract low-level features such as edges and corners. The subsequent layers in a CNN use these low-level features to extract higher-level features such as shapes and objects. The pooling layers downsample the feature maps obtained from the

convolutional layers, reducing the spatial dimensions of the feature maps while retaining the important information.

The output of the convolutional and pooling layers is then flattened and passed through a series of fully connected layers, which perform the classification or regression task. The unique aspect of CNNs is their ability to learn spatial hierarchies of features by performing convolutions and pooling operations. This allows them to effectively handle images of varying sizes and orientations, making them a powerful tool for tasks such as object detection, image segmentation, and more. We have used CNNs namely GoogLeNet, ResNet50, InceptionV3 and AlexNet. Figure 4.2 shows a general architecture of the Convolutional Neural Network.

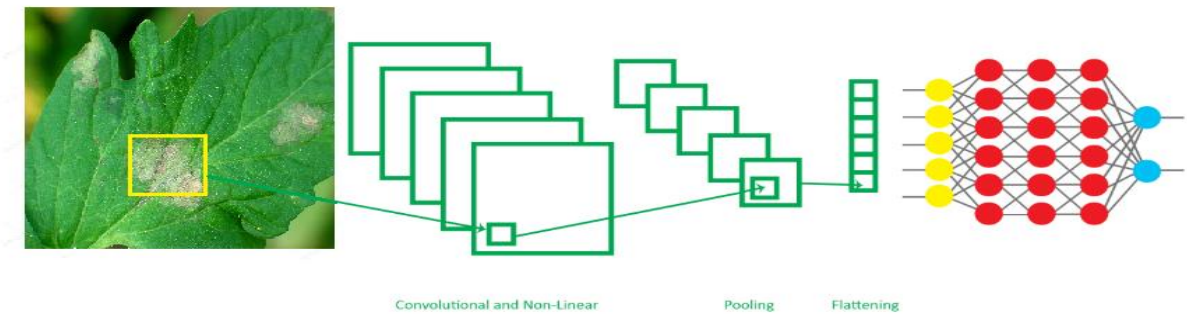


Figure 4.2: Convolutional Neural Network (Turhan, 2019) [56]

4.5.1 GoogLeNet

GoogleNet, also known as Inception-v1, is a convolutional neural network architecture developed by researchers at Google in 2014. It was designed to be a deeper and more efficient neural network for image classification tasks, with fewer parameters and lower computational complexity compared to AlexNet architectures. GoogleNet consists of 22 layers, including several inception modules that perform parallel convolutions at multiple scales. These inception modules are

designed to capture a wide range of features at different levels of abstraction, allowing the network to effectively learn complex patterns in the input images.

One of the key innovations of GoogleNet is the use of 1x1 convolutional layers to reduce the dimensionality of the feature maps before performing more computationally expensive convolutions. This helps to reduce the number of parameters in the network and improve its computational efficiency. GoogleNet achieved state-of-the-art performance on the ImageNet dataset at the time of its release, and it has since inspired numerous follow-up architectures such as Inception-v2, Inception-v3, and Inception-ResNet. To teach the network the features that are important for image classification, a sizable dataset of images is used for training. The picture data is fed into the network in the instance of crop disease detection, and the characteristics are retrieved from the images. The next step is to train a classifier with these attributes so that it can recognise the presence of a particular crop disease in fresh photos. Figure 4.3 shows the parameter settings for GoogleNet architecture.

Table 4.1: GoogLeNet architecture Parameters

Layer	**Output Shape**	**Number of Parameters**
Input layer	(None, 224, 224, 3)	0
Conv2D	(None,112, 112, 64)	9472
MaxPooling2D	(None, 56, 56, 64)	0
BatchNormalization	(None, 56, 56, 64)	256
Conv2D	(None, 56, 56, 64)	4160
Conv2D	(None, 56, 56, 192)	110784
BatchNormalization	(None, 56, 56, 192)	768
MaxPooling2D	(None, 28, 28, 192)	0
Conv2D	(None, 28, 28, 96)	18528
Conv2D	(None, 28, 28, 16)	3088
MaxPooling2D	(None, 28, 28, 192)	0
Conv2D	(None, 28, 28, 64)	12352
Conv2D	(None, 28, 28, 128)	110720

Conv2D	(None, 28, 28 , 32)	12832
Conv2D	(None, 28, 28, 32)	6176
Concatenate	(None, 28, 28, 256)	0
Conv2D	(None, 28, 28, 128)	32896
Conv2D	(None,28, 28, 32)	8224
MaxPooling2D	(None, 28, 28, 256)	0
Conv2D	(None, 28, 28, 128)	32896
Conv2D	(None, 28, 28, 192)	221376
Conv2D	(None, 28, 28, 96)	76896
Conv2D	(None, 28, 28, 64)	16448
Concatenate	(None, 28, 28, 480)	0
MaxPooling2D	(None, 14, 14, 480)	0
Conv2D	(None, 14, 14, 96)	46176
Conv2D	(None, 14, 14, 16)	7696
MaxPooling2D	(None, 14, 14, 480)	0
Conv2D	(None, 14, 14, 192)	92352
Conv2D	(None, 14, 14, 208)	179920
Conv2D	(None, 14, 14, 48)	19248
Conv2D	(None, 14, 14, 64)	30784
Concatenate	(None, 14, 14, 512)	0
Conv2D	(None, 14, 14, 112)	57456
Conv2D	(None, 14, 14, 24)	12312
MaxPooling2D	(None, 14, 14, 512)	0
Conv2D	(None, 14, 14, 160)	82080
Conv2D	(None, 14, 14, 224)	226016
Conv2D	(None, 14, 14, 64)	38464
Conv2D 27	(None, 14, 14, 64)	32832
Concatenate	(None, 14, 14, 512)	0
Conv2D 29	(None, 14, 14, 128)	65664
Conv2D 31	(None, 14, 14, 24)	12312
MaxPooling2D	(None, 14, 14, 512)	0
Conv2D 28	(None, 14, 14, 128)	65664
Conv2D 30	(None, 14, 14, 256)	295168
Conv2D 32	(None, 14, 14, 64)	38464
Conv2D 33	(None, 14, 14, 64)	32832
Concatenate 4	(None, 14, 14, 512)	0
Conv2D 35	(None, 14, 14, 144)	73872
Conv2D 37	(None, 14, 14, 32)	16416
MaxPooling2D	(None, 14, 14, 512)	0
Conv2D 34	(None, 14, 14, 112)	57456
Conv2D 36	(None, 14, 14, 288)	373536
Conv2D 38	(None, 14, 14, 64)	51264
Conv2D 39	(None, 14, 14, 64)	32832

Concatenate 5	(None, 14, 14, 528)	0
Conv2D 42	(None, 14, 14, 160)	84640
Conv2D 44	(None, 14, 14, 32)	16928
MaxPooling2D	(None, 14, 14, 528)	0
Conv2D 41	(None, 14, 14, 256)	135424
Conv2D 43	(None, 14, 14, 320)	461120
Conv2D 45	(None, 14, 14, 128)	102528
Conv2D 46	(None, 14, 14, 128)	67712
Concatenate 6	(None, 14, 14, 832)	0
MaxPooling2D	(None, 7, 7, 832)	0
Conv2D 48	(None, 7, 7, 160)	133280
Conv2D 50	(None, 7, 7, 32)	26656
MaxPooling2D	(None, 7, 7, 832)	0
Conv2D 47	(None, 7, 7, 256)	213248
Conv2D 49	(None, 7, 7, 320)	461120
Conv2D 51	(None, 7, 7, 128)	102528
Conv2D 52	(None, 7, 7, 128)	106624
Concatenate 7	(None, 7, 7, 832)	0
Conv2D 54	(None, 7, 7, 192)	159936
Conv2D 56	(None, 7, 7, 48)	39984
MaxPooling2D	(None, 7, 7, 832)	0
Conv2D 53	(None, 7, 7, 384)	319872
Conv2D 55	(None, 7, 7, 384)	663936
Conv2D 57	(None, 7, 7, 128)	153728
Conv2D 58	(None, 7, 7, 128)	106624
Concatenate 8	(None, 7, 7, 1024)	0
AveragePooling	(None, 4, 4, 512)	0
AveragePooling 1	(None, 4, 4, 528)	0
AveragePooling 2	(None, 1, 1, 1024)	0
Conv2D 21	(None, 4, 4, 128)	65664
Conv2D 40	(None, 4, 4, 128)	67712
Flatten 2	(None, 1024)	0
Flatten	(None, 2048)	0
Flatten 1	(None, 2048)	0
Dropout 2	(None, 1024)	0
Dense	(None, 256)	524544
Dense 1	(None, 256)	524544
Dense 2	(None, 256)	262400
Dropout	(None, 256)	0
Dropout 1	(None, 256)	0
Main (Dense)	(None, 38)	9766
Aux1 (Dense)	(None, 38)	9766
Aux2 (Dense)	(None, 38)	9766

Total params: 7,448,738
Trainable params:7,448,226
Non-trainable params: 512

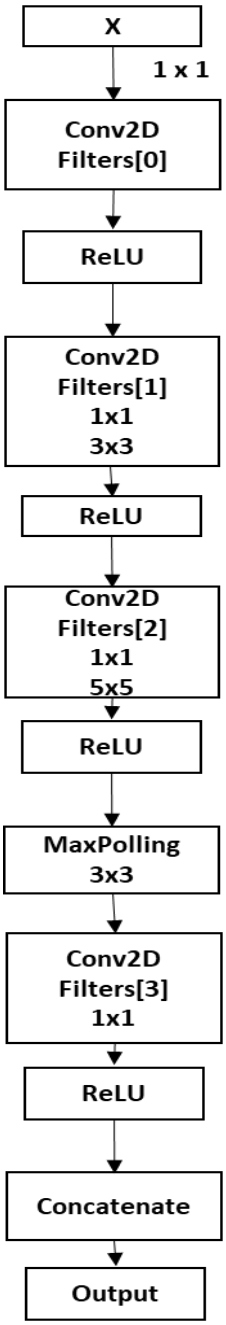


Figure 4.3: Inception block

Figure 4.4 shows an inception block in the GoogleNet architecture. An inception block consists of four paths, each of which performs a different type of convolution on the input feature maps. These four paths are then concatenated along the channel dimension and passed on to the next layer. By concatenating the output of each path along the channel dimension, the inception block is able to capture a wide range of patterns and features in the input feature maps, making it a powerful building block for deep neural networks.

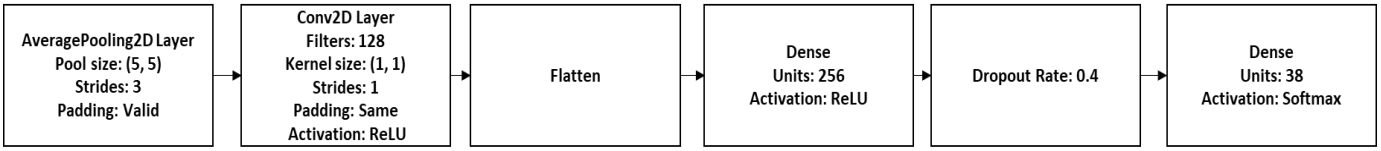


Figure 4.4: Auxiliary block

Figure 4.5 shows an auxiliary block in the GoogleNet architecture. It is an additional block that is inserted into the network at an intermediate layer, and is used to provide additional supervision and regularization during training. The complete model of GoogleNet is shown in Figure 4.6.

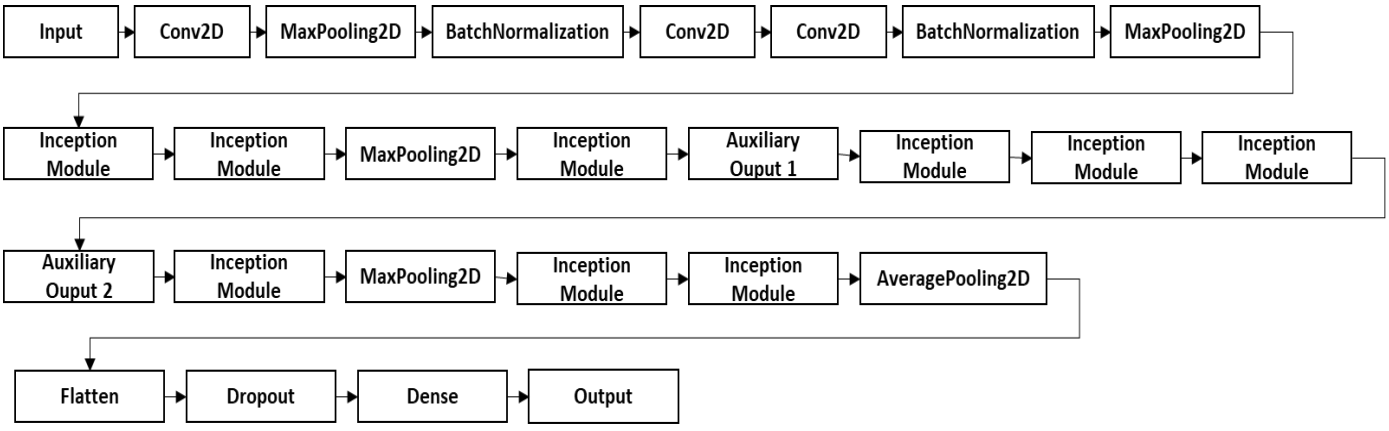


Figure 4.5: GoogLeNet model

In our implementation, we used the Keras API to build the GoogleNet model. The ‘googlenet()’ function defines the architecture of the GoogleNet model. The input to the model is an image. The

‘googlenet()’ function returns a keras model object with the specified inputs and outputs. We trained the GoogLeNet model from scratch using the Keras library in Python. The weights of the model were initialized randomly, and the model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs. No pre-trained weights were used in this study. There were 7,448,226 trainable parameters in this GoogLeNet architecture.

4.5.2 AlexNet

AlexNet is a deep convolutional neural network architecture that was developed by Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton in 2012 [57]. It consists of eight layers, including five convolutional layers and three fully connected layers, with a total of 60 million parameters. AlexNet was the winner of the ImageNet Large Scale Visual Recognition Challenge in 2012, achieving state-of-the-art results in object recognition and image classification. It was one of the first deep learning models to demonstrate the power of convolutional neural networks for computer vision tasks, and has since inspired many other architectures in the field. Figure 4.7 shows the AlexNet model architecture.

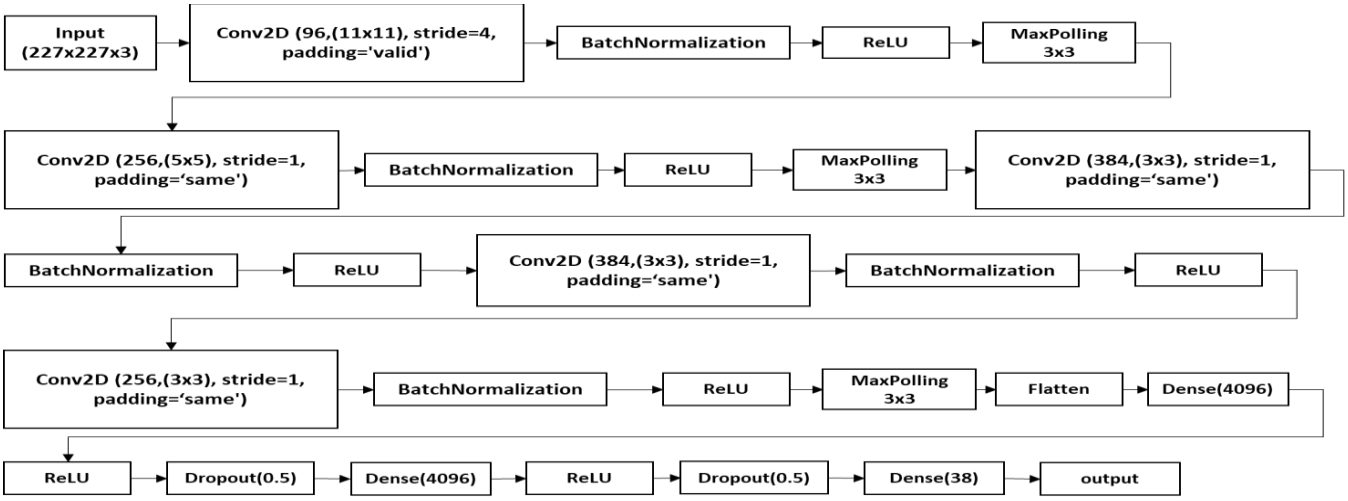


Figure 4.6: AlexNet model

Table 4.2: AlexNet architecture parameters

Layer	Output Shape	Number of Parameters
Conv2D	(None, 55, 55, 96)	34,944
BatchNormalization	(None, 55, 55, 96)	384
MaxPooling2D	(None, 27, 27, 96)	0
Conv2D	(None, 27, 27, 256)	614,656
BatchNormalization	(None, 27, 27, 256)	1,024
MaxPooling2D	(None, 13, 13, 256)	0
Conv2D	(None, 13, 13, 384)	885,120
BatchNormalization	(None, 13, 13, 384)	1,536
Conv2D	(None, 13, 13, 384)	1,327,488
BatchNormalization	(None, 13, 13, 384)	1,536
Conv2D	(None, 13, 13, 256)	884,992
BatchNormalization	(None, 13, 13, 256)	1,024
MaxPooling2D	(None, 6, 6, 256)	0
Flatten	(None, 9216)	0
Dense	(None, 4096)	37,752,832
Dropout	(None, 4096)	0
Dense	(None, 4096)	16,781,312
Dropout	(None, 4096)	0
Dense	(None, 38)	155,686
Total		58,442,534
Trainable parameters		58,439,782
Non-trainable parameters		2,752

Figure 4.8 shows the parameter setting for AlexNet. AlexNet is a convolutional neural network architecture using the Sequential API in Keras. It consists of multiple layers including Conv2D (convolutional), BatchNormalization (normalization of layer inputs), MaxPool2D (max pooling), and Dense (fully connected) layers. The model uses the ReLU activation function for convolutional and dense layers, and softmax activation for the output layer. We have 38 class labels in our dataset. The model is designed for multi-class classification, with 38 output classes.

It uses convolutional layers to extract features from the input image and fully connected layers to classify the image into one of the 38 classes. The model uses the ReLU activation function for convolutional layers. We trained the AlexNet model from scratch using the Keras library in Python. The weights of the model were initialized randomly, and the model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs. There were 58,439,782 trainable parameters in this architecture.

4.5.3 ResNet50

ResNet50 is a convolutional neural network architecture that was introduced in 2015 by researchers at Microsoft Research Asia [58]. The name "ResNet" comes from "residual network," which refers to the use of residual connections to overcome the degradation problem that can occur in deep neural networks. ResNet50 has been used for a variety of computer vision tasks, such as object detection, image classification, and image segmentation. Its architecture includes residual blocks, which allow for more efficient training and deeper network architectures. We trained ResNet50 in two way, first Learning from scratch and second Transfer learning. Figure 4.9 shows the parameter settings for ResNet50 architecture.

→ **ResNet50 Learning From Scratch:**

Table 4.3: ResNet50 architecture parameters

Layer	Output Shape	Number of Parameters
InputLayer	(None, 224, 224, 3)	0
Conv2D	(None, 112, 112, 64)	9472
BatchNormalization	(None, 112, 112, 64)	256
ReLu	(None, 112, 112, 64)	0
MaxPooling2D	(None, 56, 56, 64)	0

Conv2D 1	(None, 56, 56, 64)	36928
BatchNormalization 1	(None, 56, 56, 64)	256
ReLu 1	(None, 56, 56, 64)	0
Conv2D 2	(None, 56, 56, 64)	36928
Conv2D 3	(None, 56, 56, 64)	4160
BatchNormalization 2	(None, 56, 56, 64)	256
BatchNormalization 3	(None, 56, 56, 64)	256
Add	(None, 56, 56, 64)	0
ReLu 2	(None, 56, 56, 64)	0
Conv2D 4	(None, 56, 56, 128)	73856
BatchNormalization 4	(None, 56, 56, 128)	512
ReLu 3	(None, 56, 56, 128)	0
Conv2D 5	(None, 56, 56, 128)	147584
Conv2D 6	(None, 56, 56, 128)	8320
BatchNormalization 5	(None, 56, 56, 128)	512
BatchNormalization 6	(None, 56, 56, 128)	512
Add 1	(None, 56, 56, 128)	0
ReLu 4	(None, 56, 56, 128)	0
Conv2D 7	(None, 56, 56, 256)	295169
BatchNormalization 7	(None, 56, 56, 256)	1024
ReLu 5	(None, 56, 56, 256)	0
Conv2D 8	(None, 56, 56, 256)	590080
Conv2D 9	(None, 56, 56, 256)	33024
BatchNormalization 8	(None, 56, 56, 256)	1024
BatchNormalization 9	(None, 56, 56, 256)	1024
Add 2	(None, 56, 56, 256)	0
ReLu 6	(None, 56, 56, 256)	0
GlobalAveragePooling	(None, 256)	0
Dense	(None, 38)	9766
Total params: 1,250,918 Trainable param:1,248,102 Non trainable para:2,816		

We used two functions in ResNet50, ‘residual_block’ and ‘ResNet50’. Figure 4.10 shows the ResNet50 model architecture.

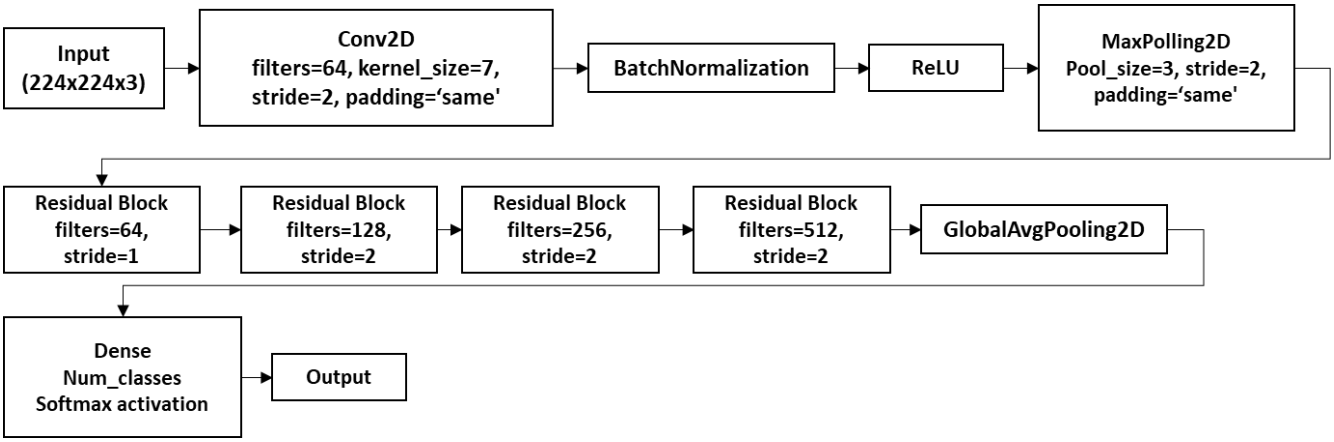


Figure 4.7: ResNet50 model

The ‘residual_block’ function takes an input tensor, the number of filters to use, and an optional stride value. It applies two convolutional layers, each with a 3x3 kernel size and 'same' padding, followed by batch normalization and ReLU activation. The residual connection is added by applying a 1x1 convolutional layer to the input tensor with the same number of filters and stride value, followed by batch normalization. The output of the convolutional layers is added to the residual connection, and then passed through a ReLU activation.

The ’ResNet50’ function takes an input shape and the number of classes to predict, and defines a ResNet50 architecture using the ‘residual_block’ function. It applies a convolutional layer with 64 filters, a 7x7 kernel size, and 'same' padding, followed by batch normalization and ReLU activation. It then applies max pooling with a pool size of 3x3 and stride of 2. The ResNet blocks are applied with increasing filter sizes of 64, 128, 256, and 512. The output of the last ResNet block is passed through global average pooling, and then a fully connected layer with softmax activation to predict the class probabilities. Overall, it is a ResNet50 neural network architecture

that can be used for crop disease detection image classification tasks. We trained the ResNet50 model from scratch. The model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs. There were 1,248,102 trainable parameters in this ResNet50 architecture.

→ **ResNet50 Transfer Learning:**

ResNet50 is a deep neural network with 50 layers, and it has been pre-trained on the ImageNet dataset, which consists of over 1 million labeled images in 1,000 categories. This pre-training enables ResNet50 to recognize a wide range of objects and features in images. Figure 4.11 shows the parameter settings for ResNet50 Transfer Learning.

Table 4.4: ResNet50 architecture for Transfer Learning Parameters

Layer	Output Shape	Number of Parameters
InputLayer	(None, 224, 224, 3)	0
ZeroPadding2D	(None, 230, 230, 3)	0
Conv2D	(None, 112, 112, 64)	9472
BatchNormalization	(None, 112, 112, 64)	256
Activation	(None, 112, 112, 64)	0
ZeroPadding2D	(None, 114, 114, 64)	0
MaxPooling2D	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 64)	4160
BatchNormalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 64)	036928
BatchNormalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 256)	16640
Conv2D	(None, 56, 56, 256)	16640
BatchNormalization	(None, 56, 56, 256)	1024
BatchNormalization	(None, 56, 56, 256)	1024

Add	(None, 56, 56, 256)	0
Activation	(None, 56, 56, 256)	0
Conv2D	(None, 56, 56, 64)	16448
Batch Normalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 64)	36928
BatchNormalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 256)	16640
BatchNormalization	(None, 56, 56, 256)	1024
Add	(None, 56, 56, 256)	0
Activation	(None, 56, 56, 256)	0
Conv2D	(None, 56, 56, 64)	16448
BatchNormalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 64)	36928
BatchNormalization	(None, 56, 56, 64)	256
Activation	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 256)	16640
BatchNormalization	(None, 56, 56, 256)	1024
Add	(None, 56, 56, 256)	0
Activation	(None, 56, 56, 256)	0
Conv2D	(None, 28, 28, 128)	32896
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 128)	147584
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 512)	131584
Conv2D	(None, 28, 28, 512)	66048
BatchNormalization	(None, 28, 28, 512)	2048

BatchNormalization	(None, 28, 28, 512)	2048
Add	(None, 28, 28, 512)	0
Activation	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 128)	65664
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 128)	147584
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 512)	66048
BatchNormalization	(None, 28, 28, 512)	2048
Add	(None, 28, 28, 512)	0
Activation	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 128)	65664
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 128)	147584
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 512)	66048
BatchNormalization	(None, 28, 28, 512)	2048
Add	(None, 28, 28, 512)	0
Activation	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 128)	65664
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 128)	147584
BatchNormalization	(None, 28, 28, 128)	512
Activation	(None, 28, 28, 128)	0
Conv2D	(None, 28, 28, 512)	66048

BatchNormalization	(None, 28, 28, 512)	2048
Add	(None, 28, 28, 512)	0
Activation	(None, 28, 28, 512)	0
Conv2D	(None, 14, 14, 256)	131328
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	525312
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024

Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0

Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 14, 14, 256)	262400
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 256)	590080
BatchNormalization	(None, 14, 14, 256)	1024
Activation	(None, 14, 14, 256)	0
Conv2D	(None, 14, 14, 1024)	263168
BatchNormalization	(None, 14, 14, 1024)	4096
Add	(None, 14, 14, 1024)	0
Activation	(None, 14, 14, 1024)	0
Conv2D	(None, 7, 7, 512)	524800
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 512)	2359808
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 2048)	2099200
Conv2D	(None, 7, 7, 2048)	1050624
BatchNormalization	(None, 7, 7, 2048)	8192
BatchNormalization	(None, 7, 7, 2048)	8192
Add	(None, 7, 7, 2048)	0
Activation	(None, 7, 7, 2048)	0
Conv2D	(None, 7, 7, 512)	1049088
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 512)	2359808
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 2048)	1050624

BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 2048)	1050624
BatchNormalization	(None, 7, 7, 2048)	8192
Add	(None, 7, 7, 2048)	0
Activation	(None, 7, 7, 2048)	0
Conv2D	(None, 7, 7, 512)	1049088
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 512)	2359808
BatchNormalization	(None, 7, 7, 512)	2048
Activation	(None, 7, 7, 512)	0
Conv2D	(None, 7, 7, 2048)	1050624
BatchNormalization	(None, 7, 7, 2048)	8192
Add	(None, 7, 7, 2048)	0
Activation	(None, 7, 7, 2048)	0
GlobalAveragePooling	(None, 2048)	0
Dense	(None, 38)	77862
Total param: 23,665,574 Trainable param: 77,862 Non trainable param: 23, 587, 712		

It is a pre-trained ResNet50 model from the Keras applications module, sets the weights to be those pre-trained on the ImageNet dataset, and freezes all layers in the base model to prevent them from being updated during training.

The ‘include_top’ parameter is set to False, which means that the fully connected layer at the top of the network, which is responsible for classifying the input image into one of 1000 classes in the

original ImageNet dataset, is not included. Instead, a new dense layer with softmax activation is added on top of the base model to classify the input images into one of 38 classes in a new task. The ‘GlobalAveragePooling2D’ layer is used to reduce the spatial dimensions of the output of the base model to a 1D vector, which is then passed through the new dense layer with softmax activation to predict the class probabilities.

Overall, it is the starting point for transfer learning, where the pre-trained ResNet50 model is fine-tuned on a plantvillage dataset for crop disease detection. The new dense layer added on top of the base model is trained on our dataset to adapt the ResNet50 model to detect crop diseases. We used transfer learning to train the ResNet50 model. The model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs. There were 77,862 trainable parameters in this ResNet50 architecture.

4.5.4 InceptionV3

InceptionV3 is a deep learning model commonly used for image classification tasks. It includes inception modules, which use convolutional layers with different kernel sizes to capture information at multiple scales and resolutions. InceptionV3 has 48 layers and it uses factorized convolutional layers to reduce the number of parameters and improve computational efficiency.

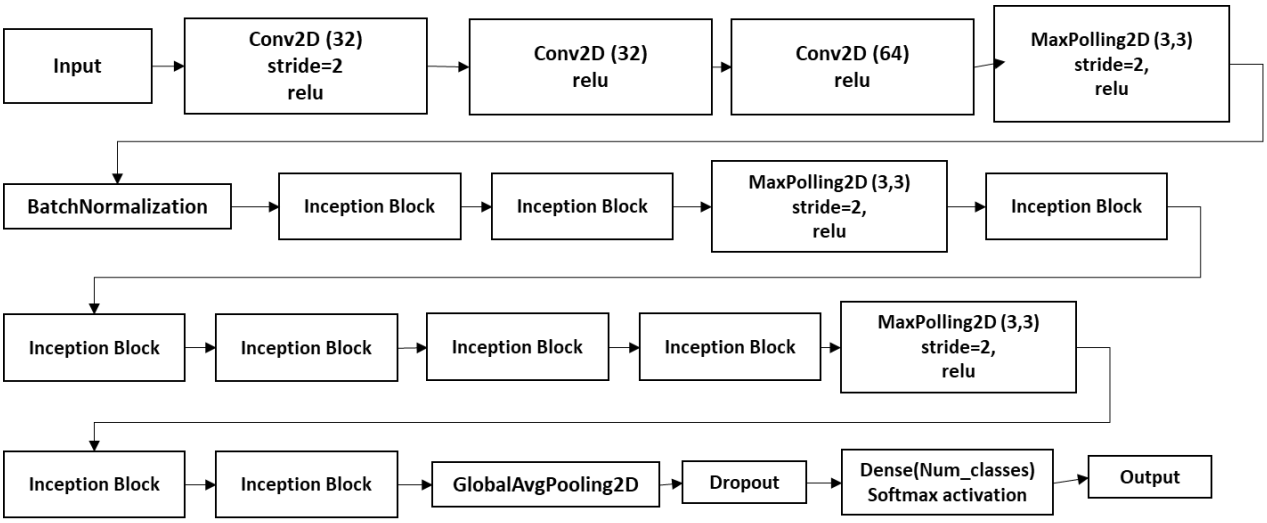


Figure 4.8: InceptionV3 model

Figure 4.12 shows the InceptionV3 model and Figure 4.13 shows the parameter settings for Inceptionv3 architecture.

→ **InceptionV3 Learning From Scratch:**

Table 4.5: InceptionV3 architecture parameters

Layer	Output Shape	Number of Parameters
InputLayer	(None, 224, 224, 3)	0
Conv2D	(None, 112, 112, 32)	896896
Conv2D	(None, 112, 112, 32)	9248
Conv2D	(None, 112, 112, 64)	18496
MaxPooling2D	(None, 56, 56, 64)	0
BatchNormalization	(None, 56, 56, 64)	256
Conv2D	(None, 56, 56, 96)	6240
Conv2D	(None, 56, 56, 16)	1040
MaxPooling2D	(None, 56, 56, 64)	0
Conv2D	(None, 56, 56, 64)	4160
Conv2D	(None, 56, 56, 128)	110720
Conv2D	(None, 56, 56, 32)	12832
Conv2D	(None, 56, 56, 32)	2080

Concatenate	(None, 56, 56, 256)	0
Conv2D	(None, 56, 56, 128)	32896
Conv2D	(None, 56, 56, 32)	8224
MaxPooling2D	(None, 56, 56, 256)	0
Conv2D	(None, 56, 56, 128)	32896
Conv2D	(None, 56, 56, 192)	221376
Conv2D	(None, 56, 56, 96)	76896
Conv2D	(None, 56, 56, 64)	16448
Concatenate	(None, 56, 56, 480)	0
MaxPooling2D	(None, 28, 28, 480)	0
Conv2D	(None, 28, 28, 96)	46176
Conv2D	(None, 28, 28, 16)	7696
MaxPooling2D	(None, 28, 28, 480)	0
Conv2D	(None, 28, 28, 192)	92352
Conv2D	(None, 28, 28, 208)	179920
Conv2D	(None, 28, 28, 48)	19248
Conv2D	(None, 28, 28, 64)	30784
Concatenate	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 112)	57456
Conv2D	(None, 28, 28, 24)	12312
MaxPooling2D	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 160)	82080
Conv2D	(None, 28, 28, 224)	226016
Conv2D	(None, 28, 28, 64)	38464
Conv2D	(None, 28, 28, 64)	32832
Concatenate	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 128)	65664
Conv2D	(None, 28, 28, 24)	12312
MaxPooling2D	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 128)	65664
Conv2D	(None, 28, 28, 256)	295168
Conv2D	(None, 28, 28, 64)	38464

Conv2D	(None, 28, 28, 64)	32832
Concatenate	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 144)	73872
Conv2D	(None, 28, 28, 32)	16416
MaxPooling2D	(None, 28, 28, 512)	0
Conv2D	(None, 28, 28, 112)	57456
Conv2D	(None, 28, 28, 228)	373536
Conv2D	(None, 28, 28, 64)	51264
Conv2D	(None, 28, 28, 64)	32832
Concatenate	(None, 28, 28, 528)	0
Conv2D	(None, 28, 28, 160)	84640
Conv2D	(None, 28, 28, 32)	16928
MaxPooling2D	(None, 28, 28, 528)	0
Conv2D	(None, 28, 28, 256)	135424
Conv2D	(None, 28, 28, 320)	461120
Conv2D	(None, 28, 28, 128)	102528
Conv2D	(None, 28, 28, 128)	67712
Concatenate	(None, 28, 28, 832)	0
MaxPooling2D	(None, 14, 14, 832)	0
Conv2D	(None, 14, 14, 160)	133280
Conv2D	(None, 14, 14, 32)	26656
MaxPooling2D	(None, 14, 14, 832)	0
Conv2D	(None, 14, 14, 256)	213248
Conv2D	(None, 14, 14, 320)	461120
Conv2D	(None, 14, 14, 128)	102528
Conv2D	(None, 14, 14, 128)	106624
Concatenate	(None, 14, 14, 832)	0
Conv2D	(None, 14, 14, 192)	159936
Conv2D	(None, 14, 14, 48)	39984
MaxPooling2D	(None, 14, 14, 832)	0

Conv2D	(None, 14, 14, 384)	319872
Conv2D	(None, 14, 14, 384)	663936
Conv2D	(None, 14, 14, 128)	153728
Conv2D	(None, 14, 14, 128)	106624
Concatenate	(None, 14, 14, 1024)	
GlobalAveragePooling	(None, 1024)	
Dropout	(None, 1024)	
Dense	(None, 38)	
Total param: 5,890,358 Trainable params: 5,890,230 Non trainable params: 128		

Figure 4.13 shows an implementation of the InceptionV3 model using TensorFlow and Keras. The model consists of a stem, several inception blocks, and a classifier. The stem includes several convolutional layers and a max pooling layer to reduce the spatial dimensions of the input image. The inception blocks use convolutional layers with different kernel sizes to capture information at different scales and resolutions. Each block includes four parallel convolutional branches that are concatenated along the channel dimension. The classifier includes a global average pooling layer, a dropout layer for regularization, and a fully connected layer with softmax activation for classification. The 'inception_block' function is used to define the structure of each inception block. We trained the InceptionV3 model from scratch. The model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs. There were 5,890,230 trainable parameters in this InceptionV3 architecture. Figure 4.14 shows the parameter settings for the InceptionV3 Transfer Learning model.

→ **InceptionV3 Transfer Learning:**

Table 4.6: InceptionV3 architecture transfer learning parameters

Layer	Output Shape	Number of Parameters
InputLayer	(None, None, None,3)	0
Conv2D	(None, None, None,32)	864
BatchNormalization	(None, None, None,32)	96
Activation	(None, None, None,32)	0
Conv2D	(None, None, None,32)	9216
BatchNormalization 1	(None, None, None,32)	96
Activation1	(None, None, None,32)	0
Conv2D_2	(None, None, None,64)	18432
BatchNormalization2	(None, None, None,64)	192
Activation2	(None, None, None,64)	0
MaxPooling2D	(None, None, None,64)	0
Conv2D_3	(None, None, None,80)	5120
BatchNormalization3	(None, None, None,80)	240
Activation3	(None, None, None,80)	0
Conv2D_4	(None, None, None,192)	138240
BatchNormalization4	(None, None, None,192)	576
Activation_4	(None, None, None,192)	0
MaxPooling2D_1	(None, None, None,192)	0
Conv2D_8	(None, None, None,64)	12288
BatchNormalization8	(None, None, None,64)	192
Activation_8	(None, None, None,64)	0

Conv2D_6	(None, None, None,48)	9216
Conv2D_9	(None, None, None,96)	55296
BatchNormalization_6	(None, None, None,48)	144
BatchNormalization_9	(None, None, None,96)	288
Activation_6	(None, None, None,48)	0
Activation_9	(None, None, None,96)	0
AveragePooling	(None, None, None,192)	0
Conv2D_5	(None, None, None,64)	12288
Conv2D_7	(None, None, None,64)	76800
Conv2D_10	(None, None, None,96)	82944
Conv2D_11	(None, None, None,32)	6144
BatchNormalization_5	(None, None, None,64)	192
BatchNormalization_7	(None, None, None,64)	192
BatchNormalization_10	(None, None, None,96)	288
BatchNormalization_11	(None, None, None,32)	96
Activation_5	(None, None, None,64)	0
Activation_7	(None, None, None,64)	0
Activation_10	(None, None, None,96)	0
Activation_11	(None, None, None,32)	0
Concatenate	(None, None, None,256)	0
Conv2D_15	(None, None, None,64)	16384
BatchNormalization_15	(None, None, None,64)	192
Activation_15	(None, None, None,64)	0

Conv2D_13	(None, None, None,48)	12288
Conv2D_16	(None, None, None,96)	55296
BatchNormalization_13	(None, None, None,48)	144
BatchNormalization_16	(None, None, None,96)	288
Activation_13	(None, None, None,48)	0
Activation_16	(None, None, None,96)	0
AveragePooling_1	(None, None, None,256)	0
Conv2D_12	(None, None, None,64)	16384
Conv2D_14	(None, None, None,64)	76800
Conv2D_17	(None, None, None,96)	82944
Conv2D_18	(None, None, None,64)	16384
BatchNormalization_12	(None, None, None,64)	192
BatchNormalization_14	(None, None, None,64)	192
BatchNormalization_17	(None, None, None,96)	288
BatchNormalization_18	(None, None, None,64)	192
Activation_12	(None, None, None,64)	0
Activation_14	(None, None, None,64)	0
Activation_17	(None, None, None,96)	0
Activation_18	(None, None, None,64)	0
Concatenate	(None, None, None,288)	0
Conv2D_22	(None, None, None,64)	18432
BatchNormalization_22	(None, None, None,64)	192
Activation_22	(None, None, None,64)	0

Conv2D_20	(None, None, None,48)	13824
Conv2D_23	(None, None, None,96)	55296
BatchNormalization_20	(None, None, None,48)	144
BatchNormalization_23	(None, None, None,96)	288
Activation_20	(None, None, None,48)	0
Activation_23	(None, None, None,96)	0
AveragePooling_2	(None, None, None,288)	0
Conv2D_19	(None, None, None,64)	18432
Conv2D_21	(None, None, None,64)	76800
Conv2D_24	(None, None, None,96)	82944
Conv2D_25	(None, None, None,64)	18432
BatchNormalization_19	(None, None, None,64)	192
BatchNormalization_21	(None, None, None,64)	192
BatchNormalization_24	(None, None, None,96)	288
BatchNormalization_25	(None, None, None,64)	192
Activation_19	(None, None, None,64)	0
Activation_21	(None, None, None,64)	0
Activation_24	(None, None, None,96)	0
Activation_25	(None, None, None,64)	0
Concatenate	(None, None, None,288)	0
Conv2D_27	(None, None, None,64)	18432
BatchNormalization_27	(None, None, None,64)	192
Activation_27	(None, None, None,64)	0

Conv2D_28	(None, None, None,96)	55296
BatchNormalization_28	(None, None, None,96)	288
Activation_28	(None, None, None,96)	0
Conv2D_26	(None, None, None,384)	995328
Conv2D_29	(None, None, None,96)	82944
BatchNormalization_26	(None, None, None,384)	1152
BatchNormalization_29	(None, None, None,96)	288
Activation_26	(None, None, None,384)	0
Activation_29	(None, None, None,96)	0
MaxPooling2D_2	(None, None, None,288)	0
Concatenate	(None, None, None,768)	0
Conv2D_34	(None, None, None,128)	98304
BatchNormalization_34	(None, None, None,128)	384
Activation_34	(None, None, None,128)	0
Conv2D_35	(None, None, None,128)	114688
BatchNormalization_35	(None, None, None,128)	384
Activation_35	(None, None, None,128)	0
Conv2D_31	(None, None, None,128)	98304
Conv2D_36	(None, None, None,128)	114688
BatchNormalization_31	(None, None, None,128)	384
BatchNormalization_36	(None, None, None,128)	384
Activation_31	(None, None, None,128)	0
Activation_36	(None, None, None,128)	0

Conv2D_32	(None, None, None,128)	114688
Conv2D_37	(None, None, None,128)	114688
BatchNormalization_32	(None, None, None,128)	384
BatchNormalization_37	(None, None, None,128)	384
Activation_32	(None, None, None,128)	0
Activation_37	(None, None, None,128)	0
AveragePooling_3	(None, None, None,768)	0
Conv2D_30	(None, None, None,192)	147456
Conv2D_33	(None, None, None,192)	172032
Conv2D_38	(None, None, None,192)	172032
Conv2D_39	(None, None, None,192)	147456
BatchNormalization_30	(None, None, None,192)	576
BatchNormalization_33	(None, None, None,192)	576
BatchNormalization_38	(None, None, None,192)	576
BatchNormalization_39	(None, None, None,192)	576
Activation_30	(None, None, None,192)	0
Activation_33	(None, None, None,192)	0
Activation_38	(None, None, None,192)	0
Activation_39	(None, None, None,768)	0
Concatenate	(None, None, None,160)	0
Conv2D_44	(None, None, None,160)	122880
BatchNormalization_44	(None, None, None,160)	480
Activation_44	(None, None, None,160)	0

Conv2D_45	(None, None, None,160)	179200
BatchNormalization_45	(None, None, None,160)	480
Activation_45	(None, None, None,160)	0
Conv2D_41	(None, None, None,160)	122880
Conv2D_46	(None, None, None,160)	179200
BatchNormalization_41	(None, None, None,160)	480
BatchNormalization_46	(None, None, None,160)	480
Activation_41	(None, None, None,160)	0
Activation_46	(None, None, None,160)	0
Conv2D_42	(None, None, None,160)	179200
Conv2D_47	(None, None, None,160)	179200
BatchNormalization_42	(None, None, None,160)	480
BatchNormalization_47	(None, None, None,160)	480
Activation_42	(None, None, None,768)	0
Activation_47	(None, None, None,192)	0
AveragePooling_4	(None, None, None,192)	0
Conv2D_40	(None, None, None,192)	147456
Conv2D_43	(None, None, None,192)	215040
Conv2D_48	(None, None, None,192)	215040
Conv2D_49	(None, None, None,192)	147456
BatchNormalization_40	(None, None, None,192)	576
BatchNormalization_43	(None, None, None,192)	576
BatchNormalization_48	(None, None, None,192)	576

BatchNormalization_49	(None, None, None,192)	576
Activation_40	(None, None, None,192)	0
Activation_43	(None, None, None,768)	0
Activation_48	(None, None, None,160)	0
Activation_49	(None, None, None,160)	0
Concatenate	(None, None, None,160)	0
Conv2D_54	(None, None, None,160)	122880
BatchNormalization_54	(None, None, None,160)	480
Activation_54	(None, None, None,160)	0
Conv2D_55	(None, None, None,160)	179200
BatchNormalization_55	(None, None, None,160)	480
Activation_55	(None, None, None,160)	0
Conv2D_51	(None, None, None,160)	122880
Conv2D_56	(None, None, None,160)	179200
BatchNormalization_51	(None, None, None,160)	480
BatchNormalization_56	(None, None, None,160)	480
Activation_51	(None, None, None,160)	0
Activation_56	(None, None, None,160)	0
Conv2D_52	(None, None, None,160)	179200
Conv2D_57	(None, None, None,160)	179200
BatchNormalization_52	(None, None, None,786)	480
BatchNormalization_57	(None, None, None,192)	480
Activation_52	(None, None, None,192)	0

Activation_57	(None, None, None,192)	0
AveragePooling_5	(None, None, None,192)	0
Conv2D_50	(None, None, None,192)	147456
Conv2D_53	(None, None, None,192)	215040
Conv2D_58	(None, None, None,192)	215040
Conv2D_59	(None, None, None,192)	147456
BatchNormalization_50	(None, None, None,192)	576
BatchNormalization_53	(None, None, None,192)	576
BatchNormalization_58	(None, None, None,768)	576
BatchNormalization_59	(None, None, None,192)	576
Activation_50	(None, None, None,192)	0
Activation_53	(None, None, None,192)	0
Activation_58	(None, None, None,192)	0
Activation_59	(None, None, None,192)	0
Concatenate	(None, None, None,192)	0
Conv2D_64	(None, None, None,192)	147456
BatchNormalization_64	(None, None, None,192)	576
Activation_64	(None, None, None,192)	0
Conv2D_65	(None, None, None,192)	258048
BatchNormalization_65	(None, None, None,192)	576
Activation_65	(None, None, None,192)	0
Conv2D_61	(None, None, None,192)	147456
Conv2D_66	(None, None, None,192)	258048

BatchNormalization_61	(None, None, None,192)	576
BatchNormalization_66	(None, None, None,192)	576
Activation_61	(None, None, None,192)	0
Activation_66	(None, None, None,768)	0
Conv2D_62	(None, None, None,192)	258048
Conv2D_67	(None, None, None,192)	258048
BatchNormalization_62	(None, None, None,192)	576
BatchNormalization_67	(None, None, None,192)	576
Activation_62	(None, None, None,192)	0
Activation_67	(None, None, None,192)	0
AveragePooling_6	(None, None, None,192)	0
Conv2D_60	(None, None, None,192)	147456
Conv2D_63	(None, None, None,192)	258048
Conv2D_68	(None, None, None,192)	258048
Conv2D_69	(None, None, None,192)	147456
BatchNormalization_60	(None, None, None,768)	576
BatchNormalization_63	(None, None, None,192)	576
BatchNormalization_68	(None, None, None,192)	576
BatchNormalization_69	(None, None, None,192)	576
Activation_60	(None, None, None,192)	0
Activation_63	(None, None, None,192)	0
Activation_68	(None, None, None,192)	0
Activation_69	(None, None, None,192)	0

Concatenate	(None, None, None,192)	0
Conv2D_72	(None, None, None,192)	147456
BatchNormalization_72	(None, None, None,192)	576
Activation_72	(None, None, None,192)	0
Conv2D_73	(None, None, None,320)	258048
BatchNormalization_73	(None, None, None,192)	576
Activation_73	(None, None, None,320)	0
Conv2D_70	(None, None, None,192)	147456
Conv2D_74	(None, None, None,320)	2580487
BatchNormalization_70	(None, None, None,192)	576
BatchNormalization_74	(None, None, None,768)	576
Activation_70	(None, None, None,1280)	0
Activation_74	(None, None, None,448)	0
Conv2D_71	(None, None, None,448)	552960
Conv2D_75	(None, None, None,448)	331776
BatchNormalization_71	(None, None, None,384)	960
BatchNormalization_75	(None, None, None,384)	576
Activation_71	(None, None, None,384)	0
Activation_75	(None, None, None,384)	0
MaxPooling2D_3	(None, None, None,384)	0
Concatenate	(None, None, None,384)	0
Conv2D_80	(None, None, None,384)	573440
BatchNormalization_80	(None, None, None,384)	1344

Activation_80	(None, None, None,384)	0
Conv2D_77	(None, None, None,1280)	491520
Conv2D_81	(None, None, None,320)	1548288
BatchNormalization_77	(None, None, None,384)	1152
BatchNormalization_81	(None, None, None,384)	1152
Activation_77	(None, None, None,384)	0
Activation_81	(None, None, None,384)	0
Conv2D_78	(None, None, None,192)	442368
Conv2D_79	(None, None, None,320)	442368
Conv2D_82	(None, None, None,384)	442368
Conv2D_83	(None, None, None,384)	442368
AveragePooling_7	(None, None, None,384)	0
Conv2D_76	(None, None, None,384)	409600
BatchNormalization_78	(None, None, None,192)	1152
BatchNormalization_79	(None, None, None,320)	1152
BatchNormalization_82	(None, None, None,768)	1152
BatchNormalization_83	(None, None, None,768)	1152
Conv2D_84	(None, None, None,192)	245760
BatchNormalization_76	(None, None, None,2048)	960
Activation_78	(None, None, None,448)	0
Activation_79	(None, None, None,448)	0
Activation_82	(None, None, None,448)	0
Activation_83	(None, None, None,384)	0

BatchNormalization_84	(None, None, None,384)	576
Activation_76	(None, None, None,384)	0
Concatenate	(None, None, None,384)	0
Concatenate	(None, None, None,384)	0
Activation_84	(None, None, None,384)	0
Concatenate	(None, None, None,384)	0
Conv2D_89	(None, None, None,384)	917504
BatchNormalization_89	(None, None, None,384)	1344
Activation_89	(None, None, None,2048)	0
Conv2D_86	(None, None, None,320)	786432
Conv2D_90	(None, None, None,384)	1548288
BatchNormalization_86	(None, None, None,384)	1152
BatchNormalization_90	(None, None, None,384)	1152
Activation_86	(None, None, None,384)	0
Activation_90	(None, None, None,192)	0
Conv2D_87	(None, None, None,320)	442368
Conv2D_88	(None, None, None,384)	442368
Conv2D_91	(None, None, None,384)	442368
Conv2D_92	(None, None, None,384)	442368
AveragePooling_8	(None, None, None,384)	0
Conv2D_85	(None, None, None,192)	655360
BatchNormalization_87	(None, None, None,320)	1152
BatchNormalization_88	(None, None, None,384)	1152

BatchNormalization_91	(None, None, None,384)	1152
BatchNormalization_92	(None, None, None,384)	1152
Conv2D_93	(None, None, None,384)	393216
BatchNormalization_85	(None, None, None,192)	960
Activation_87	(None, None, None,320)	0
Activation_88	(None, None, None,384)	0
Activation_91	(None, None, None,384)	0
Activation_92	(None, None, None,384)	0
BatchNormalization_93	(None, None, None,192)	576
Activation_85	(None, None, None,320)	0
Concatenate	(None, None, None,768)	0
Concatenate	(None, None, None,768)	0
Activation_93	(None, None, None,192)	0
Concatenate	(None, None, None,2048)	0
GlobalaveragePooling	(None, 2048)	0
Dense	(None,1024)	2098176
Dense_1	(None, 38)	38950
Total param: 23,939,910 Trainable params: 13,252,006 Non trainable params: 10,687,904		

This code creates a transfer learning model using the InceptionV3 architecture that has been pre-trained on the ImageNet dataset. The pre-trained model is loaded with the weights specified by the 'weights' parameter and the top layer of the model is excluded by setting 'include_top' to False. A

global average pooling layer is then added to the model to reduce the spatial dimensions of the output from the convolutional layers. A fully connected layer with 1024 units and ReLU activation is added, followed by a softmax layer with 38 units (for classification into 38 classes). The layers of the pre-trained model are set to non-trainable to avoid modifying their weights during training. We used transfer learning to train the InceptionV3 model. The model was trained using the SGD optimizer with a learning rate of 0.005 for 30 epochs.

4.6 Model assessment

The PlantVillage dataset was split into training and test sets using different ratios to identify the optimal deep learning architecture among AlexNet, GoogLeNet, ResNet50, and InceptionV3. By using various train-test ratios, the models' ability to generalize and perform well on unseen data. The goal was to find the best performing architecture that can accurately classify the crop diseases.

1. 90:10 where 90% data is used for training the model and 10% data is used for testing.
2. 80:20 where 80% data is used for training the model and 20% data is used for testing.
3. 70:30 where 70% data is used for training the model and 30% data is used for testing.
4. 60:40 where 60% data is used for training the model and 40% data is used for testing.

Chapter 5

Results

5.1 Experiment Description

This chapter discusses the results from implementation of deep learning architectures discussed in chapter 4. The study's aim is to predict crop_disease pairs using deep learning. There are two learning approaches used to train the models which are training from scratch and transfer learning. The results suggest that AlexNet, GoogLeNet, ResNet50 and InceptionV3 did well in the task. The deep learning models were evaluated based on Accuracy and F1-score.

5.2 Evaluation Metrics

Evaluation metrics such as Accuracy, and F1 Score are used to measure the performance of the deep learning models. We will discuss the evaluation metrics which are used for image classification problem.

Accuracy: it is one of the metric which is used for evaluation of classification models. Accuracy is a measure of how well a classification model can predict the correct class label for each instance in the dataset. It is calculated by dividing the number of correctly classified instances by the total number of instances in the dataset. The resulting value is usually reported as a percentage or a decimal value between 0 and 1.

F1 Score: The F1 score is useful when we want to balance the tradeoff between precision and recall, especially when the classes are imbalanced. A high F1 score means that the model has both high precision and high recall, indicating that it is making accurate predictions while also capturing most of the positive instances in the dataset.

The formula for calculating the F1 score is:

F1 score = 2 * (precision * recall) / (precision + recall)

This formula calculates the harmonic mean of precision and recall, which is then scaled to a range of 0 to 1. The F1 score is a commonly used evaluation metric in machine learning, and it is especially useful when the classes are imbalanced or when we want to balance precision and recall.

Validation Accuracy: Validation accuracy is a performance metric that measures the accuracy of a machine learning model on a validation dataset. In machine learning, we typically split the available data into two sets: the training set and the validation set. We use the training set to train the model and the validation set to evaluate the model's performance.

Validation accuracy is the percentage of correctly classified instances in the validation set. It is a measure of how well the model can generalize to new data that it has not seen before. A higher validation accuracy indicates that the model is performing well on the validation dataset.

Validation Loss: Validation loss is a performance metric that measures the difference between the predicted and actual values of the model on a validation dataset. Validation loss measures the difference between the predicted and actual values in the validation set. It is calculated as the

average of the losses of each instance in the validation set. The loss is typically a measure of how well the model is fitting the data, and a lower validation loss indicates that the model is performing well on the validation dataset.

5.3 Ratio Comparison

For the plantvillage dataset, we compared the performance of deep learning architectures AlexNet, GoogLeNet, ResNet50, and InceptionV3 with different train-test ratios. Specifically, we used train-test ratios of 60:40, 70:30, 80:20, and 90:10 to train and test each architecture. In the PlantVillage dataset, different batch sizes were used for the train, test, and validation sets. Table 5.1 displays number of samples in the training, validation and test datasets for different training-test ratios. We split the testing set into a testing and validation subset with a 50:50 ratio.

Table 5.1 Comparison of DL Training-Test Ratios and Total Instances in Training, Validation, and Testing Sets

Train-Test Ratio	Total instances in training set	Total instances in testing set	Total instances in validation set
90 - 10	48,880	2,740	2,700
80 - 20	43,440	5,420	5,460
70 – 30	38,020	8,160	8,140
60 - 40	32,600	10,860	10,880

5.4 Summary of Results

5.4.1 AlexNet-Training from scratch

AlexNet was trained from scratch in this experiment without using pre-trained model. Table 5.2 shows the results for accuracy, loss and F1 score.

Table 5.2 Results from AlexNet Architecture

Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	**F1 Score**	1.000	1.000	0.988	0.969
	Accuracy	0.998	0.991	0.989	0.982
	Loss	0.015	0.030	0.039	0.073
	Validation Accuracy	0.992	0.988	0.989	0.979
Grayscale	**F1 Score**	0.988	0.988	0.988	0.955
	Accuracy	0.983	0.983	0.980	0.926
	Loss	0.067	0.067	0.059	0.288
	Validation Accuracy	0.983	0.983	0.969	0.930
Segmented Leaf	**F1 Score**	1.000	0.988	0.977	0.966
	Accuracy	0.986	0.985	0.975	0.970
	Loss	0.053	0.067	0.093	0.121
	Validation Accuracy	0.988	0.986	0.987	0.970

The evaluation of the AlexNet model on three different datasets, Color, Grayscale, and Segmented Leaf, demonstrated its effectiveness for crop disease detection. The model achieved high F1 score in Color and Segmented Leaf datasets with train-test split of 90:10. As the size of the training set decreased, the model's performance slightly decreased, but it still achieved good results on smaller training sets. The model's validation accuracy was consistently high across all datasets, indicating its ability to generalize well to unseen data. Overall, the AlexNet model proved to be a powerful tool, demonstrating its effectiveness on different types of image datasets.

5.4.2 GoogLeNet- Training from scratch

GoogleNet was trained from scratch in this experiment without using pre-trained model. Table 5.3 shows the results for accuracy, loss and F1 score.

Table 5.3 Results from GoogLeNet Architecture

Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	**F1 Score**	1.000	1.000	1.000	0.966
	Accuracy	0.999	0.997	0.996	0.975
	Loss	0.016	0.057	0.050	0.314
	Validation Accuracy	0.998	0.997	0.995	0.973
Grayscale	**F1 Score**	0.999	0.977	0.977	0.966
	Accuracy	0.989	0.977	0.977	0.975
	Loss	0.019	0.248	0.213	0.314
	Validation Accuracy	0.988	0.975	0.977	0.973
Segmented Leaf	**F1 Score**	1.000	1.000	0.999	0.988
	Accuracy	0.996	0.994	0.990	0.988
	Loss	0.042	0.067	0.094	0.158
	Validation Accuracy	0.998	0.995	0.995	0.990

The evaluation results of the GoogLeNet architecture on three different datasets, Color, Grayscale, and Segmented Leaf, indicate its strong performance on crop disease detection. The model achieved perfect F1 score and high accuracy on the Color dataset for all train-test splits. For the Grayscale dataset, the model performed well on the 90:10 train-test split, but its performance decreased as the size of the training set reduced. For the Segmented Leaf dataset, the model achieved perfect F1 score and high accuracy on the 90:10 and 80:20 train-test splits. However, as the size of the training set decreased, its performance slightly decreased, but it still achieved good results on the 60:40 train-test split. The model's validation accuracy was consistently high across all datasets, indicating its generalization ability to new data. Overall, the results of GoogLeNet architecture demonstrate that it is effective in image classification tasks on different types of datasets.

5.4.3 ResNet50- Training from scratch

ResNet50 was trained from scratch in this experiment without using pre-trained model. Table 5.4 shows the results for accuracy, loss and F1 score.

Table 5.4 Results from ResNet50 Architecture

Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	F1 Score	1.000	1.000	0.966	0.966
	Accuracy	0.980	0.994	0.981	0.988
	Loss	0.057	0.023	0.065	0.045
	Validation Accuracy	0.985	0.993	0.978	0.988
Grayscale	F1 Score	0.977	0.966	0.933	0.900
	Accuracy	0.958	0.941	0.912	0.894
	Loss	0.179	0.177	0.266	0.328
	Validation Accuracy	0.963	0.940	0.901	0.884
Segmented Leaf	F1 Score	1.000	1.000	0.977	0.966
	Accuracy	0.993	0.985	0.988	0.960
	Loss	0.061	0.048	0.027	0.124
	Validation Accuracy	0.995	0.987	0.988	0.960

The evaluation of the ResNet50 architecture on three different datasets, Color, Grayscale, and Segmented Leaf, while doing learning from scratch, shows its effectiveness in leaf disease detection. The model achieved high F1 score on the Color and Segmented Leaf datasets with 90:10 and 80:20 train-test splits. For the Grayscale dataset, the model's performance decreased as the size of the training set decreased, but it still achieved reasonable results on smaller training sets. Overall, the results demonstrate that the model is effective in crop disease detection tasks on different types of datasets, especially on larger training sets.

5.4.4 ResNet50- Transfer learning

ResNet50 was trained using transfer learning in this experiment. Table 5.5 shows the results for accuracy, loss and F1 score.

Table 5.5 Results from ResNet50 Architecture (Pre-trained)

Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	**F1 Score**	0.999	0.988	0.977	0.977
	Accuracy	0.994	0.992	0.990	0.989
	Loss	0.026	0.039	0.033	0.040
	Validation Accuracy	0.996	0.992	0.990	0.989
Grayscale	**F1 Score**	0.988	0.977	0.977	0.977
	Accuracy	0.988	0.982	0.988	0.975
	Loss	0.054	0.072	0.085	0.089
	Validation Accuracy	0.987	0.981	0.983	0.973
Segmented Leaf	**F1 Score**	0.988	0.988	0.988	0.988
	Accuracy	0.985	0.983	0.979	0.978
	Loss	0.066	0.071	0.077	0.080
	Validation Accuracy	0.983	0.984	0.989	0.978

The ResNet50 model was pre-trained on ‘ImageNet’ dataset, and transfer learning was used to train and test three different datasets: color, grayscale, and segmented leaf. The performance of the model was evaluated using F1 score, accuracy, loss, and validation accuracy metrics. The results showed that the ResNet50 model achieved good accuracy and F1 score on all three datasets, with the highest performance seen in the color dataset. The model also exhibited relatively low loss values, indicating efficient training.

The pre-trained model outperforms the non-pretrained model across all datasets in terms of accuracy and F1 score. The validation accuracy is also consistently higher for the pre-trained model.

5.4.5 InceptionV3- Training from scratch

InceptionV3 was trained from scratch in this experiment without using pre-trained model. Table 5.6 shows the results for accuracy, loss and F1 score.

Table 5.6 Results from InceptionV3 Architecture

Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	**F1 Score**	1.000	1.000	1.000	1.000
	Accuracy	0.997	0.995	0.989	0.985
	Loss	0.011	0.015	0.013	0.050
	Validation Accuracy	0.996	0.994	0.993	0.986
Grayscale	**F1 Score**	0.999	0.988	0.955	0.944
	Accuracy	0.994	0.980	0.978	0.961
	Loss	0.046	0.065	0.084	0.118
	Validation Accuracy	0.988	0.983	0.978	0.960
Segmented Leaf	**F1 Score**	1.000	1.000	0.977	0.966
	Accuracy	0.996	0.994	0.993	0.981
	Loss	0.022	0.025	0.037	0.060
	Validation Accuracy	0.995	0.990	0.988	0.982

The results of training the InceptionV3 model from scratch on the three datasets - color, grayscale, and segmented leaf - showed that the model achieved high accuracy and F1 scores on the color and segmented leaf datasets, with perfect F1 score achieved on the color dataset. However, the performance on the grayscale dataset was comparatively lower, with a decrease in F1 score and accuracy as the proportion of training data decreased. The model exhibited relatively low loss values and high validation accuracy, suggesting good generalization performance. These results demonstrate the potential of the InceptionV3 model, especially in scenarios with high levels of complexity and variability in image characteristics, and highlight the importance of selecting appropriate datasets and proportions for training and testing.

5.4.6 InceptionV3- Transfer learning

InceptionV was trained using transfer learning in this experiment with pre-trained model on ImageNet dataset. Table 5.7 shows the results for accuracy, loss and F1 score.

Table 5.7 Results from InceptionV3 Architecture(Pre-trained)

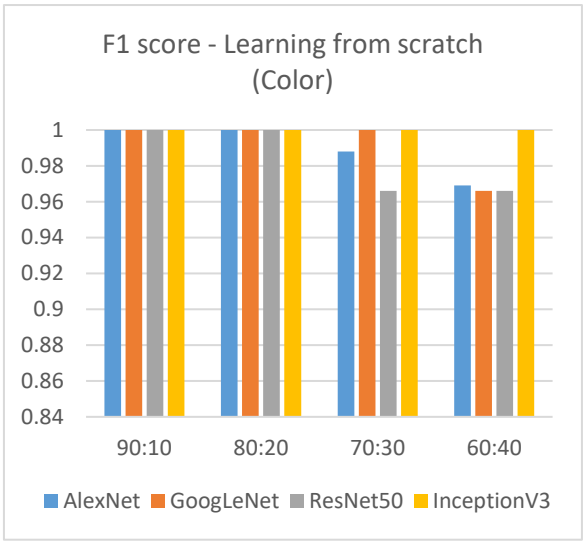
Dataset	Train-Test	90:10	80:20	70:30	60:40
Color	**F1 Score**	0.977	0.966	0.955	0.933
	Accuracy	0.955	0.957	0.937	0.901
	Loss	0.196	0.238	0.498	0.602
	Validation Accuracy	0.964	0.957	0.924	0.906
Grayscale	**F1 Score**	0.966	0.944	0.911	0.922
	Accuracy	0.958	0.924	0.912	0.905
	Loss	0.213	0.328	0.369	0.455
	Validation Accuracy	0.977	0.969	0.909	0.897
Segmented Leaf	**F1 Score**	1.000	0.966	0.944	0.877
	Accuracy	0.987	0.943	0.921	0.893
	Loss	0.125	0.341	0.496	0.662
	Validation Accuracy	0.966	0.942	0.927	0.896

The table shows the performance metrics for the InceptionV3 model with transfer learning on three different datasets: Color, Grayscale, and Segmented Leaf. Generally, the model performs well on all datasets, achieving good Accuracy and F1 score. For Segmented Leaf dataset the model achieved highest F1 Score with 90:10 train-test split. The validation accuracy is also provided, which indicates how well the model performs on unseen data.

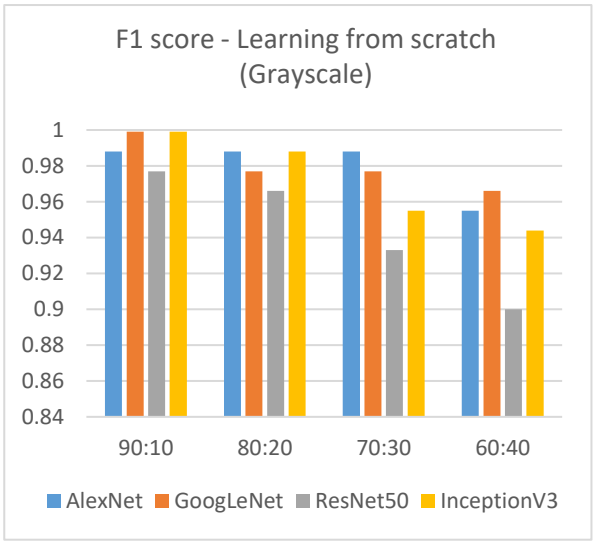
The results of non-pretrained InceptionV3 architecture show higher F1 score and accuracy compared to the pretrained InceptionV3 architecture, which shows lower scores for the pre-trained InceptionV3 architecture. Additionally, the validation accuracy is consistently higher for non-pretrained InceptionV3 architecture compared to the pretrained InceptionV3 architecture.

5.6 Comparison of deep learning architecture

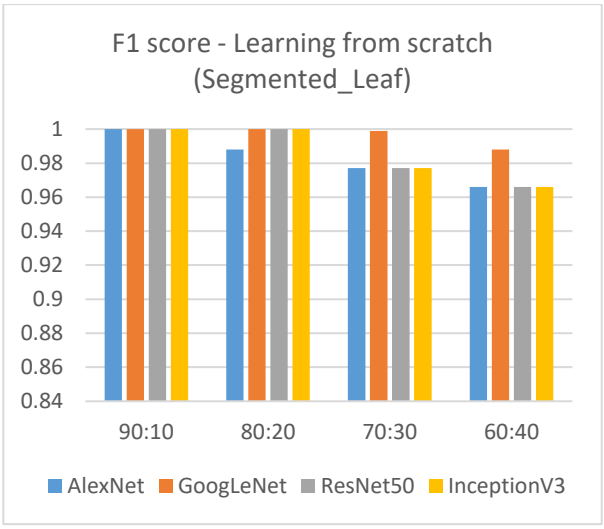
Figure 5.1 shows the bar graph for the F1 score of all the models trained from scratch on the three datasets of color, grayscale and segmented leaf for all the training-testing ratios.



a)



b)



c)

Figure 5.1 F1 score from all non-pretrained models. a) F1 score - Learning from scratch (Color), b) F1 score - Learning from scratch (GrayScale), c) F1 score - Learning from scratch (Segmented_Leaf).

Overall, the models performed well on the color dataset, achieving F1 score of 96.9% to 100%. The grayscale dataset also yielded high F1 score, with models achieving between 90% and 99.9%. The segmented_leaf dataset produced lower accuracy rates, with models achieving between 90% and 99.9%.

The InceptionV3 model achieved the highest accuracy rates in most cases, followed by GoogLeNet and ResNet50. The results suggest that the choice of dataset and model architecture has a significant impact on the accuracy of image classification tasks, highlighting the importance of proper dataset selection and model optimization for achieving high-performance image classification.

Figure 5.2 shows the bar graph for the F1 score of all the models pre-trained on ImageNet dataset and transfer learning applied to the three datasets of color, grayscale and segmented leaf for all the training-testing ratios.

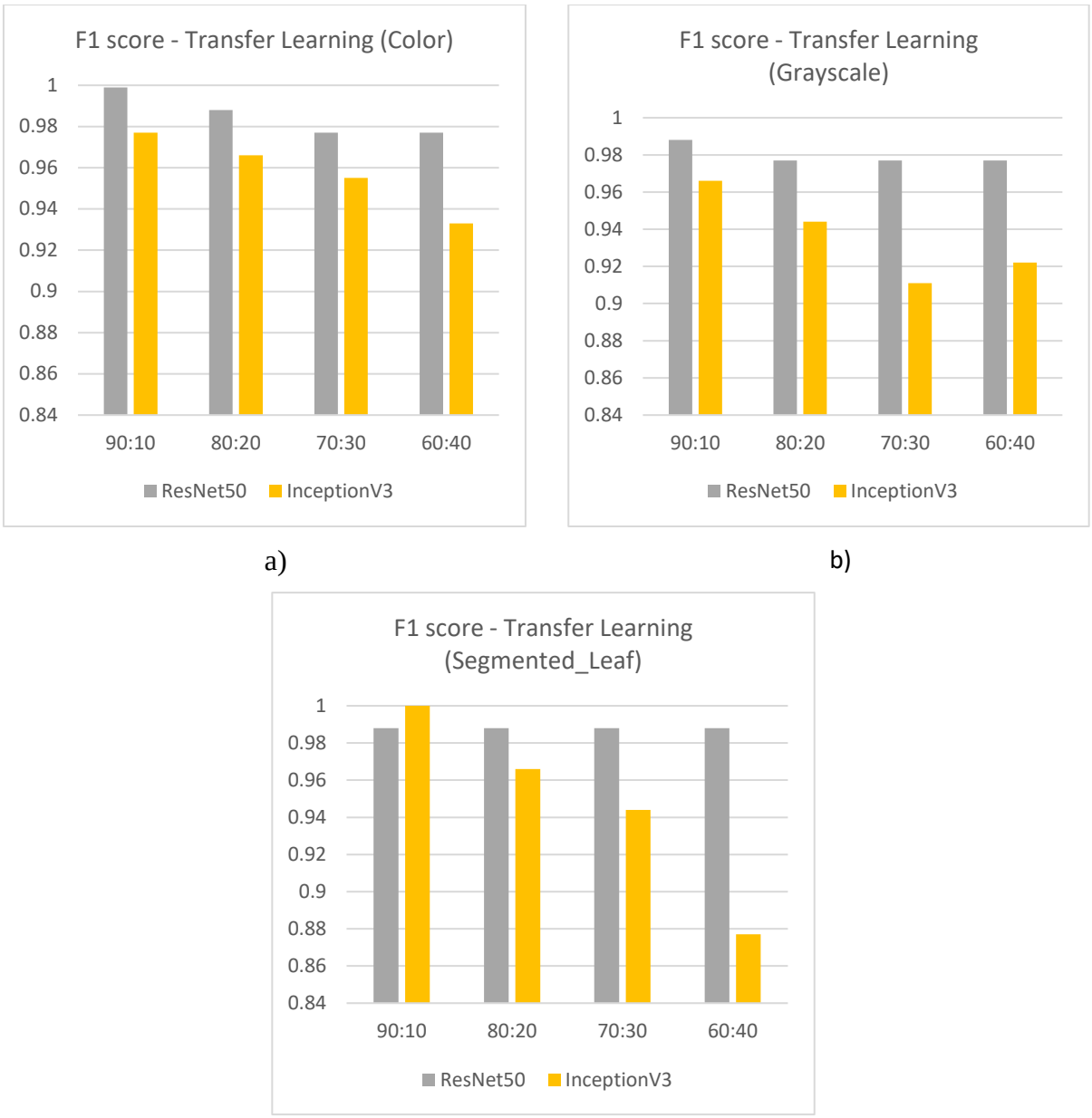


Figure 5.2 F1 score from all pretrained models. a) F1 score – Transfer Learning (Color), b) F1 score - Transfer Learning (GrayScale), c) F1 score – Transfer Learning (Segmented_Leaf).

Overall, the transfer learning models performed well on all three datasets, with ResNet50 and InceptionV3 achieving high F1 score of up to 99.9% on the color and grayscale datasets. However, the F1 score were lower for the segmented_leaf dataset, with models achieving between 87.7% and 98.8%.

The results indicate that transfer learning can significantly improve the accuracy of image classification tasks, especially for datasets with limited training samples. The ResNet50 and InceptionV3 models performed similarly well in most cases, highlighting their suitability for a wide range of image classification tasks.

Figure 5.3 shows the bar graph for the F1 score of all the models trained from scratch and pre-trained models on the three datasets of color, grayscale and segmented leaf for 90:10 training-testing ratio.

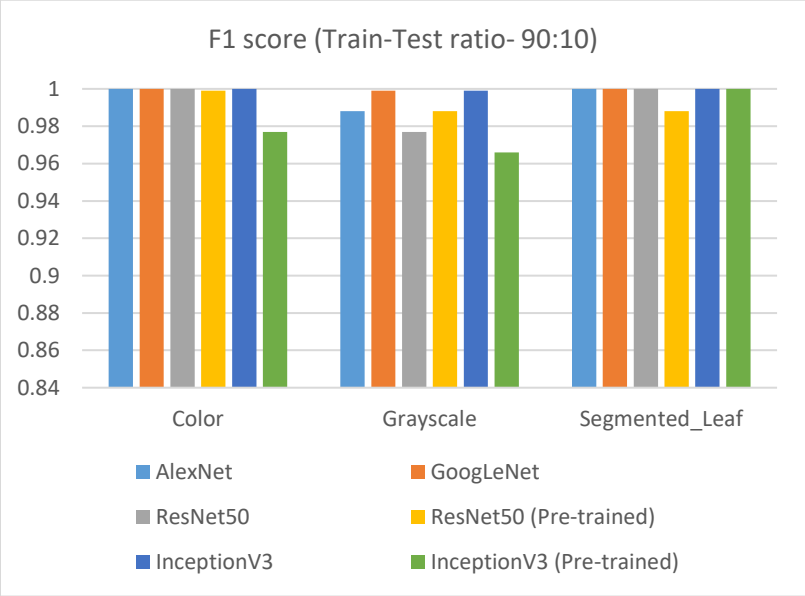


Figure 5.3 F1 score (Train-Test ratio – 90:10)

Overall, the models achieved high F1 score, with most models achieving perfect score of 1.0 on the color and segmented_leaf datasets. The pre-trained ResNet50 and InceptionV3 models performed well on the grayscale dataset, achieving F1 scores of 0.988 and 0.966, respectively. The results suggest that these models are highly effective in accurately classifying images across various image datasets, highlighting their suitability for a wide range of image classification tasks.

Pre-trained models can offer significant advantages in scenarios with limited training data, resulting in improved classification performance.

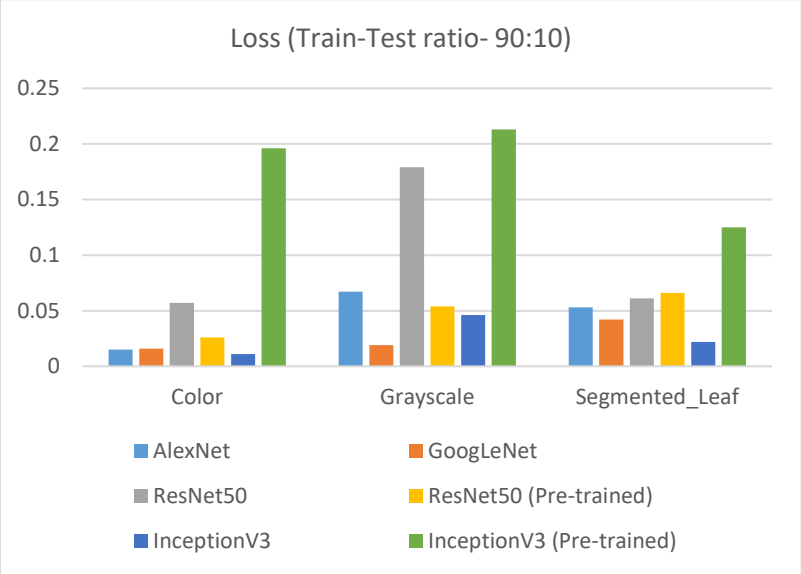


Figure 5.4 Loss (Train-Test ratio – 90:10)

Figure 5.4 shows the loss results for six different convolutional neural network models (AlexNet, GoogLeNet, ResNet50, ResNet50 pre-trained, InceptionV3, and InceptionV3 pre-trained) on three image datasets (color, grayscale, and segmented_leaf) at a train-test ratio of 90:10. Overall, the models achieved low loss values, indicating good performance in image classification tasks. The pre-trained InceptionV3 model achieved the lowest loss values on all three datasets, while the pre-trained ResNet50 and AlexNet models achieved relatively lower loss values on the color and grayscale datasets, respectively. The results suggest that these models are effective in accurately classifying images across various image datasets, with pre-trained models achieving superior performance due to the transfer of knowledge from large image datasets. The low loss values achieved by the models indicate good generalization ability and suggest their suitability for use in practical applications.

Table 5.8 Performance Comparison of Different CNN Models on Various Train-Test Splits

		AlexNet		GoogLeNet		ResNet50				InceptionV3			
		Learning from scratch		Learning from scratch		Learning from scratch		Transfer Learning		Learning from scratch		Transfer Learning	
Train-Test Split		**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**	**F1 Score**	**Accuracy**
90%-10%	Color	1	0.998	1	0.999	1	0.980	0.999	0.994	1	0.997	0.977	0.955
	Grayscale	0.988	0.983	0.999	0.989	0.977	0.958	0.988	0.988	0.999	0.994	0.966	0.958
	segmented	1	0.986	1	0.996	1	0.993	0.988	0.985	1	0.996	1	0.987
80%-20%	Color	1	0.991	1	0.997	1	0.994	0.988	0.992	1	0.995	0.966	0.957
	Grayscale	0.988	0.983	0.977	0.977	0.966	0.941	0.977	0.982	0.988	0.980	0.944	0.924
	segmented	0.988	0.985	1	0.994	1	0.985	0.988	0.983	1	0.994	0.966	0.943
70%-30%	Color	0.988	0.989	1	0.996	0.966	0.981	0.977	0.990	1	0.989	0.955	0.937
	Grayscale	0.988	0.98	0.977	0.977	0.933	0.912	0.977	0.988	0.955	0.978	0.911	0.912
	segmented	0.977	0.975	0.999	0.990	0.977	0.988	0.988	0.979	0.977	0.993	0.944	0.921
60%-40%	Color	0.969	0.982	0.966	0.975	0.966	0.988	0.977	0.989	1	0.985	0.933	0.901
	Grayscale	0.955	0.926	0.966	0.975	0.900	0.894	0.977	0.975	0.944	0.961	0.922	0.905
	segmented	0.966	0.970	0.988	0.988	0.966	0.960	0.988	0.978	0.966	0.981	0.877	0.893

Table 5.8 compares the performance of four different Convolutional Neural Network (CNN) models, namely AlexNet, GoogLeNet, ResNet50, and InceptionV3, on various train-test splits and different types of images (color, grayscale, and segmented). The evaluation metrics used are F1 Score and Accuracy. GoogLeNet architecture achieved the highest accuracy of 0.999 for color images and 0.996 for segmented images. For greyscale InceptionV3 trained from scratch gave the highest accuracy of 0.994. However, AlexNet, GoogleNet, ReNet50 trained from scratch and InceptionV3 gave the highest F1 score of 1.0. The best results were achieved with training:testing ratio of 90:10.

Table 5.9 Performance Comparison of AlexNet (Learning from scratch) and GoogLeNet (Learning from scratch) Models on Various Train-Test Splits [17]

Learning from scratch		**AlexNet**		**GoogLeNet**	
		Based on [17] Research paper	**Based on our implementation**	**Based on [17] Research paper**	**Based on our implementation**
Train-Test Split		**F1 Score**	**F1 Score**	**F1 Score**	**F1 Score**
80%-20%	Color	0.9782	1.0000	0.9836	1.0000
	Grayscale	0.9449	0.9888	0.9621	0.9777
	segmented	0.9722	0.9888	0.9824	1.0000
60%-40%	Color	0.9724	0.9699	0.9824	0.9666
	Grayscale	0.9388	0.9555	0.9547	0.9666
	segmented	0.9595	0.9666	0.9740	0.9888
50%-50%	Color	0.9644	0.9666	0.9772	1.0000
	Grayscale	0.9312	0.9111	0.9507	0.9444
	segmented	0.9551	0.9666	0.9720	0.9777
40%-60%	Color	0.9555	0.9000	0.9729	0.9888
	Grayscale	0.9088	0.9444	0.9361	0.9555
	segmented	0.9404	0.9666	0.9643	0.9666
20%-80%	Color	0.9118	0.8888	0.9430	0.9555
	Grayscale	0.8524	0.8666	0.8828	0.8666
	segmented	0.8945	0.8666	0.9377	0.9777

Table 5.9 compares the performance of two different Convolutional Neural Network (CNN) models, namely AlexNet, GoogLeNet, on various train-test splits and different types of images (color, grayscale, and segmented) in the paper by S. P. Mohanty et al. [17]. In their GoogLeNet architecture, they achieved the highest accuracy of 0.9934 for color images and 0.9925 for segmented images. In comparison to the results obtained in [17], our findings demonstrate a significant improvement. Specifically, we achieved a F1 score of 1 when utilizing an 80% - 20% train-test split, surpassing the performance of both AlexNet and GoogLeNet models.

Chapter 6

Conclusion

6.1 Conclusions

Agriculture is stated as an important sector, which can be helpful in the development of the economy and in reducing the issues associated with food scarcity. Many countries need to improve agriculture practices and utilize the advanced technologies that can be helpful in the reduction of various issues. In the present time, disease in the crops is a big issue that is affecting the productivity of the agricultural lands. The utilization of artificial intelligence and computer vision can be helpful in the detection of various diseases. Therefore, the objectives of this research work were to analyze the Image-based crop disease detection. The other objectives of the work were to analyze the types of crop diseases and relevant areas used during crop disease detection along with the identification of the challenges that occur during crop disease detection and recommending advanced image detection techniques for improving crop disease detection.

In comparison to other research studies, our work offers several notable contributions. Firstly, we trained and evaluated four distinct deep learning (DL) models using both learning from scratch and transfer learning approaches. By fine-tuning our trained models on different datasets, we can leverage the knowledge gained from our initial training and adapt it to new domains or specific tasks. This capability opens up possibilities for improving accuracy, robustness, and generalizability across various applications.

Using the deep convolutional neural network architectures (AlexNet, GoogLeNet, ResNet50, and InceptionV3) we trained the models on images of plant leaves to predict crop-disease pair. Within the PlantVillage dataset of 54,306 images containing 38 classes of 14 crop species and 26 diseases, this goal has been achieved as demonstrate by the top F1 score of 1.0. In learning from scratch we found that all deep convolutional neural network architectures (AlexNet, GoogLeNet, ResNet50, and InceptionV3) performed well while GoogleNet was the best. In transfer learning ResNet50 performed well compared to InceptionV3. The deep learning architectures used in this study achieved high accuracy and F1 score, indicating their potential for crop disease detection using crop images.

6.2 Limitations

The research encountered certain limitations stemming from constraints in hardware resources and data availability, as well as the need for more advanced deep learning (DL) models. Firstly, the limited hardware resources posed a challenge in terms of computational power and processing capabilities. The absence of high-performance computing infrastructure restricted the complexity and scale of the DL models that could be implemented. Consequently, the potential for exploring larger and more intricate models was constrained. Secondly, the availability and quality of the dataset proved to be a limiting factor. The research relied on a specific dataset, which might have had limitations in terms of size, diversity, or representativeness. The restricted dataset might have constrained the overall performance and generalizability of the DL models employed. Additionally, the rapid advancements in DL techniques and architectures necessitated further exploration of more advanced models. Understanding these limitations helps in interpreting the findings appropriately and provides valuable insights for future research endeavors.

6.3 Directions for future work

For future research, it is recommended to use larger datasets with more images to further evaluate the performance of CNN models. Additionally, more computationally powerful deep learning architectures could be explored to potentially improve classification accuracy. These improvements can contribute to the development of more accurate and efficient crop disease detection. As technology continues to advance, it is possible that in the future, image data collected from smartphones for image classification tasks could be supplemented with location and time information. By incorporating this additional information, it may be possible to further enhance the accuracy and reliability for crop disease detection. Based on the findings of such research, a smartphone-assisted crop disease diagnosis system could be developed. Such a system has the potential to significantly benefit the agricultural industry by providing a cost-effective and easily accessible solution for crop disease detection and prevention.

It is also recommended that the farmers and various stakeholders should be trained according to the need for modern technology so that various issues can be reduced. It is suggested that the information, which is taken for the experimental setup, can be improved and large data sets can be analyzed for the improvement of the result accuracy.

References

[1] V. Panchal, S. C. Patel, K. Bagyalakshmi, P. Kumar, I. R. Khan, and M. Soni, "Image-based Plant Diseases Detection using Deep Learning," *Materials Today: Proceedings*, Aug. 2021, doi: 10.1016/j.matpr.2021.07.281.

[2] J. Chen, J. Chen, D. Zhang, Y. Sun, and Y. A. Nanehkaran, "Using deep transfer learning for image-based plant disease identification," *Computers and Electronics in Agriculture*, vol. 173, p. 105393, Jun. 2020, doi: 10.1016/j.compag.2020.105393.

[3] Jain, S. Sarsaiya, Q. Wu, Y. Lu, and J. Shi, "A review of plant leaf fungal diseases and its environment speciation," *Bioengineered*, vol. 10, no. 1, pp. 409–424, Jan. 2019, doi: 10.1080/21655979.2019.1649520.

[4] K. Nagasubramanian, S. Jones, A. K. Singh, S. Sarkar, A. Singh, and B. Ganapathysubramanian, "Plant disease identification using explainable 3D deep learning on hyperspectral images," *Plant Methods*, vol. 15, no. 1, Aug. 2019, doi: 10.1186/s13007-019-0479-8.

[5] M. H. Saleem, J. Potgieter, and K. M. Arif, "Plant Disease Classification: A Comparative Evaluation of Convolutional Neural Networks and Deep Learning Optimizers," *Plants*, vol. 9, no. 10, p. 1319, Oct. 2020, doi: 10.3390/plants9101319. [Online]. Available: https://www.mdpi.com/2223-7747/9/10/1319.

[6] D. Andújar, A. Ribeiro, C. Fernández-Quintanilla, and J. Dorado, "Using depth cameras to extract structural parameters to assess the growth state and yield of cauliflower crops," *Computers and Electronics in Agriculture*, vol. 122, pp. 67–73, Mar. 2016, doi:

10.1016/j.compag.2016.01.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169916000235.

[7] N. Yang, V. Joos, A-L. Jacquemart, C. Buyens, and C. De Vleeschouwer, "Using Pure Pollen Species When Training a CNN to Segment Pollen Mixtures," 2022 [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022W/AgriVision/papers/Yang_Using_Pure_Pollen_Species_When_Training_a_CNN_To_Segment_CVPRW_2022_paper.pdf.

[8] M. K. Alsmadi, "Content-Based Image Retrieval Using Color, Shape and Texture Descriptors and Features," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 3317–3330, Feb. 2020, doi: 10.1007/s13369-020-04384-y.

[9] N. Khan, R. L. Ray, G. R. Sargani, M. Ihtisham, M. Khayyam, and S. Ismail, "Current Progress and Future Prospects of Agriculture Technology: Gateway to Sustainable Agriculture," *Sustainability*, vol. 13, no. 9, p. 4883, Apr. 2021, doi: 10.3390/su13094883.

[10] B. A. M. Ashqar and S. S. Abu-Naser, "Image-Based Tomato Leaves Diseases Detection Using Deep Learning," *dstore.alazhar.edu.ps*, 2018 [Online]. Available: http://dstore.alazhar.edu.ps/xmlui/handle/123456789/278.

[11] Y. Guo *et al.*, "Plant Disease Identification Based on Deep Learning Algorithm in Smart Farming," *Discrete Dynamics in Nature and Society*, vol. 2020, pp. 1–11, Aug. 2020, doi: 10.1155/2020/2479172.

[12] K. P. Panigrahi, H. Das, A. K. Sahoo, and S. C. Moharana, "Maize Leaf Disease Detection and Classification Using Machine Learning Algorithms," *Advances in Intelligent Systems and Computing*, pp. 659–669, 2020, doi: 10.1007/978-981-15-2414-1_66.

[13] M. Arsenovic, M. Karanovic, S. Sladojevic, A. Anderla, and D. Stefanovic, “Solving Current Limitations of Deep Learning Based Approaches for Plant Disease Detection,” *Symmetry*, vol. 11, no. 7, p. 939, Jul. 2019, doi: 10.3390/sym11070939.

[14] H. van Bruggen, A. Gamliel, and M. R. Finckh, “Plant disease management in organic farming systems,” *Pest Management Science*, vol. 72, no. 1, pp. 30–44, Oct. 2016, doi: 10.1002/ps.4145.

[15] M. Donatelli, R. D. Magarey, S. Bregaglio, L. Willocquet, J. P. M. Whish, and S. Savary, “Modelling the impacts of pests and diseases on agricultural systems,” *Agricultural Systems*, vol. 155, pp. 213–224, Jul. 2017, doi: 10.1016/j.agsy.2017.01.019.

[16] Lee, D.I.; Lee, J.H.; Jang, S.H.; Oh, S.J.; Doo, I.C. Crop Disease Diagnosis with Deep Learning-Based Image Captioning and Object Detection. Appl. Sci. 2023, 13, 3148. https://doi.org/10.3390/app13053148

[17] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using Deep Learning for Image-Based Plant Disease Detection,” *Frontiers in Plant Science*, vol. 7, Sep. 2016, doi: 10.3389/fpls.2016.01419.

[18] K. Singh, B. Ganapathysubramanian, S. Sarkar, and A. Singh, “Deep Learning for Plant Stress Phenotyping: Trends and Future Perspectives,” *Trends in Plant Science*, vol. 23, no. 10, pp. 883–898, Oct. 2018, doi: 10.1016/j.tplants.2018.07.004.

[19] N. M. Nafi and W. H. Hsu, “Addressing Class Imbalance in Image-Based Plant Disease Detection: Deep Generative vs. Sampling-Based Approaches,” *IEEE Xplore*, Jul. 01, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9145239/.

[20] K. Dokic, L. Blaskovic, and D. Mandusic, “From machine learning to deep learning in agriculture – the quantitative review of trends,” *IOP Conference Series: Earth and*

Environmental Science, vol. 614, no. 1, p. 012138, Dec. 2020, doi: 10.1088/1755-1315/614/1/012138.

[21] Javaid, Q. Niyaz, W. Sun, and M. Alam, "A Deep Learning Approach for Network Intrusion Detection System," *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, 2016, doi: 10.4108/eai.3-12-2015.2262516.

[22] D. Andújar, A. Ribeiro, C. Fernández-Quintanilla, and J. Dorado, "Using depth cameras to extract structural parameters to assess the growth state and yield of cauliflower crops," *Computers and Electronics in Agriculture*, vol. 122, pp. 67–73, Mar. 2016, doi: 10.1016/j.compag.2016.01.018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168169916000235.

[23] S. Dargan, M. Kumar, M. R. Ayyagari, and G. Kumar, "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning," *Archives of Computational Methods in Engineering*, vol. 27, no. 4, pp. 1071–1092, Jun. 2020, doi: 10.1007/s11831-019-09344-w.

[24] A.-K. Mahlein, "Plant Disease Detection by Imaging Sensors – Parallels and Specific Demands for Precision Agriculture and Plant Phenotyping," *Plant Disease*, vol. 100, no. 2, pp. 241–251, Feb. 2016, doi: 10.1094/pdis-03-15-0340-fe.

[25] M. Jia *et al.*, "Quantifying Chlorophyll Fluorescence Parameters from Hyperspectral Reflectance at the Leaf Scale under Various Nitrogen Treatment Regimes in Winter Wheat," *Remote Sensing*, vol. 11, no. 23, p. 2838, Jan. 2019, doi: 10.3390/rs11232838. [Online]. Available: https://www.mdpi.com/584528.

[26] M. T. Kuska and A.-K. . Mahlein, “Aiming at decision making in plant disease protection and phenotyping by the use of optical sensors,” *European Journal of Plant Pathology*, vol. 152, no. 4, pp. 987–992, Mar. 2018, doi: 10.1007/s10658-018-1464-1.

[27] G. Polder, N. van de Westeringh, J. Kool, H. A. Khan, G. Kootstra, and A. Nieuwenhuizen, “Automatic Detection of Tulip Breaking Virus (TBV) Using a Deep Convolutional Neural Network,” *IFAC-PapersOnLine*, vol. 52, no. 30, pp. 12–17, 2019, doi: 10.1016/j.ifacol.2019.12.482.

[28] S. Thomas *et al.*, “Benefits of hyperspectral imaging for plant disease detection and plant protection: a technical perspective,” *Journal of Plant Diseases and Protection*, vol. 125, no. 1, pp. 5–20, Sep. 2018, doi: 10.1007/s41348-017-0124-6.

[29] F. Cheshkova, “A review of hyperspectral image analysis techniques for plant disease detection and identif ication,” *Vavilov Journal of Genetics and Breeding*, vol. 26, no. 2, pp. 202–213, Apr. 2022, doi: 10.18699/vjgb-22-25. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8983301/.

[30] Q. Mu, Z. Kang, Y. Guo, L. Chen, S. Wang, and Y. Zhao, “Hyperspectral image classification of wolfberry with different geographical origins based on three-dimensional convolutional neural network,” *International Journal of Food Properties*, vol. 24, no. 1, pp. 1705–1721, Jan. 2021, doi: 10.1080/10942912.2021.1987457.

[31] R. Pieruschka and U. Schurr, “Plant Phenotyping: Past, Present, and Future,” *Plant Phenomics*, vol. 2019, pp. 1–6, Mar. 2019, doi: 10.34133/2019/7507131.

[32] Afzal, Z. K. Shinwari, S. Sikandar, and S. Shahzad, “Plant beneficial endophytic bacteria: Mechanisms, diversity, host range and genetic determinants,” *Microbiological Research*,

vol. 221, pp. 36–49, Apr. 2019, doi: 10.1016/j.micres.2019.02.001. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0944501318304592.

[33] L. K. Hema, D. Vijendra Babu, A. Navaneetharajan, K. Vijayakumar, and S. Dhayanithi, "Agriculture Resources for Plant-Leaf Disease Identification using Deep Learning Techniques," *Journal of Physics: Conference Series*, vol. 1964, no. 6, p. 062027, Jul. 2021, doi: 10.1088/1742-6596/1964/6/062027.

[34] S. S. Kumar and B. K. Raghavendra, "Diseases Detection of Various Plant Leaf Using Image Processing Techniques: A Review," *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)*, Mar. 2019, doi: 10.1109/icaccs.2019.8728325.

[35] F. Martinelli, R. Scalenghe, S. Davino, S. Panno, G. Scuderi, P. Ruisi, ...and A.M. Dandekar, "Advanced methods of plant disease detection. A review", *Agronomy for Sustainable Development*, vol. 35, pp. 1-25, 2015 [online]. Available: https://link.springer.com/article/10.1007/s13593-014-0246-1

[36] V. Singh and A. K. Misra, "Detection of plant leaf diseases using image segmentation and soft computing techniques," *Information Processing in Agriculture*, vol. 4, no. 1, pp. 41–49, Mar. 2017, doi: 10.1016/j.inpa.2016.10.005.

[37] R. Katafuchi and T. Tokunaga, "Image-based Plant Disease Diagnosis with Unsupervised Anomaly Detection Based on Reconstructability of Colors," *arXiv:2011.14306 [cs]*, Sep. 2021 [Online]. Available: https://arxiv.org/abs/2011.14306.

[38] G. Arnal Barbedo, "Plant disease identification from individual lesions and spots using deep learning," *Biosystems Engineering*, vol. 180, pp. 96–107, Apr. 2019, doi: 10.1016/j.biosystemseng.2019.02.002.

[39] Newhart, K.B., Holloway, R.W., Hering, A.S. and Cath, T.Y., 2019. Data-driven performance analyses of wastewater treatment plants: A review. Water research, 157, pp.498-513.

[40] S. Ghosh, "SOME STUDIES ON DIFFERENT DATA MINING APPROACHES" (Doctoral dissertation, UNIVERSITY OF KALYANI). 2015 [Online] https://www.researchgate.net/profile/Soumadip-Ghosh/publication/323401290_Some_Studies_on_Different_Data_Mining_Approaches/links/5a950ee6aca272140567a1c2/Some-Studies-on-Different-Data-Mining-Approaches.pdf

[41] D avid. P. Hughes, M. S. (2016). *An open access repository of images on plant health to enable the development of mobile disease diagnostics.* Retrieved from https://arxiv.org/abs/1511.08060

[42] Sapkal and U. Kulkarni, "Comparative study of Leaf Disease Diagnosis system using Texture features and Deep Learning Features," *International Journal of Applied Engineering Research*, vol. 13, no. 19, pp. 14334–14340, 2018 [Online]. Available: https://www.ripublication.com/ijaer18/ijaerv13n19_39.pdf

[43] Magsi, J. A. Mahar, M. A. Razzaq, and S. H. Gill, "Date Palm Disease Identification Using Features Extraction and Deep Learning Approach," *IEEE Xplore*, Nov. 01, 2020. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9318158/.

[44] M. K. Alsmadi, "Content-Based Image Retrieval Using Color, Shape and Texture Descriptors and Features," *Arabian Journal for Science and Engineering*, vol. 45, no. 4, pp. 3317–3330, Feb. 2020, doi: 10.1007/s13369-020-04384-y.

[45] X. Xie, Y. Ma, B. Liu, J. He, S. Li, and H. Wang, “A Deep-Learning-Based Real-Time Detector for Grape Leaf Diseases Using Improved Convolutional Neural Networks,” *Frontiers in Plant Science*, vol. 11, Jun. 2020, doi: 10.3389/fpls.2020.00751.

[46] M. VIANA, “Usage and economic values of the timber of nonnative tree species in the Alps,” pp. 1–90, 2022.

[47] J. Deng, W. Dong, R. Socher, L. -J. Li, Kai Li and Li Fei-Fei, "ImageNet: A large-scale hierarchical image database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, 2009, pp. 248-255, doi: 10.1109/CVPR.2009.5206848.

[48] S. P. Mukherjee, “A Guide to Research Methodology,” Sep. 2019, doi: 10.1201/9780429289095.

[49] M. A. Ragab and A. Arisha, “(PDF) Research Methodology in Business: A Starter’s Guide,” *ResearchGate*, 2018. [Online]. Available: https://www.researchgate.net/profile/Mohamed-Ragab-22/publication/321769066_Research_Methodology_in_Business_A_Starter

[50] S. A. Sanchez, H. J. Romero, and A. D. Morales, “A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework,” *IOP Conference Series: Materials Science and Engineering*, vol. 844, no. 1, p. 012024, Jun. 2020, doi: 10.1088/1757-899x/844/1/012024.

[51] H. Jin, Q. Song, and X. Hu, “Auto-Keras: An Efficient Neural Architecture Search System,” *arXiv.org*, 2019. [Online]. Available: https://arxiv.org/abs/1806.10282

[52] Rafid and I. Rafid, “Performance evaluation for Kruskal’s and Prim’s Algorithm in Minimum Spanning Tree using Networkx Package and Matplotlib to visualizing the MST

Result Performance evaluation for Kruskal's and Prim's Algorithm in Minimum Spanning Tree using Networkx Package and Matplotlib to visualizing the MST Result," 2019.

[53] P. Lemenkova, "R Libraries {dendextend} and {magrittr} and Clustering Package scipy.cluster of Python For Modelling Diagrams of Dendrogram Trees," *Carpathian Journal of Electronic and Computer Engineering*, vol. 13, no. 1, pp. 5–12, Sep. 2020, doi: 10.2478/cjece-2020-0002.

[54] Koech, K. E. (2020). Cross-Entropy Loss Function. *Towards Data Science*, 1. Retrievedfrom https://towardsdatascience.com/cross-entropy-loss-function-f38c4ec8643e

[55] Ruder, S. (2016, 1 19). *An overview of gradient descent optimization algorithms*. Retrieved from ruder.io: https://www.ruder.io/optimizing-gradient

[56] Turhan, F. (2019, 12 9). Week 2 – Plant Disease Detection. Retrieved from https://medium.com:https://medium.com/bbm406f19/week-2-plant-disease-detection-9bdd819b870

[57] Alake, R. (2020). What AlexNet Brought To The World Of Deep Learning. *Medium*, 1. Retrieved from https://towardsdatascience.com/what-alexnet-brought-to-the-world-of-deep-learning-46c7974b46fc

[58] Lenyk, Z. (2021, 2 3). *Microsoft Vision Model ResNet-50 combines web-scale data and multi-task learning to achieve state of the art*. Retrieved from Microsoft: https://www.microsoft.com/en-us/research/blog/microsoft-vision-model-resnet-50-combines-web-scale-data-and-multi-task-learning-to-achieve-state-of-the art/#:~:text=Microsoft%20Vision%20Model%20ResNet%2D50%20is%20a%20large%20pretrained%20vision,Multimedia%