HAWASSA UNIVERSITY

INSTITUTE OF TECHNOLOGY

FACULTY OF INFORMATICS

DEPARTMENT OF COMPUTER SCIENCE

POSTGRADUATE PROGRAM

ENSET DISEASE DETECTION AND CLASSIFICATION USING DEEP-LEARNING TECHNIQUES

BY: ENDASHAW NIGUSE ASTATEKE

MAY 2024

HAWASSA, ETHIOPIA

ENSET DISEASE DETECTION AND CLASSIFICATION USING DEEP-LEARNING TECHNIQUES

ENDASHAW NIGUSE ASTATEKE

THESIS SUBMITTED TO HAWASSA UNIVERSITY,

INSTITUTE OF TECHNOLOGY,

DEPARTMENT OF COMPUTER SCIENCE,

SCHOOL OF GRADUATE STUDIES,

HAWASSA, ETHIOPIA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF MASTER OF SCIENCE IN COMPUTER SCIENCE.

Major advisor: Degif Teka (PhD)

Co-advisor: Wogene Alemayehu (MSc)

May 2024

DECLARATION

This thesis is my original work, which I hereby certify has not been submitted for credit toward a degree at any other university. I have duly recognized all of the sources of the materials used in this thesis.

Name:_____ Signature _____ Date 25/06/2024

I have given my approval as the thesis advisor for this MSc thesis to be submitted for examination.

Name: Dr. Degif Teka_____

Signature: _____

Place and Date of Submission: _____

ADVISORS APPROVAL SHEET-I

This is to certify that the thesis entitled "Enset disease detection and classification using Deep Learning Techniques" submitted in partial fulfillment of the requirements for the degree of Master's with specialization in Master's in Computer Science, the Graduate Program of the Department of Computer Science and has been carried out by Endashaw Niguse Astateke ID. No. "GPCoScw/0009/12", under our supervision. Therefore, we recommend that the student has fulfilled the requirements and hence hereby can submit the thesis to the department.

Dr. Degif Teka	_____	25/06/2024
Name of Major Advisor	Signature	Date
Wogene Alemayehu	_____	25/06/2024
Name of Co-Advisor	Signature	Date

EXAMINORS' APPROVAL SHEET

SCHOOL OF GRADUATE STUDIES

HAWASSA UNIVERSITY EXAMINORS' APPROVAL SHEET

(Submission sheet -II)

We, the undersigned, members of the board of examiners for the final open defense by "Endashaw Niguse Astateke", have read and evaluated his thesis, entitled "Enset Disease Detection and Classification Using Deep Learning Techniques," and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirements for the master's degree in computer science.

Dr. Degif Teka	_____	25/06/2024
Name of Major Advisor	Signature	Date
Wogene Alemayehu	_____	25/06/2024
Name of Co-advisor	Signature	Date
Zemenu Haile	_____	25/06/2024
Name of Internal Examiner-I	Signature	Date
Gezahegn Gaje	_____	25/06/2024
Name of Internal Examiner-II	Signature	Date
Dr.Eng. Mesay Adinew		25/06/2024
Name of External Examiner	Signature	Date
_____	_____	_____
SGS Approval	Signature	Date

Final approval and acceptance of the thesis is contingent upon the submission of the final copy of the thesis to the School of Graduate Studies (SGS) through the Department/School Graduate Committee (DGC/SGC) of the candidate's department.

Stamp of SGS Date: _____

Remark

- Use this form to submit the thesis with MINOR CORRECTION suggested by the examining board

- 3 copies

Acknowledgment

I want to express my sincere gratitude to my thesis advisor, Dr. Degif Teka, for their guidance, support, and valuable feedback throughout the research process. Their knowledge and expertise have been instrumental in shaping this thesis and I am truly thankful for their dedication and encouragement.

I would like to thank Hawassa University for providing the necessary resources and support for this project. The academic environment and research facilities have been crucial in the completion of this thesis.

I appreciate the relevant information input, sample photos, and categorization of the data that Firdu Azerefegn (PhD), experts from Hawassa University Agriculture College, protection laboratory officers, and Gedeo and Sidama Agriculture extension workers provided.

Finally, I want to thank my friends and family for their continuous encouragement and support during my academic career. Their love and devotion have been a never-ending source of inspiration and motivation.

Thank you to everyone who has contributed to the completion of this thesis. Your support and guidance have been invaluable, and I am truly grateful for all that you have done.

Table of Contents

Contents Pages

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

List of Abbreviations and Acronyms

Abbreviations and Acronyms	Meaning
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
DBN	Deep Belief Network
DL	Deep Learning
SNNPR	Southern Nation Nationalities and People Regions
PCR	Polymerase Chain Reaction
DNA	Deoxyribonucleic Acid
ELSV	Enset Leaf and Stem Strek
OL	Oligoclonal Antibody
HCI	Human and Computer Interaction
ReLu	Rectifier Linear Unit
RGB	Red, Green, and Blue
VGG	Visual Geometry Group
KNN	K-Nearest Neighbor
AI	Artificial Intelligence
CPU	Central Processing Unit
VS	Visual Studio Code
IOS	iPhone Operating System
CNTK	Microsoft Cognitive tool kit
ENN	Epistemic Neural Network
CMYK	Cyan, Magenta, Yellow, and Black
HSV	Hue, Saturation, and Value
GD	Gradient Decent
SGD	Stochastic Gradient Decent
CA	Classification Accuracy
EnsetDNet	Enset Diseases Neural Network (Developed Model)

ABSTRACT

Ethiopians, especially those in Sidama and Central Ethiopia, are the main consumers of enset. Enset is thought to be a staple food source for 20 million people in Ethiopia. Ethiopians mostly employ enset plants as a staple food crop. For many Ethiopians, the plant's root and stems are a major source of energy due to their high fiber and carbohydrate content. Typically, stems are picked, cleaned, and fermented to produce kocho and bulla, which are food items that are similar to bread.

This research investigates the application of deep learning techniques, specifically Convolutional Neural Networks (CNNs), for the detection and classification of diseases affecting Enset leaves and stems. By employing sophisticated image processing tools and methodologies, the study aims to improve the accuracy and efficiency of disease identification in Enset plants. Experimental findings underscore the effectiveness of the proposed CNN models in achieving notable accuracy rates in disease detection, showcasing the potential of deep learning in revolutionizing agricultural practices. The study not only emphasizes the importance of advanced image processing in agricultural contexts but also underscores the necessity for further research in crop disease detection to enhance agricultural sustainability and productivity.

We collected from the Central Ethiopia Region (Wonago and Dilla) and the Sidama Region (Hawassa Zuria, Boricha, Yirgalem, and Aletawondo) to use a 5,000 image dataset. There are 1000 images in each class. A total of 700 training, 200 validation, and 100 testing images were chosen. We used pre-trained models, MobileNetV3Small and EfficientNetB7, to compare the results with the newly developed model. Bacterial wilt, Mosaic Virus, Bacterial leaf spot, Insect pest, and Healthy Leaves are the disease classes. The Nadam optimizer, 32 batch sizes, 65 epochs, and 0.001 learning rate are the selected hyperparameters. The model was stable at epoch 65 and has an accuracy rate of 99.30%. EfficientNetB7 and MobileNetV3Small, the pre-trained models, have accuracy rates of 95.32% and 93.08%, respectively. The developed model's output has a high degree of accuracy in identifying and classifying diseases in Enset leaves.

Keywords: Computer vision, Deep Learning, Convolutional Neural Network, Enset and Enset Leaf Diseases.

CHAPTER ONE

1. Introduction

This chapter introduces the research by presenting a comprehensive overview of the study area, as well as the background of the study. It defines the motivation for conducting this research, including a statement of problems, research questions and hypothesis, objectives, and scope of the study. Furthermore, it highlights the significance of this study and outlines the structure of the thesis.

1.1. Background of the study

About 20 million people primarily in southern Ethiopia use enset as a staple or co-staple food crop, making it a significant source of food. Several enset cultivars of land can be found throughout Ethiopia's various administrative zones, which vary greatly in terms of altitude and agroclimatic zones. [1]. Enset (Enset ventricose (Welw.) Cheesman) is a major multi-purpose crop in Ethiopia, which has been identified as the center of origin and diversity of enset. Enset is an endemic root crop plant to Ethiopia that is perennial, herbaceous, and has long, broad leaves. It is a member of the Musaceae family. Enset is a versatile root crop with almost all of its parts having some use. In Ethiopia, enset is used for fiber, food, medicine, and fodder. Over 20 percent of Ethiopians, especially those living in the south and southwest, depend on this crop for their traditional basic food security. Enset is grown and distributed at altitudes between 1600 and 3000 meters above sea level with an average annual rainfall of 1100 to 1500 mm and it is chiefly propagated vegetatively [2].

Reduced rural poverty, regenerated food systems, restored degraded lands, overcome water scarcity, enhanced adaptation and mitigation to climate change, and biodiversity protection are all possible outcomes of agroforestry[3].

A healthy Enset leaf is typically green, without discoloration, spots, or physical damage. It indicates that the plant is thriving and receiving proper care and nutrients. Healthy leaves are important for the overall growth and development of the Enset plant, as they are responsible for photosynthesis, which is essential for the plant's survival.

In the country's main Enset-growing regions, bacterial wilt, Insect pest, Bacterial Leaf spot, Mosaic virus, and mealybug infections are common and can, in severe circumstances, result in 100% destruction of agricultural fields.

Currently, bacterial wilt and enset mealybugs have infected over 80% of Enset farms, and no Enset clone has been identified that is resistant to bacterial wilt[4].

Enset (Ensete ventricosum [Welw.] Cheesman) is the source of the staple food of Sidama people of southwestern Ethiopia. Enset is a drought-resistant crop that is native to East Africa and is extensively dispersed throughout sub-Saharan Africa. It is not widely recognized as a food plant outside of Ethiopia, where millions of people, especially in the Sidama Region and Southern Ethiopia, depend on it for their food[5].

In the world of agricultural need, the automatic detection of plant diseases using plant leaves is of major importance. Additionally, the early and timely detection of plant diseases improves agricultural production and quality. Farmers in most agricultural countries lose significant amounts of money each year to crop diseases[6].

The symptoms of plant diseases can be observed on leaves with the naked eye, however, the complexity of the process is fast increasing. Even experienced agricultural specialists and plant pathologists may frequently fail to correctly diagnose specific diseases due to this complexity, the vast number of cultivated crops, and their ongoing psychopathological issues. As a result, they may draw incorrect conclusions and overlook important solutions. Professional farmers could benefit significantly from an automated system that can recognize diseases of plants based on visual symptoms and the appearance of the plant. Farmers will find this to be a helpful strategy that alerts them at the proper moment to prevent the disease from spreading over a broad area[7].

Leaf and stem Disease Identification has been a key problem in the field of agriculture for a long time. Plant disease prediction accurately depends upon three factors host, environment, and pathogens. Efficient design of Plant Disease Detection systems requires a comprehensive understanding of their capabilities and the methodologies employed in existing disease identification systems. Academic research in Plant Pathology demonstrates the rapid advancement in the field of Plant disease identification.

Startups have started developing Smartphone applications that use image-based algorithms to diagnose plant disease.

Moreover, they also provide information on how to take care of plants efficiently. However, such algorithms still lack the power to analyze plants in different lighting conditions [8].

Nowadays, the advent of smartphone technology has made a significant contribution to the identification of plant diseases, and it has been technically implemented to improve image-processing techniques from simple photography to enset plant leaf. Diagnosis can lead to immediate interventions to reduce the impact of plant diseases on food supply. Currently deep learning (DL) techniques are being used in various agricultural sectors. Currently, it is very important in the detection and classification of plant diseases in their early stages.

Deep learning is a new trend of machine learning with cutting-edge results in many fields of research, including the field of computer vision and bioinformatics. Differing approaches to deep learning are currently used for plant disease detection and the most common are Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Belief Network (DBN), etc. The ability to use raw data directly without handmade materials is the primary advantage of deep learning[9].

Bacterial, fungal, and viral infections have a significant impact on plant health and introduce diseases that affect the growth of produce. In addition, the over-reliance on fungicides and pesticides to remedy this issue is not only costly but has a considerably negative impact on the environment. Early detection and targeted treatment of plant diseases are crucial in assisting farmers to implement appropriate measures for the preservation of affected plants.

Under the concept of deep learning, a deep feature is the consistent response of a unit in the hierarchical model to an input. Convolutional Neural Networks (CNN) have multiple hidden layers and convolution core layers that deepen the network architecture. Features extracted from a CNN network are called "deep features." Deep features are the properties of the data obtained by passing it through a multi-layered artificial neural network and subjecting it to convolution processes[10].

Therefore, a study on the disease incidence and morphology of fungal, bacterial, viral, and nematode pathogens disease in Ethiopia is valuable for gaining new information used in the planning of early disease prediction.

In this study, Convolutional Neural Network (CNN), which is the most popular deep learning method, will be used for the detection and classification of various enset leaf and stem diseases based on the enset leaf and stem image from the farm and augmentation techniques will be used.

Figure 1.1 Example of healthy Enset leaf and Diseased Enset leaf

1.2. Statement of the Problem

Enset is a key staple food and income generator in many parts of Ethiopia, but the plant is vulnerable to various leaf diseases that can negatively impact crop yield and quality. Current methods for detecting and classifying Enset leaf and stem diseases rely on expert manual inspection, which is time-consuming, subjective, and may result in misdiagnosis. There is a need for an automated and efficient system for detecting and classifying Enset leaf and stem diseases, to improve disease management strategies and ensure the sustainability of Enset production. Deep learning techniques, such as convolutional neural networks, have shown promise in various plant disease detection and classification tasks, but their applicability to Enset leaf and stem diseases has not been thoroughly investigated. Therefore, the problem statement is to develop a deep learning-based system for the automated detection and classification of Enset leaf and stem diseases, to provide accurate and timely diagnosis to farmers and researchers. Among the major problems in Enset disease are the cost, time, and labor of the agricultural sector.

There is limited research on the impact of diseases on the yield and quality of Enset. Recent research has indicated that Convolutional Neural Networks (CNNs) possess the capability to detect and categorize plant leaf diseases. However, the application of CNNs for the detection and classification of Enset leaf and stem diseases has yet to be fully explored. Therefore, a comprehensive examination of CNNs for detecting and categorizing Enset leaf and stem diseases is necessary to fill this research gap. This exploration should include the gathering of enset leaf and stem disease samples and the creation of CNN models to detect and classify them, as well as assessing and comparing the performance of these models with existing methods for disease detection and classification. Deep learning is a cutting-edge image-processing method that is still relatively new but produces reliable results [11].

“Dr. Alemayehu Chala and Dr. Firdu Azerefegne who are plant pathologists from Hawassa University College of Agriculture recommended doing a computer-assisted algorithm in the study of enset leaf and stem diseases in Sidama and SNNPRs research area and farms.” There is no way for farmers to diagnose and classify early enset leaf and stem diseases. Getting some advice from an expert takes time and most of the time it is more than human accuracy. This research will use Convolutional Neural Network (CNN) to identify and classify enset leaf and stem diseases. In these specific areas, local farmers lack the necessary technological tools to easily identify bacterial and insect pests and viruses diseases. The researcher decided to do this study because it saves time for farmers, increases productivity, and reduces losses.

The problem statement for this study is to develop a deep learning-based system that can detect and classify diseases in enset plants accurately and efficiently. The system should be able to differentiate between different diseases such as Enset bacterial wilt, Enset Bacterial leaf spot, Enset Mosaic Virus, and Insect pests, and provide real-time feedback to farmers for timely intervention. This will help improve the overall health and productivity of enset plants and ensure food security for communities relying on this crop.

1.3. Motivation

The motivation behind the research stems from the urgent need to address the devastating impact of plant diseases on enset crops, an important staple food for many communities in Ethiopia. Dr. Alemayehu Chala, and Dr. Firdu Azerefegn, a plant pathologist who works closely with farmers, have seen first-hand the detrimental effects of these diseases on enset crops, leading to economic losses for farmers and food shortages for communities. Traditional methods of disease detection and classification are often time-consuming, labor-intensive, and may not always provide accurate results. Additionally, I interviewed farmers and agriculture workers in Wongao and Aletawondo, and they informed me that there are currently no technological instruments available to identify Enset diseases. Those issues inspired me to conduct this research.

1.4. Research Question

- How can we differentiate between healthy and diseased Enset leaves effectively?
- How do deep learning models accurately differentiate between different types of Enset diseases based on images of plant symptoms?
- What deep learning techniques can be applied to accurately detect and classify enset leaf diseases in agricultural settings?
- What are the key features or characteristics that deep learning models use to detect and classify Enset diseases?

1.5. Objectives of the Study

1.5.1. General Objective

This study aims to develop automated systems for the early detection and classification of diseases in Enset plants using the deep learning method and Improving the accuracy and efficiency of disease detection in Enset crops.

1.5.2. Specific Objectives

To achieve the main objective of this research, the following specific objectives are included in the study:

1. To collect a comprehensive dataset of Enset leaf and stem images with various diseases for training the deep learning model.
2. To preprocess and clean the collected dataset to improve the accuracy of the deep learning model.
3. To choose and implement a suitable deep learning architecture for Enset leaf and stem disease detection and classification.
4. To evaluate the performance of the developed deep learning model in terms of accuracy, precision, recall, and F1 score.
5. To compare the performance of the proposed deep learning model with existing pre-train methods for Enset leaf and stem disease detection and classification.
6. To provide recommendations for improving the accuracy and efficiency of Enset leaf disease detection and classification using deep learning techniques.

1.6. Scope of the Study

The focus of this study will be on the development of the Enset leaf and stem disease detection and classification model for determining whether a plant is healthy or not, using a variety of enset leaf and stem diseases. After the detection and classification of the disease, there may be a way to recommend the appropriate medicine for that disease, however recommending the appropriate treatment for the identified disease, and estimating the severity of the detected disease is beyond the scope of this thesis work. The research focuses on Enset leaf and stem diseases such as Bacterial, Viral, and Insect pest diseases. The proposed system mainly focuses on only the enset plant leaf and stem.

1.7. Limitation of the study

The study only addresses five classes. They include healthy, Bacterial leaf spot, bacterial wilt, mosaic virus, and insect pests. Moreover, the Enset leaf and stem disease is the only subject covered; the root and other components are not included.

1.8. Significance of the Study

This study will be of great benefit to researchers and farmers in the field. At the end of the study, this research can be applied to different enset farms:

- This study is a good opportunity for those who are engaged or planning to engage in the agricultural sector in our country.
- This study can be applied to other plant pathogens and classifications.
- It will be a source of information and solutions for the agriculture expert.
- Enset leaf disease detection and classification system development of this application will increase the effectiveness of disease-controlling mechanisms.
- Minimize the needs of experts: since we are developing an automatic system that uses a CNN to identify enset leaf disease, it will minimize the needs of experts in that area.

1.9. Organization of the Study

The following sections of this thesis have been structured as follows: Chapter two includes a comprehensive literature review on the Enset plant domain, including Enset Bacterial and fungus diseases, machine learning, deep learning, and relevant studies. This chapter concludes with a summary of studies directly relevant to this thesis. Chapter three briefly discusses the methodologies employed for this study, such as data collection methods, materials and tools used, model selection, architectural design and Proposed model design of the EnsetDNet model, and evaluation techniques. Chapter |Four presents experimental parameters for detecting and classifying enset plant leaf disease using images of infected and healthy enset leaves and stems. Additionally, various experimental results are presented with detailed discussions. Finally, Chapter Five includes the conclusion and recommendations for further research.

CHAPTER TWO

2. Literature Review

This chapter provides a review of both conceptual and related literature. The conceptual review covers the characteristics of enset and its associated leaf diseases, as well as an overview of the deep learning algorithm CNN and pre-trained models, along with an examination of performance evaluation metrics. The related literature section critically analyzes articles and research works that are relevant to the study to identify gaps in knowledge.

2.1. Literature Review

The study aims to comprehend the application of image processing and convolutional neural networks (CNN) in resolving relevant issues in previous research by analyzing various literature sources such as books, the internet, and journals. The review will primarily focus on the most recent and current literature. Identification of gaps in previous solutions will be conducted to gather inputs for the proposed solution. As such, this investigation and analysis of all possible references and journals related to the research is a crucial component.

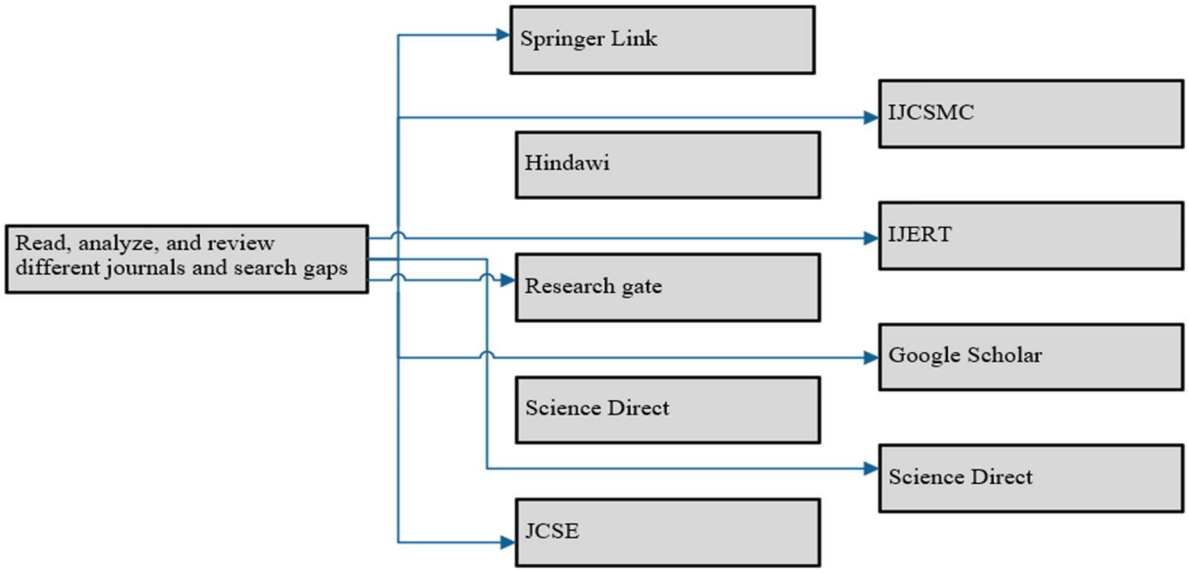

Figure 2.1 Digital Library for the research

Literature Review to understand the subject matter, the kinds of literature related to the work that we are going to develop will be reviewed.

This will help us to summarize all related works that are relevant to this work. Since this work is on developing a system that identifies enset disease, we will look at related topics that deal with Agro-industrial disease detection and classification mechanisms through computer vision.

2.2. Enset

The Enset plant is widely cultivated in the Sidama, South Ethiopia, South Western Ethiopia and Central Ethiopia Region (previously named SNNPR). Enset serves as a staple food crop for the Sidama community and holds a significant cultural and livelihood impact. The primary purpose of cultivating enset is for its starchy stem, which is harvested and processed into a traditional food known as "Kocho". Kocho is integral to the Sidama diet and is often eaten with various other dishes. The Sidama people have developed unique farming techniques specific to enset cultivation. Typically grown in family gardens, enset plants demand substantial time and effort for their care. Although these plants take several years to mature, they ultimately provide a reliable source of food for households. Apart from its culinary uses, enset possesses practical applications such as ropemaking from its leaves, basket-weaving, thatch production for roofing, alcohol production from the starchy pulp, and fabric weaving using fibers from the stem. Overall, within the Sidama region, enset bears immense cultural, nutritional, and economic significance by deeply intertwining with the lives of its people and playing a vital role in ensuring their food security and preserving traditional practices.

Enset, an important food crop cultivated in Ethiopia, plays a crucial role in ensuring food security, particularly in the Sidama, southern and southwestern regions of the country. The demand for the crop is increasing throughout the country. The overall production is decreasing due to the impact of enset bacterial wilt, Enset Mealy bug, Bacterial leaf spot, Bacterial blight, and crown rot.

The use of resistant types and clones, cultural practices, and clean planting materials are some of the strategies for managing disease. Over the past 50 years, various researchers have screened varieties against the disease at the open field level, producing results that have generated controversy[12].

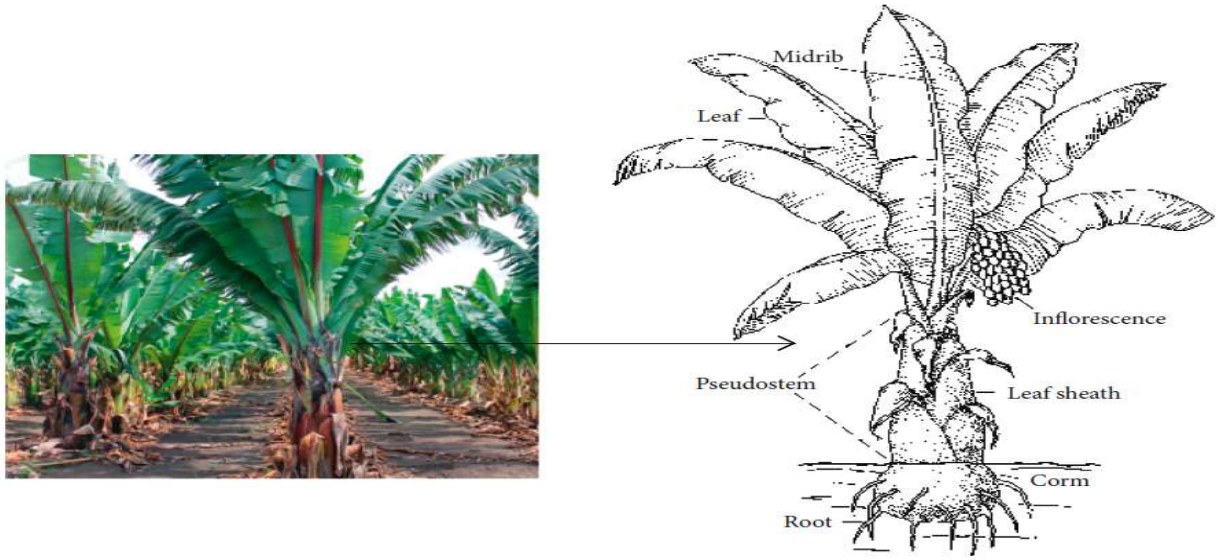

Figure 2.2 Enset plant and its part

Enset is a strategic food reserve that is ideally suited to bridge food shortage, a common phenomenon in Ethiopia [8]. Outside Ethiopia, the inflorescence is eaten as an enjoyment in Malawi and the flower bud is consumed as a boiled vegetable in many parts of Southeast Asia [29]. The type and number of enset clones cultivated in southwestern Ethiopia vary from place to place. Large numbers of clones are cultivated for multiple purposes, under different site and climate requirements. In each area, different clones are grown for different purposes [24]. The results show that enset products partially or completely fulfill the various livelihood needs of farmers. In areas where enset is an Indigenous crop, it is the major source of human food supplemented with legumes and animal products. The cultivation of enset has experienced significant growth in both area coverage and production. It has been reported that the annual production of enset foods exceeds 2 million tons. Enset crop loss is infrequent, and production is high as compared to other crops. Three main food items are prepared from enset: *Kocho* (fermented product from the corm and pseudostem, Figure 2), *Bulla* (dehydrated product of the juice from the decortication of the pseudostem and grating of corm), and *Amicho* is the stripped corm of younger plants of enset, boiled and consumed [13].

Although practically all of the plant's parts can be used for a variety of other reasons, Enset is primarily grown as a source of nutritious human food. A food item known as "Kocho" is produced locally when the starch contained in the pseudostem, corm, and stalk of the Enset flower is scraped and typically fermented.

"Amicho," a stripped corm of a younger plant of Enset, is another food type made from the plant that is boiled and eaten like a sweet potato, cassava, or Irish potato). Before it reaches maturity, a young Enset plant can be pulled up and its corm boiled for instant consumption in times of food scarcity. specific infections can also be cured with the corms of specific landrace plants. The perennial crop is used as a Source of staple food for humans in many areas of Southern Nations, Nationalities, and Peoples' Region and some parts of Oromia [14].

2.2.1. Types of Enset leaf diseases and Healthy leaf

Plant (Ensete ventricosum). Enset is an important food crop in Ethiopia, where it is widely grown and used as a staple food. Several bacterial diseases affect enset, leading to significant yield losses. Some of the common bacterial diseases include:

2.2.1.1. Bacterial Wilt

Enset bacterial wilt, which is caused by Xanthomonas campestris pv. Musacearum, has long been a major obstacle to enset production in Ethiopia. In addition to this, other biotic and abiotic factors also negatively affect productivity. It is known that Musacearum can infect any tissue in an enset or banana and then spread throughout the body. Wounded roots, pseudostems, and leaves are the entry points for the infection into the host. If the infection happens in the lower portions of the plant (rhizome or pseudostem), the bacteria may go upward through the vascular tissues, or if the infection occurs through the inflorescence, the bacteria can spread downward.

Once it has entered the plant, the pathogen is able to establish and cause disease rapidly provided the bacterium is received on a receptive infection court (wound, recently dehisced bract, etc.) in a viable state and the environment is conducive to infection and tissue colonization. In regions suitable for banana and enset cultivation, environmental conditions are likely to be conducive to disease development. After the establishment of the disease in the plant, it can spread by different mechanisms to new fields [15].

The disease attacks almost all varieties of commonly grown banana cultivars and enset clones reported that Xcm might have the potential to infect maize, sugarcane, and sorghum; therefore, these plants may act as alternative hosts and reservoirs for infection. Pathogenicity tests were conducted to determine the potential host range of the pathogenic Xcm and the response of different plant species, including wild and cultivated enset, Canna species, and cereal crops (maize, sorghum, and finger millet).

The results indicated that the pathogen may also have other possible hosts in addition to banana crops and cultivated enset. Alemayehu Chala describes common symptoms of bacterial wilt, including yellowing from the point to the margin of the infected leaf, lesions saturated in water along the midrib of the infected leaf, necrosis, and browning of the leaf. Finally turning wilting and dryness was observed in wild enset, canna spp., maize, sorghum, and finger millet varieties after two to four weeks after inoculation depending on the plant tested [15]. Figure 2.3 shows of infected leaf by bacterial wilt.

Figure 2.3 Picture of Enset Bacterial Wilt

2.2.1.2. Enset Insect pest

In the southern Ethiopian enset growing regions, the enset root mealybug (Cataenococcus enset Williams and Matile-Ferrero, 1999) is a major pest. In Wonago, Ethiopia, it first appeared in 1988. It is mostly found in mixed agricultural systems where fruit trees and other cash crops like coffee are cultivated side by side, below 2,000 meters above sea level (masl). Mealybug infections cause endemic plants to grow more slowly and have dried out their lateral leaves. All plant age groups are attacked by the insects, although symptoms are most severe on enset plants that are 2 to 4 years old. Mealybugs with enset roots are found on roots and corms. However, as portions of the roots dry out and die during severe droughts, mealybugs usually migrate towards the corm[16].

The enset root mealybug (Cataenococcus enset) is the most significant and common insect pest of Ethiopia's enset growing areas. A variety of biotic pressures are impacting enset production and productivity in the country's production areas. This root mealybug is a generic term for several Pseudococcidae feeding on underground plant parts[17]. Figure 2.4 shows of infected leaf by Insect pest diseases.

Figure 2.4 Picture of Enset Insect Pest

2.2.1.3. Bacterial leaf spot

This disease is caused by Xanthomonas campestris. It is characterized by the development of small, water-soaked spots on the leaves, which later turn yellow and then brown. The spots may coalesce, leading to the death of the whole leaf. The cause of this disease can be recognized as the bacteria Pseudomonas syringe. It is characterized by the development of small, water-soaked spots on the leaves, which later turn brown and necrotic. The spots may enlarge, causing the leaves to curl and die. The bacteria may also infect the stems and fruits of the plant, leading to significant yield losses. On leaves, bacterial spots appear as small (less than ⅛ inch), occasionally water-soaked (i.e., wet-looking) circular patches. Spots may start yellow-green but as they age, they may turn brownish-red. In addition to widespread leaf yellowing and leaf loss, serious diseases can also cause [18]. Figure 2.5 shows infected leaves by Bacterial leaf spot diseases.

Figure 2.5 Picture of Bacterial leaf spot

2.2.1.4. Enset Virus

A virus that resembles the badnavirus previously reported and has bacilliform particles has been isolated from enset with leaf streak disease and further characterized utilizing molecular tools such as inverse PCR, rolling circle amplification, and nucleotide sequencing.

Banana streak virus OL antibodies were used to paint the virus particles at a medium level, revealing their serological association. The virus is most closely related to the Sugarcane bacilliform Guadaloupe D virus, which was recently identified from sugarcane and has an overall nucleotide identity of 73.6%, according to sequence analysis of its circular dsDNA genome.
Enset leaf and stem streak virus (ELSV) is the proposed name for the virus, which is sufficiently different from other species in the genus Badnavirus based on molecular criteria.[19]. Figure 2.6 shows of the infected steam by the Enset mosaic virus disease.

Figure 2.6 Picture of Enset virus.

2.2.1.5. Healthy Enset Leaves

Healthy leaves are those that remain free from any form of disease and display normal function, structure, development, and synthesis during the process of photosynthesis. In a research setting, a healthy leaf will not exhibit any symptoms on its upper or lower surfaces and should maintain its typical green coloration. Leaves serve as a primary source of carbohydrates, starch, dietary fiber, phenolic compounds, high-quality proteins, and other bioactive nutrients. It is a gigantic leafy monocarpic evergreen perennial root crop and the main food source mostly for the southwestern part of Ethiopian highlands [20]. The healthy leaf samples are shown in Figure 2.7.

Figure 2.7 Picture of Healthy Enset leaves

2.3. Data Science

Data science is an interdisciplinary field that involves the use of statistical analysis, machine learning, and other advanced techniques to extract insights and knowledge from data. It involves the application of computational, mathematical, and statistical principles to identify patterns and trends in large sets of data and to convert this raw information into actionable insights. Data scientists use various tools and technologies to gather, process, interpret, and communicate data, and they work across a wide range of industries and domains, such as healthcare, finance, marketing, and logistics, among others. The field of data science is constantly evolving, as new techniques and technologies emerge, making it an exciting and challenging career choice for anyone interested in working with data.

2.4. Computer Vision

Computer vision is a discipline within the realm of artificial intelligence that focuses on teaching computers to comprehend and analyze visual data. By utilizing digital images captured by cameras and videos, as well as employing deep learning models, machines can effectively recognize and categorize objects, subsequently responding accordingly to their visual observations. Computer vision is a combination of image processing and pattern recognition. The output of the Computer Vision process is image understanding. Development of this field is done by adapting the ability of human vision to take information. Computer Vision is the discipline of extracting information from images, as opposed to Computer Graphics. The development of computer vision depends on the computer technology system, whether about image quality improvement or image recognition[21]. The main objective of Computer Vision is to develop algorithms and techniques for extracting information and data from images. On the other hand, Image Processing focuses on the implementation of computational transformations, including sharpening and contrast enhancement, among others. It also has similar meaning and sometimes overlapping with In Human and Computer Interaction (HCI). HCI coverage focus on full design, interface and all aspects of technologies related to the interaction between human and computer. HCI is then developed as a separate discipline (which is the field of interdisciplinary science) which discusses the interrelationships between human-computer mediated by technology development including human aspects. Functionally, computer vision and human vision are the same with the aim of interpreting spatial data, i.e., data indexed by more than one dimension. However, computer vision cannot be expected to replicate just like the human eye[21].

In the field of image processing, "computer vision" is becoming a more fashionable term. Recognition of things automatically is becoming more and more important with the development of computer vision applications. The computer vision community has benefited from deep convolution neural networks (Deep CNNs), which have shown outstanding results in a variety of domains, including speech recognition, natural language processing, object detection, video processing, and picture classification and segmentation[22].

2.5. Deep Learning

The neural network architecture is used in most deep learning techniques, and it is for this reason that deep learning models are known as deep neural networks. The term “deep” refers to the many hidden layers in the network. A traditional neural network can only have up to two hidden layers, but a deep neural network can have close to 150 of them. Deep learning models use large data sets called the training data set and neural networks to learn features from the data, and due to this, there is no need for the engineer to manually extract features from the data to train the machine.

One of the most common types of deep networks is called the convolutional neural network, or CNN. This type of network is well suited to process two-dimensional data, such as images. A CNN combines the features it has learned with the features in the input data and uses the two-dimensional convolutional layers to process information. A CNN eliminates the need to extract features from data manually.

With the rapid development of deep convolutional neural networks (CNNs), significant progress has been made in computer vision in several areas including object identification, semantic segmentation, image classification, and image super-resolution reconstruction. The CNN has superior features for autonomous learning and expression, and feature extraction from original input data can be realized by means of training CNN models that match practical applications [33].

This means that the engineer does not need to classify the features or identify the images to assist the network in categorizing them. The CNN extracts the features directly from the images or input data. The engineer does not train the data to choose some features from the information provided to it.

The CNN learns what features it needs to look for when the engineer feeds it the training data set. It is for this reason that computers or models with CNN is used to classify objects. A CNN learns to identify the different characteristics and structures of an image. It does this by using the many hidden layers within the network. Every hidden layer identifies complex features of the image, and the complexity increases as the hidden layers increase. For example, the initial hidden layer can detect colors in the image, while the last layer can identify different objects in the background.

Deep learning is an advanced class of machine learning algorithms that have utilized several sequential layers. Each layer uses the output of the preceding layer as input. The learning process can be unsupervised, supervised, or semi-supervised. LeCun et al define deep learning as a 13-representation learning method[34]. Deep learning does not have to divide the feature extraction and the classification because of the model automatically extracts the features while training the model. It is used in many research areas such aces as image processing, image restoration, speech recognition, natural language processing, and bioinformatics. Deep learning is a subpart of Machine Learning, for individual definitions.

- Artificial Intelligence provides an intelligent system
- Machine Learning is automatically through experience and makes predictions on data
- Deep Learning is multiple layers of transformation

Input values are pass through this network of hidden layers until they eventually converge to the output layer. The output layer is used for predicting the result. What these small weights should be, we typically use an algorithm called backpropagation.

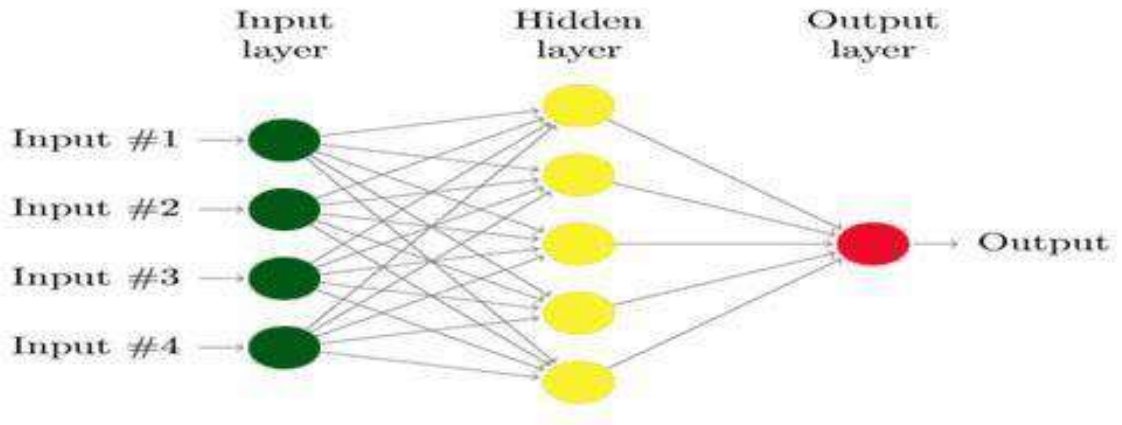

Figure 2.8 Neural network layer.

2.5.1. Role of Deep Learning in Image Processing

As all other machine learning techniques work in a way of taking the training database of that training, they process new input and input and take the decisions. However, as opposed to this deep learning works with neural networks and can make the conclusions of its own without the need for labeled training data. This method is useful for a self-driving car so that it can differentiate between a signboard and a pedestrian. Neural networks use algorithms that are present in the network side by side and the output of one algorithm is subject to the outcome of another algorithm. This creates a system that can think as a human being for making decisions. In addition, this makes a model, which is a perfect example of an artificial intelligence system.

2.5.2. Convolutional Neural Network (CNN)

A convolutional neural network is selected as a deep learning method in this study. CNN, which can easily identify and classify objects with minimal pre-processing, is successful in analyzing the visual image and can easily separate the required features with its multi-layered structure. It consists of four main layers: - convolutional layer, pooling layer, activation function layer, and fully connected layer. Fig 2.9 shows a general CNN architecture. Layers of CNN are the Convolution layer, Activation layer, pooling layer, Dropout Layer, fully connected Layer.

Convolutional Neural Networks (CNNs) were inspired by the visual system's structure, and in particular by the models of it proposed in. According to Neocognitron, when neurons with the same parameters are applied on patches of the previous layer at different locations, a form of translated invariance is acquired. This is the basis for the first computational models based on this local connectivity between neurons and on hierarchically organized transformations of the image. Yann LeCun and his collaborators later designed Convolutional Neural Networks employing the error gradient and attaining very good results in a variety of pattern recognition tasks [35].

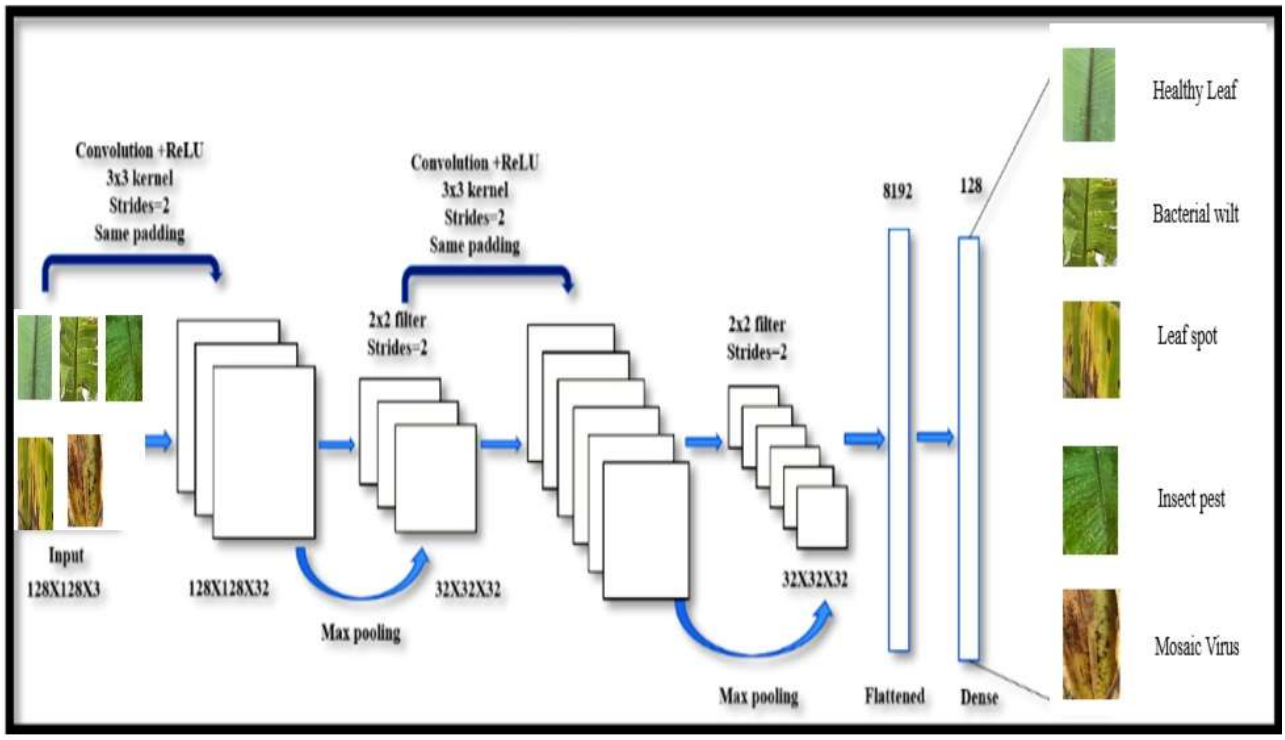

Figure 2.9 A general CNN architecture

i) Convolution Layer

The fundamental component of the CNN is the convolution layer. It carries the majority of the computational pressure on the network. This layer does a dot product between two matrices: the confined area of the receptive field is one matrix, and the other matrix is the set of learnable parameters, also referred to as a kernel. Compared to a picture, the kernel is more in-depth but less in space. This indicates that the depth fills up all three channels, yet the kernel height and breadth are spatially little if the image consists of three (RGB) channels.

The kernel creates the image representation of that receptive region by sliding across the image's height and width during the forward pass. This results in the creation of an activation map, a two-dimensional representation of the image that shows the kernel's response at each spatial location in the image. A stride is the name given to the kernel's sliding size.

If we have an input of size W x W x D and D out the number of kernels with a spatial size of F with stride S and amount of padding P, then the size of output volume can be determined by the following formula:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Equation 2.1 Size of output volume formula

ii) Parameters, which helps in adjusting CNN's performance

Stride defines the number of pixels by which we must move our filter over the image so that we can focus on a new set of pixels while doing convolution. Stride's value ranges from 1 to 3 depending upon the amount of loss that we can be accommodated during convolution. The amount of loss in the image increases with the increasing value of stride.

- Padding: - It is a process of adding zeroes around the border of the original image symmetrically. This helps us obtain the feature map output to be of a size as per our requirement. Commonly it is used to preserve the dimension of the image after convolution.
- Filter: - These are also called kernels. These may be of many types. Each filter increases the depth of the output generated after convolution. Therefore, if we are using three filters then the depth of the output will be three. The output of convolution depends on stride, Depth, and padding on three parameters. We must finely tune these parameters to obtain the desired output.

2.5.3. Image Processing

Image processing is a method that performs the analysis and manipulation of digitized images, to improve the quality of the image. The primary advantages of digital image processing techniques are their adaptability, recurrence, and accuracy in preserving the original data[23].

Image preprocessing is an essential process in the field of image processing and computer vision. Its primary purpose is to improve image data by enhancing certain features and suppressing unwanted distortions. The enhancement of features is application specific. When it comes to satellite sensor data, errors related to pixel geometry and brightness values are commonly present. In image preprocessing, these errors are rectified using suitable mathematical models, which can be either definite or statistical in nature.

Image preprocessing also includes primitive operations to reduce noise, contrast enhancement, image smoothing and sharpening, and advanced operations such as image segmentation [23].

The image is determined based on the size of the pixel neighborhood used to calculate the brightness of a new pixel.

preprocessing techniques can be categorized as,

- Pixel brightness transformations
- Geometric transformations
- Preprocessing techniques that use a local neighborhood of the processed pixel
- Image restoration that requires knowledge about the entire image

2.5.4. Image Preprocessing

An essential first stage in the processing of images, image preprocessing includes a set of procedures meant to get raw or unprocessed images ready for additional examination, deciphering, or enhancement. This crucial phase helps enhance the quality of images, mitigate noise, correct anomalies, and extract relevant information, ultimately leading to more accurate and reliable results in subsequent tasks such as image analysis, recognition, and classification [24].

Image Preprocessing is a crucial step in preparing images for further analysis or machine learning tasks. During image pre-processing, an image is altered to provide more information about the affected area of a leaf. With good visual interpretation, it presents leaf image information effectively [25].

2.5.5. Image segmentation

Digital image processing involves the utilization of computer algorithms to conduct image processing on digital images. Within the field of image processing, image segmentation stands as a significant and demanding process. The technique of image segmentation is employed to divide an image into meaningful portions that exhibit comparable features and properties. The main aim of segmentation is simplification i.e. representing an image into meaningful and easily analyzable way. Image segmentation is a necessary first step in image analysis[26].

Convolutional neural networks are the most popular machine learning methods at present. In CNN models, a set of kernels (or filters) are utilized to learn various characteristics, and these learnable kernel values are updated based on the value of the cost function. The convoluted output from each kernel is considered as the feature map and is passed to the subsequent layers [27].

2.5.6. Image Acquisition

Image acquisition is the process of obtaining digital images from a variety of sources such as cameras, scanners, and digital video recorders. The images may be obtained from digital or analog sources and the information is then stored in a digital form in the computer system. It also involves capturing images from physical objects such as paintings and documents. The data is then stored in a digital form in the computer system, allowing for further manipulation and analysis. Image acquisition is an important part of digital image processing, as it is necessary for any image processing task. Quantum Detector is that the most vital mechanism of image sensing and acquisition it relies upon the energy of absorbed photon getting used to promote electrons from their stable state to a better state above an energy threshold [28].

2.5.7. Feature extraction

To recognize different objects, we need to extract visual features which can provide a semantic and robust representation[29]. In feature extraction [30], the original feature space is converted to a more compact new space. All of the original features are converted into the new, reduced area via a smaller representative set, which replaces the original features rather than removing them. when the input data contains an excessive number of features, it becomes unmanageable to process. Consequently, the input data needs to be converted into a smaller set of features through transformation.

To perform this, CNN architecture was built with multiple overlying convolutional layers. Normally, a convolutional layer will contain convolution and activation functions that are non-linear in our case, and the utilization of ReLU and pooling process have been enabled [31].

2.5.8. Image Detection and Classification

Detection is the process of detecting or noticing something in the real world. So Detection in this study is used to refer to detecting if a enset leaf is infected with diseased or health. Classification refers to the task of labeling a category prediction given input data.

The method of classifying an image involves grouping pixels based on the data values they contain. For a pixel to be classified into a specific class, it must meet certain requirements in that class. If using the training data, the user was able to identify the classes of the data, the classes might be known; if not, the classes are unidentified.

In general, the process of image classification is to extract image features and then classify the extracted features. Therefore, how to extract image features and analyze image features is the key point of image classification [32].

2.5.9. Dimension Reduction in CNN

During convolution operation, the dimension of the input image is reduced after a filter matrix is convoluted over the input image. It also increases the depth of the image as this helps in identifying the regions in the image.
The following are definitions of different layers shown in the above architecture:

2.5.9.1. Pooling Layer

Pooling layers are a type of convolutional layer used in deep learning. Pooling layers reduce the spatial size of the input, making it easier to process and requiring less memory. Pooling also helps to reduce the number of parameters and makes training faster. There are two main types of pooling: max pooling and average pooling. Max pooling takes the maximum value from each feature map, while average pooling takes the average value. Pooling layers are typically used after convolutional layers in order to reduce the size of the input before it is fed into a fully connected layer.

2.5.9.2. Activation Layer

This layer mostly uses ReLu, softmax, and sigmoid as activation. Function ReLu is a function that is used to set all negative values to zero and keeps the positive value as it is. The convolution and pooling layer follows this step. In numerous fields such as computer vision, they are becoming the state of art achieving near-human or better performance. They sound interested but designing a CNN is a herculean task in itself. Till now there is no fixed formula for the design of CNN. Many researchers have come up with general suggestions such as. But they don't always hold, as the task-dependent as important on data as it does on the algorithm. CNN exploits the spatial hierarchical feature of data, extracting features, and helps classify them into different classes. This has led to the development of a stream of data augmentation and pre-processing to increase the data, as more data allows for the chance of better training and avoiding overfitting. This helps build models that are more robust to new samples as we try to make it more generalized to noise at the training phase.

2.5.9.3. Dropout Layer

The dropout layer is usually applied after the layer containing neurons in the fully connected network. The dropout layer is a regularization layer. It randomly drops out the weights of some neurons in every iteration so that the other neurons get more weightage in that iteration. Usually, people take dropouts of 20-50%. By using the dropout layer, normally the performance of a network increases 1 to 2%.

2.5.9.4. Fully Connected Layer

In a convolutional neural network (CNN), fully connected layers are among the most fundamental layer types. A fully connected layer is one in which every neuron is connected to every other neuron in the layer above, as the name indicates.

When using the attributes that have been learned by the preceding layers to create predictions, fully linked layers are usually employed at the end of a CNN.

The structure often takes the shape of a pyramid when viewed from the top down; the number of parameters in each layer gradually converges until the necessary number of classes is reached. Increasing the number of hidden units in the layer can increase the learning ability of the network, but there is a saturation of the increase in the accuracy of the network. There is no formulation of the units you choose as it is a hit-and-trial usually. These also pile up with the fully connected subset of the network. Most of the network in research usually performs well with the number of units in multiples of 64. Two to three layers networks are good if enough patterns are being passed to the network after the flattening of the convolutional layers.

2.5.10. Pre-train CNN Architecture

Pre-trained convolutional neural networks based on features including channel boosting, depth, multi-path, width, and attention are available. Most pre-trained models are created using large datasets, such as MNIST, CIFAR, and ImageNet.

2.5.10.1. Efficient Net B7

EfficientNetB7 is a pre-trained deep learning model that belongs to the EfficientNet family of models. EfficientNet models are neural networks that have been designed to be highly efficient in terms of computational resources while maintaining high accuracy on various computer vision tasks.

Pre-trained models like EfficientNetB7 have been trained on large datasets such as ImageNet, which allows them to learn rich feature representations that can be transferred to new tasks with minimal fine-tuning [36].

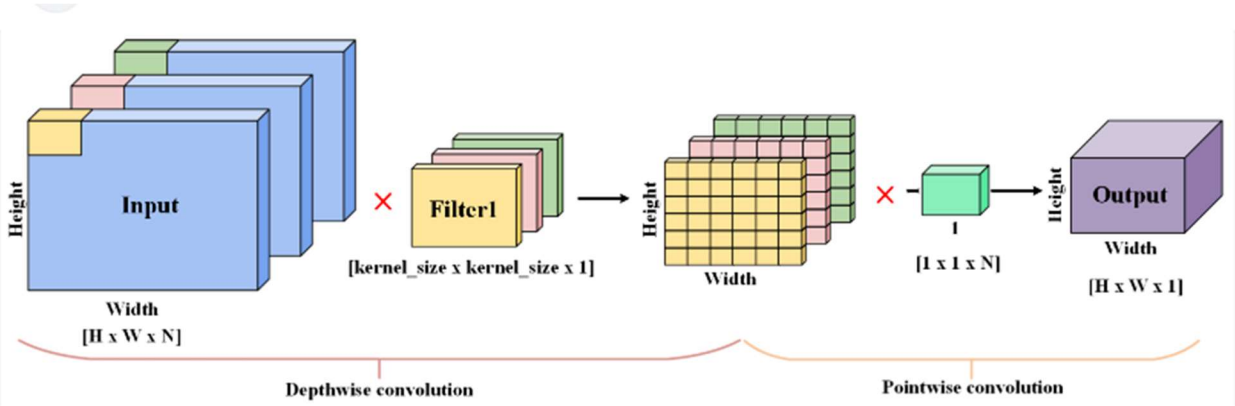

Figure 2.10 EfficientNetB7 General architecture

2.5.10.2.MobileNetV3Small

Utilizing convolutional neural networks, deep learning provides an invaluable tool for analyzing visual images. From image and video recognition to medical image analysis and language processing, the applications for this deep neural network are vast and varied.

MobileNetV3 is a convolutional neural network designed to be optimized for use on mobile phone CPUs using a hardware-aware network architecture search (NAS) combined with the NetAdapt algorithm, as well as further improved through novel architectural enhancements [37]. The MobileNetV3 architecture integrates several building blocks and components from previous versions, as presented in Fig 2.11.

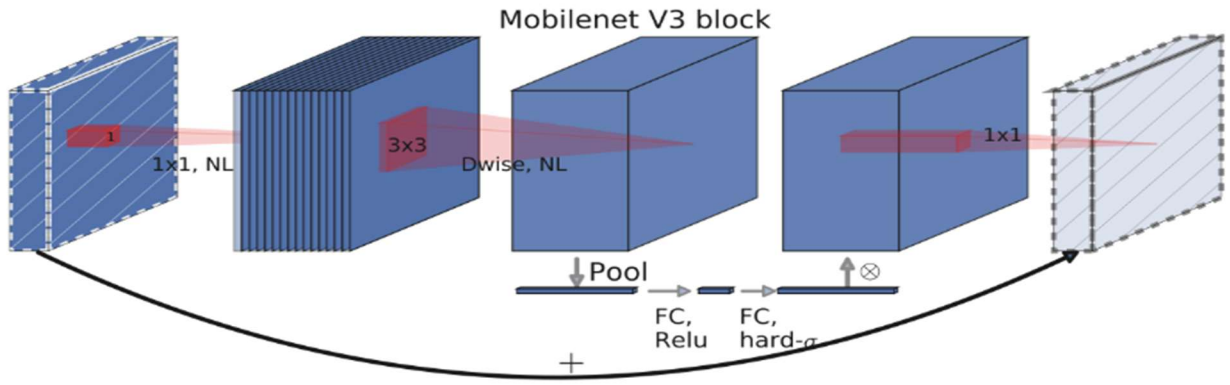

Figure 2.11 MobileNetV3 architecture

2.5.11. Optimization Techniques

Throughout the learning process, optimization techniques are used to minimize the objective function, also known as the loss function, to maximize production efficiency. Optimization algorithms are the foundation of a machine's experience-based learning capabilities. The goal of supervised learning is to find a function that approximates the underlying function based on observed samples. The initial stage is to identify a diverse range of functionalities, such as neural networks, that can represent the intended function. By minimizing a certain loss function, the function's parameter is found in the second phase. In the third phase, test error is the outcome of predicting unknown test data using the function discovered in the second step. The test error can be decomposed into representation error, optimization error and generalization error, corresponding to the error caused by each of the three steps [38].

2.6. Related works

The agriculture sector is concerned with quality and quantity management of their crop production, and crop disease detection is a major issue. Recent works on the use of deep learning to classification and identification plant leaf diseases in enset, early leaf, cassava, pepper and banana have been reviewed and discussed.

Krishnan VG, Divya V, Rao PV, Deepa J.[25] introduces an automated segmentation and classification model for detecting banana leaf diseases, leveraging image processing and deep learning techniques. While the study makes significant strides in automating disease detection in agriculture, a notable research gap lies in the optimization and further refinement of the segmentation and classification models.

Enhancing the algorithm's accuracy, sensitivity, and specificity could lead to more reliable disease identification outcomes, thereby improving the overall effectiveness of disease detection in banana plantations. Additionally, integrating real-time environmental data, such as climatic conditions and soil parameters, into the algorithm could enhance its predictive capabilities and potentially enable the development of an early warning system for banana diseases, addressing a critical need in agricultural sustainability. The researcher underscores the importance of early disease detection to mitigate crop destruction and highlights the transformative potential of advanced technologies like hyperspectral imaging and artificial intelligence in disease identification.

However, the study acknowledges the challenges associated with acquiring labeled data for training disease detection models, particularly in the early stages of infection. This limitation poses a significant hurdle in developing robust classification algorithms and points to a research gap that warrants further exploration. By addressing these gaps through the optimization of segmentation and classification models and integrating environmental data, future research can advance automated disease detection systems, contributing to enhanced agricultural productivity and sustainability.

Kishori Patil [7] present method that identifies whether an early leaf disease is healthy or infected. For disease detection and classification, image processing tools and the machine learning mechanism are proposed. Crop disease will be detected through various stages of image processing such as image acquisition, pre-processing of image, image feature extraction, feature classification, disease prediction and fertilizer recommendation detection of disease is important because it will may help farmers to provide proper solution to prevent this disease[39].

("Crop Diseases and Pest Detection using Deep Learning and Image ...") The proposed system has achieved an accuracy of 86.26% this is done based on particular properties between pixels in the image or their texture Then to classify the features which are represent the given image statistical analysis tasks are performed. Machine learning is used to compare image features. Even though the obtained accuracy is high, it is not sufficient to predict or differentiate between the health's of diseased leaves. In addition, the type of diseases was not identified. In machine learning, more intervention that is human is involved for getting results.

Selvaraj et al. [40] presents a significant contribution to the field of plant disease recognition, specifically in the context of banana crops. The researchers address a crucial research gap by introducing a novel approach that utilizes deep transfer learning methods for the automated detection of banana pest and disease symptoms in real field images. This gap in the literature pertains to the limited exploration of efficient and accurate techniques for identifying diseases on different parts of banana plants in real-time scenarios. Furthermore, the research conducted by Selvaraj et al. not only fills this gap but also demonstrates the potential impact of their findings on the agricultural sector. The study's focus on creating models that can detect and classify banana diseases with accuracy ranging from 70% to 99% showcases the effectiveness of deep learning techniques in addressing complex agricultural challenges.

The study introduces a novel method using deep transfer learning to automatically detect banana pest and disease symptoms in different parts of banana plants from field images. This approach offers a practical and applicable solution for classifying diseases in banana plants, setting it apart from existing methods for plant disease classification the research is limited to specific diseases, so it is better to extend to other banana diseases.

Tesage Yigbeta, Million Meshesha, and Muktar [4] aim to develop a model that can automatically identify diseases and pests in Enset plants using image processing and deep convolutional neural networks (CNN). The methodology involves various steps such as resizing images, segmenting, extracting features, applying pooling layers, using activation functions and dropout, and conducting classification. The model achieved an accuracy of 99.95% in predicting test datasets. Experimental results demonstrated high accuracy and low loss when different optimizers were utilized. Specifically, the VGG16 model accurately predicted all test dataset images with 99.98% accuracy and a loss of 0.36%.

The proposed model effectively detects Bacterial Wilt, root mealybug, and healthy Enset images with high accuracy while requiring minimal computational effort. However, it is important to note that the scalability and generalizability of the proposed model have not been explicitly addressed in this study. Additionally, there is no discussion on potential limitations or challenges when implementing the model in real-world Enset farming environments.

The research by Vandana Chaudhari and Manoj P. Patil [41] focuses on developing an automated system that utilizes ensemble machine learning to detect and classify diseases in banana plants. The research includes the methodology, experimental results, and the advantages of this method. However, there are some areas that require further attention.

Firstly, there is a lack of detailed discussion regarding the scalability and practical applicability of the proposed system. Investigating how the system can be implemented in real-world agricultural settings and its potential impact on crop management would be beneficial. Secondly, conducting a more comprehensive comparison with existing methodologies for plant disease detection would greatly benefit this research. A thorough analysis of the strengths and limitations of the proposed approach in comparison to other methods would provide valuable insights. Lastly, there is limited discussion on the limitations and potential future directions of the proposed approach in the document. Addressing these areas would enhance the overall depth and practicality of this research.

Kerim Karadag [42] used three different classification algorithms have been compared in detection and classification like Naïve Byes, Artificial Neural Network and KNN ruled based classification. Every technique has certain limitations. Those are used a limited dataset, and others focused on only two classes healthy or not healthy. The researcher uses only 84 image, which is very limited dataset and it is not applicable for other pepper specious.

[43]The proposed system is implemented with Keras (Tensor Flow as a backend) using Python programming language. The deep learning system that we developed is called BWENet. The model is trained for 100epochs, a batch size of 32, and a starting or initial learning rate of 0.001. Because we have trained in different parameters, we achieved better accuracy. Hyperparameter are selected based on empirical optimization. Convolutional Neural Network approach is used. Training accuracy and 90.53% testing accuracy, the total time to train the model is around 5:35 hour, and the weight file size for VGG19 250 is much higher than 18.4.

In the proposed model bacterial wilt disease detection and grading, the size of the model cannot influence the accuracy or loss of the model. There is some limitation in the research work, which only work on detection it is not classified into different stages and not differentiated by, and if there is some recommend which medication is used when we detect some diseases, but it is limited to only detection.

Table 2.1 Summary of related work

Author	Title	Methodology used	Accuracy	Gap
Kishori Patil	Leaf Disease Detection Using Deep Learning Algorithm	image acquisition, preprocessing of the image, extraction of features from the image, and classifying leaf diseases depends on image features that is color features, shape features, and texture features	86.26%	The type of diseases was not identified. In machine learning, more intervention that is human is involved in getting results.
Tesage Yigbeta Million Meshesha Muktar Bedaso	Enset(Enset entricosum) Plant Disease and Pests Identification Using Image Processing and Deep Convolutional Neural Network	1. Data Acquisition 2. Data Augmentation 3. Image Processing 4. Image data pre-processing 5. Segmentation 6. Feature extraction 7. Classification	99.97%	The obtained system accuracy is very high. This research limited only on Bacterial wilt and enset root mealy bugs.
Vandana Chaudhari and Manoj P. Patil	Detection and Classification of Banana Leaf Disease Using Novel Segmentation and Ensemble Machine Learning Approach	KNN, SVM, Naïve Bayes, and ensemble classification methods were applied	95 %	The document does not extensively discuss the limitations and potential future directions of the proposed approach. Addressing these gaps would enhance the overall depth and applicability of the research.

Kerim Karadag	Detection of pepper fusarium disease using machine learning algorithms based on spectral reflectance	The Wavelet Transformation method is used to obtain the feature vector including important information of the records and it allows time-scale analysis of stationary or non-stationary signals.	84	The researcher uses only 84 image, which is very limited dataset and it is not applicable for other pepper specious.
Desalegn Ashebir and Getahun Tadesse	BWENet: Enset Detection and Grading of Bacterial Wilt Using Deep Convolutional Neural Network	Deep Convolutional Neural Network is used	90.53	The researcher only limited on Bacterial wilt diseases and only it detects types of diseases there is no classification method mentioned in the research.
Krishnan VG, Divya V, Rao PV, Deepa J.	An automated segmentation and classification model for banana leaf disease detection	The hybrid segmentation called total generalized variation fuzzy C means (TGVFCMS) method is used for segmenting the affected area on the leaves.	93.45%	The research needs further exploration and optimization of the segmentation and classification models for banana leaf disease detection. Investigating the integration of real-time environmental data, such as temperature, humidity, and soil conditions, into the disease detection algorithm to create an early warning system for banana diseases will be valuable including in the research.

Selvaraj et al. P	AI-powered banana diseases and pest detection	Deep transfer learning methods for automated detection of banana pest and disease symptoms in field images. three different architectures were employed: ResNet50, InceptionV2, and MobileNetV1.	over 70%	The article presents a novel technique that uses deep transfer learning to automatically identify indications of diseases and pests in various banana plant sections using field images. Despite other plant disease classification techniques, this method provides a useful and usable means of categorizing illnesses in banana plants. Since the research is restricted to specific diseases, it would be better to include diseases affecting bananas. There is also a limit to the accuracy Using an efficient CNN algorithm and minimizing overfitting are the better ways to improve accuracy.

CHAPTER THREE

3. Research Methodology

3.1. General approach

This chapter discusses the methodologies utilized in completing this thesis, including implementation methods for the model, data collection and preparation, system software and hardware configuration, EnsetDNet Model design, and evaluation techniques for assessing the model. The experimental research approach employed in this thesis involves maintaining a set of variables as constant while measuring another set as the subject of experimentation. Various experiments are conducted using different dataset ratios, as well as different activation functions and hyperparameters.

- A literature review on previous research related to the identification of plant diseases utilizing image processing and deep learning methodologies.
- Collect healthy and Diseased Enset leaf images from enset farm.
- The preparation of a required dataset involves the application of various techniques, including image preprocessing, splitting, and augmentation.
- Develop a Convolutional Neural Network model to detect and classify the leaf diseases.
- Evaluate the proposed model with a valid dataset.
- Assess the impact of various learning parameters, such as learning rate, batch size, number of epochs and the ratio between training, validation, and test splits.

3.1.1. Research Flow Diagram

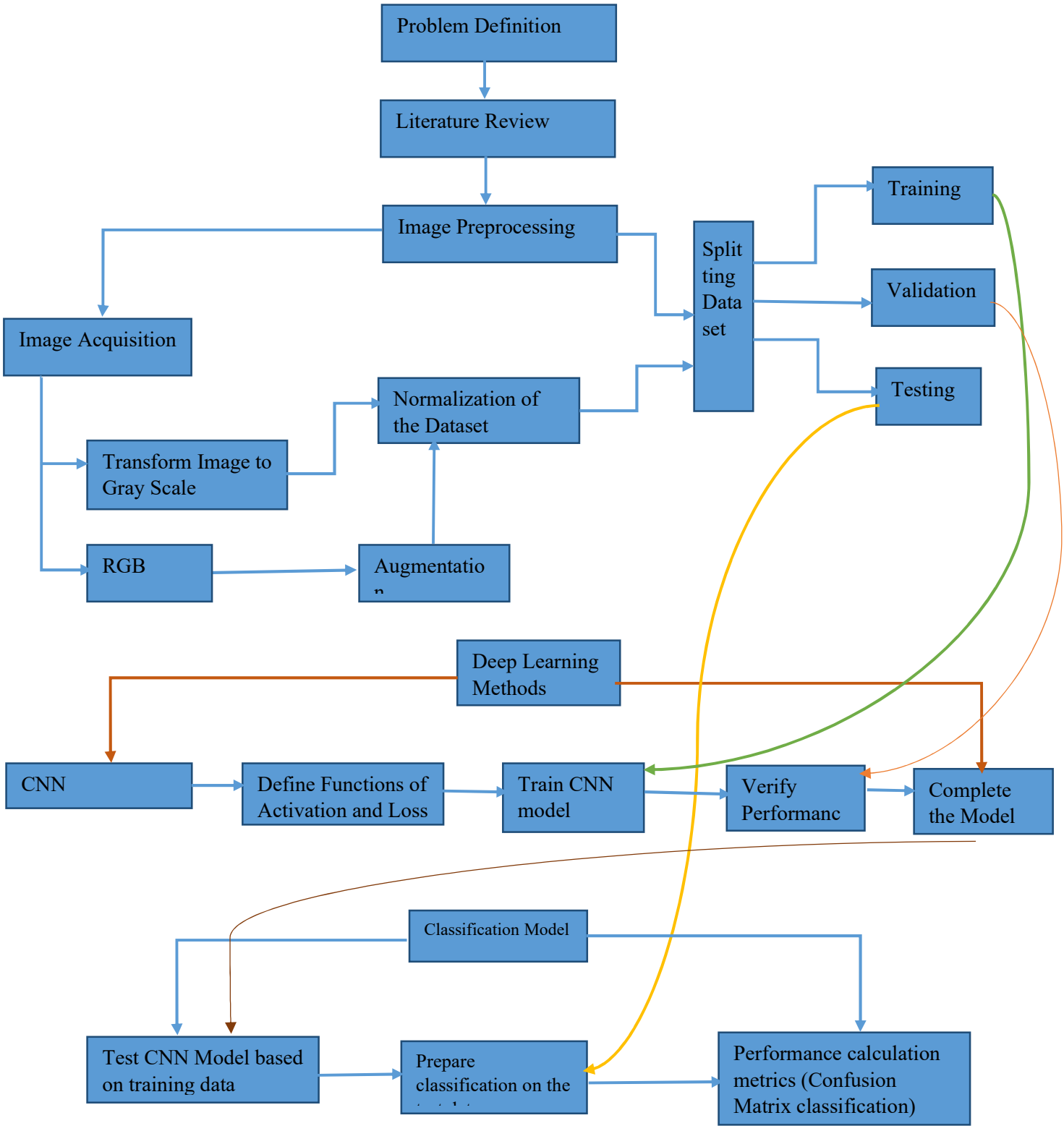

Figure 3.1 Research Flow diagram.

3.2. Implementation

3.2.1. Environment

Anaconda Python 3.10 was used for the research's implementation, together with TensorFlow and Keras, Visual Studio Code, and Jupyter Notebook as the primary integrated development environments. A local Windows 11, 64-bit operating system is used for the analysis.

3.2.2. Experiment

The EnsetDNet model is first developed during experiments. Pretrained models MobileNetV3Small and EfficientNetB7 were chosen to classify the images using a transfer learning approach.

3.2.3. EnsetDNet Developing

We developed our model using a sequential approach rather than a functional approach. Because functional models have a larger range of input and output and are more complex. A sample code for building an EnsetDNet model is shown in Figure 3.2.

```
model = Sequential()
#Adds a max-pooling layer with a 2x2 pool size.
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Flatten())
model.add(Dense(64, activation='relu'))
model.add(Dense(5, activation='softmax'))
model.compile(optimizer= 'Nadam', loss='categorical_crossentropy', metrics=['accuracy'])
summary = model.summary()
print(summary)
model.save('model_original.keras')
```

Figure 3.2 Shows a Sample code for EnsetDNet Model

This is a CNN example based on the EnsetDNet model that makes use of MaxPooling2D, which employs a sliding window across 2D input space, and Conv2D, a 2-D convolutional layer.

- Hyperparameters, which stand for the factors determining the neural network's structure and training methodology, are set before training begins.
- Conv2D with the following hyperparameters: kernel size, filters, and input shape (128, 128, 3) with a softmax 64 activation function for each layer.
- MaxPooling2D layer to minimize the new features' spatial size; MaxPooling2D is the 2D input space (2, 2).
- Repeat with an increasing kernel size of 32, 64, or 128.
- Describe the last fully connected layer, which indicates how many classes are healthy and how many are unhealthy.
- An image classification is produced by the Softmax activation function. Dense (64, "SoftMax" activation)
- Lastly, I printed the model's summary.

3.2.3.1. Parameters for layers

- Convolutional layer filter size and the number of filters should depend on the complexity of the dataset and the depth of the neural network. In our experiments, we used 32, 64, and 128 filters.
- In our developed model, the number of filters, or kernel size, is 3x3.
- The images' width and height were supplied earlier. Three-dimensional input, usually an RGB image, is entered into two-dimensional convolutional layers.
- To avoid overfitting and further reduce the size of the compressed image, max pooling is applied. It involves using a moving window across a 2D input space, with the output, or 2x2, being its highest value.

3.2.3.2. Parameter to fit the model

The smaller tests were slow to proceed, thus there are 65 epochs in total. There is sufficient room for the loss function to converge at this epoch number. This indicates that we are allowing the architecture adequate time to learn. The average time is 54 seconds for each epoch, or 54x65 = 3,510 seconds for all 65 epoch iterations.

3.2.4. Problem Formulation

This section of the study aims to find the primary limitation of the present research concerning the detection and classification of enset plant leaves and stems and proposes potential solutions to address the identified issue. The research focuses on detecting and classifying diseases in enset plant leaves, as well as providing medical treatment. The primary objective of this investigation is to forecast the state of enset leaves. An advanced deep learning-based convolutional neural network model will be trained on images obtained from a digital camera and marked based on the researcher's visual interpretation of the image. The effectiveness of the deep learning model will be assessed using confusion matrices and classification accuracy metrics.

3.2.5. Description of the Study Area

The study area is located in Sidama and Central Ethiopia Regional states namely Hawassa Zuria, Boricha, Yirgalem, Aletawondo, Wonago, and Dilla. In these areas, the climate, temperature, and soil conditions are convenient for enset production, and the plant is abundant.

3.2.6. Study Subject

The study will use Convolutional Neural Network (CNN) methods to detect and classify healthy and unhealthy leaves and stems. The classification will be done by using pre-trained networks-EfficientNetB7 and MobileNetV3 which have been comparatively successful.

3.2.7. Study Design

To achieve the objective of the present study experimental research design will be used. To make this research effective and accurate Deep learning techniques are applied to the images. Images will be taken by digital camera and images previously collected by plant pathologists.

The performance of the Deep learning model will be evaluated based on the Confusion matrices and classification accuracy. The confusion matrix is a correlation between the predictions of a model and the actual class labels of the data points.

3.2.8. Data collection

Based on the research we collected healthy and infected enset leaf and stem images directly from enset farms from Sidama and Central Ethiopia Regional states namely Hawassa Zuria, Boricha, Yirgalem, Aletawondo, Wonago, and Dilla by using Canon EOS1000D digital cameras and images previously collected by plant pathologist.

3.2.9. Evaluation

The use of deep learning models to identify enset leaf and stem disease has received a great deal of attention, so it is important to evaluate their strengths and weaknesses. In this study, the researcher will present an evaluation study that compares the application of deep learning models to the development and classification of enset disease in several stages.

3.2.10. Data Management and Analysis

Comparison and different graphs will be used to evaluate the results that are obtained from the experiments. The proposed method will be evaluated using objective judgments through various commonly used criteria for healthy and unhealthy enset leaf evaluation. This objective evaluation utilizes some Confusion matrix measures to detect and classify the output leaf image.

3.2.11. Materials and tools

An investigation of existing software tools and their libraries is carried out to identify the best tool for implementing the CNN algorithm for enset picture classification. During our inquiry, we discovered that there are tools that are generic for both deep learning and machine learning algorithms, as well as tools that are unique to one of them. Before selecting the tools, we evaluated various characteristics that will assist us in selecting the proper software tools and libraries. Other criteria include selecting tools with sufficient learning materials, such as free video lessons, existing experience, and tools that must be utilized in machines with limited resources (such as CPU only).

We used Python as a programming language with TensorFlow and Kera's packages on an Anaconda and VS Code environment to develop the CNN algorithm. These tools meet all the criteria for consideration and are written in Python, which we are familiar with. Training from the ground up.

3.2.11.1. Software Tools

Anaconda is a free and open-source distribution of the Python and R programming languages for image processing, data science, machine learning, and related applications that aims to expand and simplify package management and deployment[44]. It includes several IDEs for writing code, including Qt Console, Jupiter Notebook, Visual Studio, and Spyder. To accomplish the coding, we used Jupiter Notebook and Visual Studio. It is simple to use and operates in a web browser.

Visual Studio Code, sometimes known as VS Code, is a cross-platform, free, open-source code editor that is compatible with Python and a wide range of languages and functionalities. It provides Azure deployment for online and cloud applications, extensions, Git integration, IntelliSense, and debugging.

TensorFlow is a free and open-source deep-learning package built by Google that is now the most well-known and quickest. It may be used on any desktop running Windows, macOS, or Linux, as well as in the cloud as a service and on mobile platforms like as iOS and Android. TensorFlow's architecture is responsible for data preprocessing, model construction, model training, and model estimation. Tensors (n-dimensional arrays) are used in all TensorFlow computations to represent various types of data. TensorFlow also uses a graph architecture to represent the sequence of computations performed during training. It has two CPU and Google Collab distributions.

Keras is a Python-based high-level neural network API that runs on top of TensorFlow, Theano, or Microsoft Cognitive Toolkit (CNTK). It is very easy to create a Model, user-friendly, easily expandable with Python, and most significantly, it contains pre-trained CNN models such as MobileNetV3 Small and EfficientNetB7 that we used during the experiment. It enables quick prototyping and supports CNN, ENN, or a combination of the two.

Adobe Photoshop Using Adobe Photoshop 2023 to crop, remove background noise and compress and resize a picture is an easy method to minimize the file size and prepare the image for the training data.

Visio 2019: The system architecture is designed using Visio 2019. This program was used to easily build a diagram and create a Gant chart using ready-made templates, assisting in the simplification of difficult information.

Mendeley: By importing it into Microsoft Word, the researcher used this program for reference citation needs by the correct citation standard, and Version 1.19.8 was used.

3.2.11.2. Hardware tool

The sample photographs from the field were captured with a Canon EOS1000D digital camera. An average machine with 11th Gen Intel(R) Core(TM) i7-1185G7 @ 3.00GHz 1.80 GHz processor, memory 16 GB was used to implement the CNN algorithm with the selected software tools.

3.3. Appropriate initialization

Achieving high performance in neural networks heavily depends on selecting the appropriate initialization method for each layer. Properly initializing the weights can greatly impact the network's ability to learn from training data, as it determines the initial position of model parameters that the loss function optimizes during model design. One commonly used initialization method is Xavier, also known as Glorot initialization.

This approach aims to facilitate signal propagation during both forward and backward passes through a layer, which is particularly effective for linear activation functions and sigmoid activations due to their roughly linear unsaturated interval.

Xavier's method considers the number of inputs and outputs of each layer and initializes the weights using a normal distribution with a standard deviation that depends on these values. It works well for networks with a deep architecture.

3.3.1. Image Acquisition and Dataset

A dataset was assembled from images taken with a Canon EOS 1000d 10-megapixel resolution Camera and from plant pathologists previously collected. These images were collected from five distinct farms growing Hawassa Zuria Boricha, Yirgalem, Aletawondo, Wonago, and Dilla during two different times (May-June and December-January) to minimize the loss of disease features caused by specular reflection. The images experienced manual segmentation using Adobe software to remove background information, which eliminates the possibility of the deep learning model learning features from irrelevant information. The images were preprocessed to meet the required input dimensions for deep learning models (specifically 128 x 128 x 3).

The resulting dataset consisted of five classes: five types of diseases and healthy enset leaf, as shown in Table 3.1 The performance of deep learning models is heavily dependent on the quantity of data available.

Insufficient input data can lead to overfitting of the trained model, resulting in poor performance during validation. The literature suggests that there is a lack of clarity regarding the determination of the optimal number of images necessary for classification[45].

To conduct this experiment, it was necessary to generate a dataset of enset leaf images that did not previously exist. This was accomplished by taking photographs of enset leaves from various farms and from previously collected by plant pathologists. The resulting dataset included images of both healthy and diseased enset stems and leaves.

Specifically, the dataset consisted of approximately 5000 images capturing five distinct types of enset leaf diseases. The dataset utilized in this study comprises 5 classes, consisting of a healthy enset leaf and 4 distinct types of enset leaf diseases as follows: 1. Enset Bacterial wilt 2. Bacterial leaf spot 3. Enset Mosaic Virus and 4. Enset insect pest diseases.

Class and Data types	Enset Bacterial wilt	Bacterial leaf spot	Enset Healthy leaf	Insect Pest Diseases	Enset Mosaic Virus Diseases	Total Data
Sample	5	5	5	5	5	25
Collected Images	520	370	505	365	325	2085
Augmented	475	625	490	630	670	2890
Total	1000	1000	1000	1000	1000	5000

Table 3.1 The disease's name and number of Images collected.

3.3.2. Image augmentation

Image augmentation is a technique used in deep learning and computer vision to increase the amount and diversity of training data by transforming existing images to create new variations. This helps prevent overfitting and improves the model's ability to generalize to new, unseen data.

Image augmentation, dropout, transfer learning, and other methods are utilized to increase the size of the data set. Innovative Augmentation builds a network that generates augmented Data automatically during the training phase, lowering network loss[46].

Common image augmentation techniques include:

- Random cropping: generating new images by randomly selecting its region of interest from the original image.

- Rotations: rotating the image by a certain angle to generate variations.
- Flipping: flipping the image horizontally or vertically to generate a new variation.
- Scaling: resizing the image to a different size to generate variations.
- Color adjustments: changing the brightness, contrast, and saturation of the image to generate new variations.

Parameter	Values
Rotation range	20
Rescale	1./255
Shear Range	0.2
Zoom Range	0.2
Horizontal Flip	True
Width Shift range	0.2
Height Shift Range	0.2
Fill mode	Nearest

Table 3.2 Augmentation Techniques used

By applying these techniques, the model is shown to a wider range of images and enhances its ability to identify patterns in the data, resulting in better performance and accuracy.

3.3.3. Image Pre-processing

The datasets we utilized, namely our collected dataset and Plant village datasets, consist of RGB images of arbitrary size. The data set collected a dimension of 3888 x 2592 to conform to the input image shape assumed by most neural network models, we resized them to a uniform aspect ratio of 128x128 pixels. In addition, we performed preprocessing steps such as removing low-frequency background noise and normalizing intensity. As part of our image processing procedure, we manually cropped all images to create a square around the leaves and stems and ensured that each image contained all the necessary information for feature extraction. Then we perform a preprocessing step and in this step, each image in the dataset is resized then thresholding and contour masking is performed [25].

3.3.4. Feature extraction

Following preprocessing, the portion of the image that corresponds to the disease is separated and designated as a region of interest for further image processing. Additional features are subsequently extracted from this region based on characteristic symptoms of the disease to facilitate the detection and classification of different types of diseased leaves.

To perform this, CNN architecture was built with multiple overlying convolutional layers. Normally, a convolutional layer will contain convolution and activation functions that are non-linear in our case, and the utilization of ReLU and pooling process have been enabled [31].

3.3.5. Categorizing images according to their real classes

We talked with Plant pathologists before beginning data collection, and then we talked with them once more after preprocessing and data collection about how to categorize our data into the right classifications. Plant Protection professional from Hawassa University Agriculture College's Laboratory then manually tagged photos with the relevant classes.

3.3.6. Converting RGB to Grayscale Images

Using multiple color models, such as RGB (red, green, blue), CMYK (cyan, magenta, yellow, and key or black), HSV (hue, saturation, value), etc., one can convert an image from one color to shades of gray using a process called gray scaling. Various image processing methods can be used to convert an RGB image to a grayscale image. In our experiment, we used the TensorFlow, keras preprocessing ImageDataGenerator Python function. This procedure was carried out in the process of identifying an effective model.

3.3.7. Creating and building the model

The fields of image and natural language processing heavily rely on the CNN algorithm. Because of its multi-layered structure, it can swiftly extract the essential pieces and correctly evaluate visual images. It can also recognize and categorize items with minimal to no pre-processing. Through the process of training, validating, and testing the gathered data, a CNN-based model was created.

3.3.8. Analyzing the model's performance

Using training, verifying, and testing datasets based on actual and predicted classes, the model's performance is assessed. The model's main performance metrics in classification tasks are F1 score, accuracy, precision, and recall.

3.3.9. Data Splitting

The collected photos were split up into three sections. 10% is used for model testing, 20% is used for validation, and 70% is used for training. We get accurate numbers for the model because of this data separation.

3.3.10. Contrasting the created model with CNN pre-train models

We use transfer learning to adapt the trained model to fit a particular set of related problems, or we use the model exactly as is. To compare our generated model's performance we used two pre-trained models, namely MobileNetV3Small and EfficientNetB7.

3.3.11. Types of Optimization Algorithm

In this section, we will explain the prevalent optimization algorithm that holds significance in optimizing our computation. This is crucial as, when dealing with a large quantity of images, the backpropagation step tends to slow down significantly. This is because the weights and biases are not updated until the gradient descent is computed for the entire data set, which may contain thousands or even millions of images.

3.3.12. Mini-batch Gradient Descent

Mini-batch Gradient Descent is a stochastic optimization algorithm used in machine learning for optimizing the parameters of a model. It is a variation of the Gradient Descent algorithm, where instead of updating the parameters for all the training examples at once, the updates are made for a small subset of the training examples, known as a mini-batch. It is possible to calculate gradient descent for a mini-batch at each interaction, rather than for the entire data set. However, it is crucial to note that all weights and biases should be updated with the values obtained from the only batch at each step, known as an epoch.

The mini-batch size is typically chosen to be a small fraction of the entire training set, typically between 32 and 512 examples. This allows for faster convergence and better utilization of computer resources, as the algorithm can be parallelized across multiple processing units.

In Mini-batch Gradient Descent, the gradient is calculated for each mini-batch using the same formula as in Gradient Descent, but with the summation over the mini-batch rather than over the entire training set. The parameters are then updated based on the average of the gradients over the mini-batch. There are a lot of different optimization algorithms the most common and cited in the paper are GD.

3.3.13. Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an iterative optimization algorithm used in machine learning to minimize the cost function of a model. It is a variant of gradient descent that randomly samples a small subset of the training data (called a mini batch) to compute the gradient and update the model parameters in each iteration.

The benefits of using SGD over regular gradient descent include faster convergence, improved generalization, and the ability to handle larger datasets. However, SGD may be less accurate than gradient descent due to the randomness introduced by the sampling of mini batches. SGD can be used in various machine learning models such as logistic regression, neural networks, and deep learning models. It is commonly used in deep learning models, where the large size of the training data and model parameters make the use of traditional gradient descent infeasible.

3.4. Methods of Measuring Performance

3.4.1. Cross-Entropy Loss Function

Cross entropy is the standard loss function utilized for binary classification tasks, specifically designed for situations where the target variables range between 0 and 1. From a mathematical perspective, it is the optimal loss function under the maximum likelihood inference framework. The cross-entropy loss should be evaluated as a primary measure and only replaced if necessary.
During neural network training, the cost function is the key to adjusting a neural network's weights to create a better-fitting machine learning model. Specifically, during forward propagation, the neural network is run on training set data, and outputs are generated which in the case of classification indicate the probability or confidence in possible labels.
The loss function determines a penalty for each difference between the target label and the neural network's outputs after comparing these probabilities to the target labels. During backpropagation, the partial derivative of the loss function is calculated for each trainable weight of the neural network[47]. The weights are adjusted by these partial derivatives. Under normal conditions, backpropagation iteratively adjusts the trainable weights of a neural network to produce a model with a lower loss[47].

The loss function, Binary cross-entropy, or log loss is a prevalent measure used in machine learning and deep learning algorithms to optimize the performance of binary classification models. It assesses the disparity between predicted probabilities and actual labels.

3.4.2. Regularization Techniques

Neural networks tend to overfit, meaning that they exhibit high accuracy for training data but poor performance on new datasets. To address this challenge, regularization techniques can be employed. Dropout and early stopping are the two most widely used regularization methods.

3.4.3. Dropout

The introduction of stochasticity in neuron activations is being proposed as an effective approach to regulating deep neural networks. Dropout, which randomly drops the value of neurons and sets them to zero during network optimization, is one such method. The outcome is a network that generalizes better and is less susceptible to overfitting.

3.4.4. Early Stopping

It has been previously explained that an overfitting model demonstrates high performance on training data while displaying poor performance on unobserved data. The network's performance on unobserved data is assessed through a validation set during training. This set is a representative portion of the test set that has not yet been observed by the model. Validation error rises as the training error declines in an overfitting network. By monitoring the lowest validation error after each epoch, one can stop training when the model's validation set performance fails to improve for several epochs.

3.4.5. Regularization Methodologies

Any data can be memorized by deep and massive neural networks. During training, their accuracy on the training set often improves while it worsens on the test set. This is referred to as overfitting. These various regularization procedures are used to deal with the overfitting problem.

3.4.6. Metrics for Evaluating the Model's Accuracy

The F1 score and accuracy are used in this experiment to assess the model's performance. The model is trained on 80% of the data, with the remaining 15% utilized for validation and the last 5% for testing.

In this classification job, the metrics used to evaluate the model are:

- F1 score
- Precision
- Accuracy of classification accuracy (CA)
- Recall

Accuracy: The model's accuracy will be calculated as the proportion of correct predictions of the top class (the class with the highest probability as indicated by the CNN model) and the target class previously specified by the author. It is shown by the below equation:

$$\text{Accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Equation 3.1 Accuracy

Where TP (True Positive) represents the positive instances that are correctly classified as positive, TN (True Negative) represents the negative instance that is correctly classified as negative, FP (False positive) represents the negative instance that is wrongly classified as positive, FN (False Negative) represents the positive that is wrongly classified as negative.

Precision is measured as the fraction of true positives (TP) divided by the sum of the relevant classes, i.e., the sum of true positives and false positives. It is expressed by the following formula:

$$Precision = \frac{TP}{TP + FP}$$

Equation 3.2 Precision

Recall - is calculated as a fraction of true positives divided by the sum of true positives and false negatives. It is shown by the below formula:

$$\text{Recall} = \frac{TP}{TP+FN}$$

Equation 3.3 Recall

The F1 score will be utilized due to the imbalanced nature of the dataset. The F1 score is defined as the harmonic mean of precision and recall, and can be expressed as follows:

$$\text{F1 Score} = \frac{2*\text{precession}*\text{Recall}}{\text{Precession} + \text{Recall}}$$

Equation 3.4 F1 Score

Macro-Average: This is possibly the easiest of the various averaging methods. To obtain the macro-averaged F1 score, the arithmetic mean of each per-class F1 score is applied. This method holds all classes to the same standards regardless of their support settings. Equation 3.5 shows the classification report's macro-average. The macro average is equal to the F1 score $n\ c$ = 1 for all X100 classes. Equation 3.5: The classification report's macro average.

$$\text{Macro average} = \frac{\sum_{c=1}^{n} \text{F1 score}}{\text{total number of classes}} \text{X100}$$

Equation 3.5 Macro-Average of Formula

Weighted Average: It is calculated by averaging the fraction of accurate predictions made in each class over all classes. This can be expressed as the class's total number of instances divided by the number of successfully anticipated instances. Equation 3.6 shows the classification report's weighted average. Average Weighted = ∑ F1 score x support ratio $n\ F$=1. Equation 3.6, the categorization report's average.

$$\text{Weighted Average} = \sum_{F=1}^{n} \text{F1 score x support proportion}$$

Equation 3.6 Weighted Average Formula

A confusion matrix: - is a type of evaluation statistic that is used to describe how well a categorization task performs. A confusion matrix can be used to calculate precision, recall, and accuracy.

Confusion matrix	Positive (Actual)	Negative (Actual)
Positive (prediction)	True positive (TP)	False Positive (FP)
Negative (prediction)	False Negative (FN)	True Negative (TN

Table 3.3 Confusion matrix

3.5. The proposed system's design

It explores the proposed model's design and descriptions, the extraction of features, and how classification is executed in both the proposed model and other relevant models through a technique known as training from scratch. These topics are briefly elaborated upon.

3.5.1. Data Preparation

There are 5000 (five thousand) photos in the entire collection, divided into five classes. There are one thousand (one thousand) relevant photos in each class. We used 700 images (seven hundred) for training, 200 images (two hundred) for validation, and 100 images (one hundred) for testing from each class. I have collected 2085 images directly from the farm and the plant pathologist. The remaining images have been augmented.

3.5.2. Model Selection

The literature on computer vision, specifically image classification, has extensively explored the use of Convolutional Neural Networks (CNN) as a deep learning algorithm. CNNs offer a promising approach to adaptive image processing, performing tasks such as feature extraction, classification, training, testing, and accuracy evaluation. Unlike traditional methods that require separate preprocessing or feature extraction stages, CNNs directly operate on the data with preprocessing included. Furthermore, CNNs naturally integrate feature extraction and classification into a single framework.

The key advantage of CNN over its predecessors is that it automatically discovers the significant properties without any human supervision, which makes it the most used. In the field of machine learning, deep learning (DL) has become one of the most popular studies due to its enormous popularity. Therefore, we have dug in deep with CNN by presenting the main components of it [48].

3.6. EnsetDNet

A sequential model created and developed to detect Enset leaf and stem disease is the EnsetDNet architecture. The photos are resized to 128x128*3, with 3 indicating the image's color channel. Three fully connected layers with a flattening layer, five maximum pooling layers, and five convolutional layers make up EnsetDNet. ReLu is implemented in the output layer after SoftMax and convolutional and fully connected layers. There are 310,917 total parameters in the network. The architecture of EnsetDNet is shown in Figure 3.2.

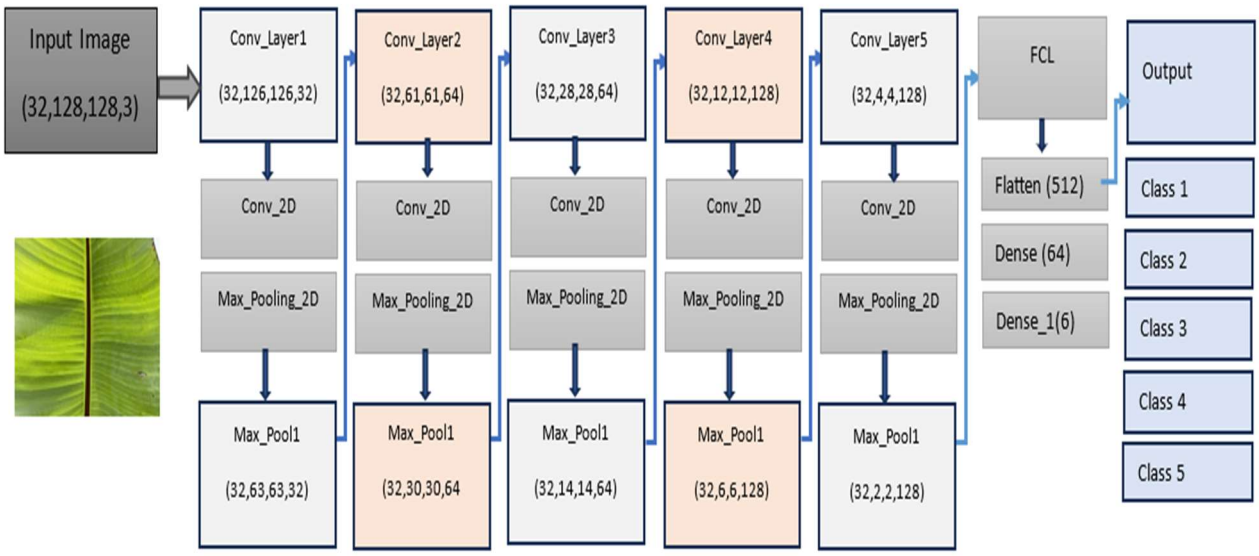

Figure 3.2 Architecture of EnsetDNet

3.6.1. Overview of EnsetDNet Architecture

Every convolution layer's output size can be calculated by considering the input picture size (W), kernel size (K), stride (S), and amount of padding (P). For every convolutional layer, the exact result size is provided by the following formula. The output volume for convolution layers of EnsetDNet is displayed in Equation 3.7.

$$\text{Output size for the next convolution} = \left[\frac{\text{Win}-\text{K}+2\text{P}}{\text{S}}\right] + 1$$

Equation 3.7 Output size for the next convolution layers of EnsetDNet

Where: W is the input volume, K is the Kernel size, P is the padding and S is the stride.

The initial spatial size of the input volume (W) is 128x128x3, and they are modified convolution processes; the filter's initial size is 3x3, and its stride is 1. There is no zero padding P in our network, and P's value is constant to zero throughout the model; their sizes are modified following a few convolution and pooling operations.

3.6.1.1. Input layer

The input layer in our EnsetDNet model receives 128x128x3 RGB and grayscale photos with five different classes: Healthy, Bacterial Leaf Spot, Mosaic Virus, Bacterial Wilt, and Insect Pest.

3.6.1.2. Convolution Layer

The EnsetDNet model consists of five convolutional layers. ReLu nonlinearity is an activation function that is used.

- The 126 x 126 x 32 input image is filtered by the first convolution layer of the model using 32 kernel 3 x 3 kernels with a stride of 1. There are 896 total parameters in this layer.
- The pooled output of the first convolutional layer, which measures 61 x 61, is filtered by the second convolutional layer using 32 kernels of size 3 x 3. In total, there are 18496 parameters.
- The third convolutional layer uses 32 3x3 kernels to filter the pooled output of the first convolutional layer, which has a size of 28x28. There are 36928 parameters in all.
- The third convolutional layer receives the pooled output of the first convolutional layer, which measures 28x28. It filters the input using 32 3x3 kernels. The total parameters for it are 36928. There are 32 12x12x128 kernels in the fourth convolutional layer. 73856 are its overall parameters. Lastly, there are 32 4x4x128 kernels in the fifth convolutional layer. 147584 are its overall parameters.

3.6.1.3. Pooling Layer

Using the highest value from the filtered image, this layer compresses the image into a smaller size and creates a matrix to manage overfitting. Five max-pooling layers make up the EnsetDNet model.

- Using a filter with an image size of 63x63 and 64 filters, the first max-pooling layer reduces the output of the first convolution layer. The output of the second convolutional layer is sent into the second max-pooling layer, which pools using 30x30 picture sizes and 64 filters.
- The images in the third max-pooling layer are 14 by 14 and include 64 filters. The images in the fourth max-pooling layer are 6x6 and have filters set to 128. Images in the fifth max-pooling layer are 2x2 and have 128 filters.

3.6.1.4. Fully connected layer

The output layer is one of three completely linked layers in the EnsetDnet model. With 512 and 64 neurons, it flattens the final convoluted data into a vector value before accepting the output of the five-layer convolution layer. This layer computes the class score and the number of neurons that were predefined throughout the model's development. All the neurons in this layer are connected to each other layer.

3.6.1.5. Out Put Layer

The final layer of the model is known as the output layer, and it consists of five neurons with SoftMax activation functions. Because it assigns five classes in our model in a categorical approach. The parameters of the EnsetDNet model are shown in Figure 3.3.

```
#Structure of the Model
model.summary()
```

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d (MaxPooling2 D)	(None, 63, 63, 32)	0
conv2d_1 (Conv2D)	(None, 61, 61, 64)	18496
max_pooling2d_1 (MaxPoolin g2D)	(None, 30, 30, 64)	0
conv2d_2 (Conv2D)	(None, 28, 28, 64)	36928
max_pooling2d_2 (MaxPoolin g2D)	(None, 14, 14, 64)	0
conv2d_3 (Conv2D)	(None, 12, 12, 128)	73856
max_pooling2d_3 (MaxPoolin g2D)	(None, 6, 6, 128)	0
conv2d_4 (Conv2D)	(None, 4, 4, 128)	147584
max_pooling2d_4 (MaxPoolin g2D)	(None, 2, 2, 128)	0
flatten (Flatten)	(None, 512)	0
dense (Dense)	(None, 64)	32832
dense_1 (Dense)	(None, 5)	325

Total params: 310917
Trainable params: 310917
Non-trainable params: 0

Figure 3.3 EnsetDNet Model Parameters

The EnsetDNet model has 310,917 parameters, as shown in above figure 3.3. Compared to other pre-trained CNN architectures, this model shows high performance despite requiring minimum resources and data for training. The number of parameters for each convolution layer can be verified using the following equation. The total parameters of the convolution layers can be found using the formula in Equation 3.8.

$$Number\ of\ Parameter = (Kernel\ size * Input + 1) * Number\ of\ Filters$$

Equation 3.8 The total parameters of the EnsetDNet model convolution layers

As an example, the first convolutional layer in our model has parameters (3*3*3+1) *32=896; the second convolutional layer has parameters (3*3*32+1)*64=18496; and further on.

3.6.2. Hyperparameters of EnsetDNet Model

Optimization algorithms: The EnsetDNet model is trained with a backpropagation of error algorithm to update the weights and a mini-batch gradient descent optimization algorithm to reduce the error rate. It minimizes the loss function by adjusting parameters and updating the model's weight. In the experiment, we used the Nadam optimizer to optimize our model.

Learning rate: The EnsetDNet model is trained using backpropagation, and weight adjustments are contingent on the rate of learning. During backpropagation, it establishes the amount of weight to update. When we give a smaller number, the model performs better than a model with a faster learning rate. Learning rates of 0.1, 0.01, and 0.0001 were used in the experiment. Therefore, a learning rate of 0.001 is ideal for every experiment.

Loss function: The final fully linked layer of the model, the output layer, uses activation functions that are closely related to the loss function. Instead of using binary cross entropy, we choose the categorical cross-entropy loss function for the type of problem that the EnsetDNet model is designed to address. We had five classes in all because we had more than two sessions.

Activation function: In the EnsetDNet model, experiments are performed using Sigmoid, SoftMax, and ReLu in the hidden layers; SoftMax performs better than the others. The model's output layer uses the SoftMax activation function since it is suitable for categorical classification problems.

Number of epochs: The model is trained using a variety of epochs, beginning at 25, 35, 45, 55, 65, 75, 85, 95, 105 and 115 epochs. After multiple tests, the model's optimal value is found at epoch 65.

Batch size: Using batch sizes of 32, 64, and 128 for the training dataset, we experiment to build our model. By optimizing for other hyperparameters, we were able to obtain the ideal model with a batch size of 32.

CHAPTER FOUR

4. Result and Discussion

This chapter provides an in-depth explanation of the research implementation process used to construct an effective and efficient model, including all the details of the experiments, including discussions, accuracy and loss graphs, and experiment outcomes.

4.1. Experiment Result

We divided the dataset into training, validation, and testing sets and applied convolutional neural network (CNN) models for disease detection and classification. The color feature of the images is used to categorize them. We can categorize the image as healthy, diseased, or as a variety of diseases.

To examine the performance of categorization, three distinct scenarios are carried out in the experiment. Pre-trained models serve as the foundation for the first two scenarios, while our study's innovative EnsetDNet model is used in the third.

4.1.1. Parameter tuning for the EnsetDNet model

4.1.1.1. Dataset ratio for training and testing

The results of the experiment with a variable ratio for the training and testing split with a batch size of 32, an epoch of 65, and a learning rate of 0.001. The data splitting ratio is shown in Table 4.1.

Training Split	Testing Split	Accuracy			Loss		
		Training	Validation	Testing	Training	Validation	Testing
80%	20%	97.83%	97.78%	97.01%	4.53%	6.54%	6.03%
70%	30%	98.91%	99.30%	98.92%	2.46%	1.75%	98.02%
60%	40%	96.99%	96.03%	95.98%	8.05%	9.10%	9.88%

Table 4.1 Data Split ratio

4.1.1.2. Number of Epochs

Table 4.2 shows the accuracy and loss values obtained from the experiment at various epochs. The experimental results using various epochs are shown in Table 4.2.

Epochs	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
25	96.63%	95.24%	95.64%	18.26%	15.53%	15.61%
35	96.82%	96.41%	97.34%	13.08%	12.08%	14.13%
45	98.43%	97.56%	98.19%	8.60%	10.08%	11.34%
55	96.39%	97.09%	97.19%	6.02%	8.32%	8.99%
65	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%
75	98.86%	96.93%	96.01%	6.46%	13.65%	14.50%
85	99.19%	99.09%	98.16%	7.08%	11.03%	12.88%
95	98.12%	98.98%	99.02%	3.89%	2.98%	3.65%
105	99.04%	99.10%	99.60%	1.08%	2.67%	1.43%
115	99.80%	99.89%	99.01%	0.67%	2.01%	1.08%

Table 4.2 Result using various Epochs

4.1.1.3. Batch Size

The outcomes of our studies with various batch sizes are displayed in Table 4.3. Table 4.3 shows the various batch size results from our study.

Batch Size	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
32	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%
64	98.42%	98.67%	98.87%	4.42%	3.92%	2.98%
128	96.45%	96.78%	97.01%	6.03%	7.45%	8.32%

Table 4.3 Results various Batch Sizes

4.1.1.4. Learning Rate

The EnsetDNet model experiment obtained the following result when it used various learning rates with an epoch of 65 and a batch size of 32. On RGB images, various learning rate values are displayed in Table 4.4.

Learning Rate	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
0.1	20.97%	18.07%	18.93%	193.07%	189.04%	188.08%
0.01	18.09%	17.77%	17.39%	183.31%	185.45%	185.95%
0.001	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%
0.0001	89.06%	88.04%	88.99%	25.07%	28.51%	28.60%

Table 4.4 The difference in learning rate for RG

4.1.1.5. Fully connected layer activation function

The EnsetDNet model experiment with batch size 32 and epoch 65 obtains its output by using sigmoid and SoftMax activation functions in the fully connected layers. The fully connected layers that arise from various activation functions are displayed in Table 4.5.

The various activation functions in Table 4.5 produce RGB images.

Activation Function	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
SoftMax	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%
Sigmoid	97.92%	98.17%	97.72%	6.09%	4.89%	4.73%

Table 4.5 The various activation functions

4.1.1.6. Model Optimizer

The EnsetDNet model experiment obtained the following result when it used various optimizers with an epoch of 65 and a batch size of 32. experimenting with various optimizers' values are displayed in Table 4.6.

Optimizer	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
SGD	81.97%	80.50%	81.67%	50.85%	48.74%	50.43%
RMSProp	89.41%	88.98%	88.91%	31.08%	26.91%	26.08%
Adadelta	67.77%	67.37%	69.40%	112.86%	107.08%	111.70%
Adam	97.06%	98.61%	98.09%	8.53%	4.43%	4.98%
Adagrad	86.11%	84.86%	85.74%	39.84%	41.90%	40.46%
Adamax	96.07%	96.97%	97.82%	9.32%	6.86%	6.06%
Nadam	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%

Table 4.6 Experimental values using various optimizers.

4.2. Result Analysis of EnsetDNet Model

4.2.1. EnsetDNet model results are analyzed using RGB and grayscale photos

The RGB provides a training accuracy of 98.91%, and the selected hyperparameters provide a validation accuracy line value of around 99.30%. Decreasing from 0.0413 to 0.0121, respectively, are the training and validation loss curves. With the validation accuracy increasing and the validation loss decreasing, the curves show that the proposed model is not overfitted. RGB image accuracy and loss curves are shown in Figure 4.1.

Figure 4.1 RGB image accuracy and loss curves.

A validation accuracy line with chosen hyperparameters has a value of around 95.28%, whereas training accuracy on grayscale images is obtained at 95.80%. There has been an increase in the training and validation loss curves from 1.6 to 0.1293, respectively. In light of this, RGB photos are more accurate. Grayscale accuracy and loss curves Images are shown in Figure 4.2.

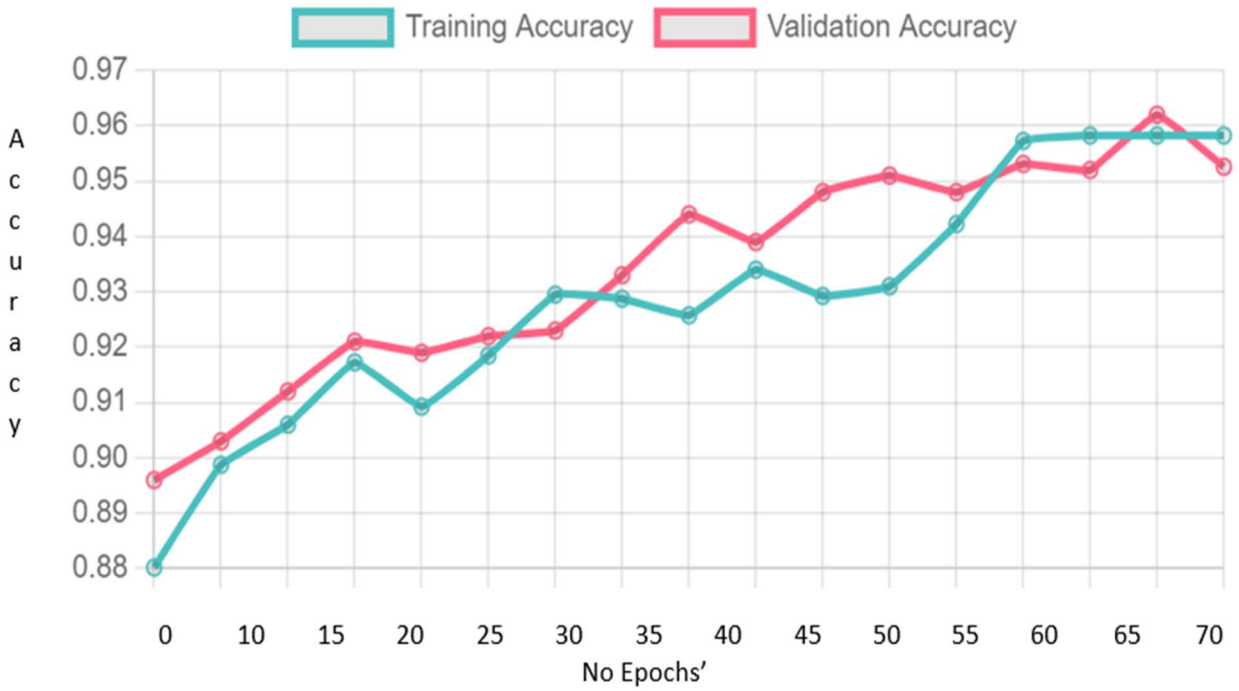

Figure 4.2 Gray Scale image accuracy and loss curves

We experimented with both RGB and grayscale image datasets to build effective models. On the other hand, RGB image datasets perform the best. Table 4.7 below displays the accuracy and loss values of grayscale and RGB images.

Metrics	Accuracy			Loss		
	Training	Validation	Testing	Training	Validation	Testing
Results using RGB photos	98.91%	99.30%	98.92%	4.03%	2.46%	1.75%
Results using Gray photos	94.98%	95.06%	95.74%	11.08%	9.07%	9.76%

Table 4.7 Grayscale and RGB image accuracy and loss values.

Results from the EnsetDNet model experiment, which was conducted on RGB and grayscale photos, are shown in the above table. However, the RGB images achieve more accuracy. Thus, for all of our studies, we chose the RGB image dataset.

4.2.2. EnsetDNet Model Classification Result

To train and test the network of this developed model, RGB and grayscale images were used to differentiate between five classes. The Nadam optimizer, a learning rate of 0.001, a sparse categorical cross-entropy loss function, and 65 epochs of training were used for both networks.

4.2.3. Model confusion matrices

Several parameters that may impact the model's efficiency are used to assess the EnsetDNet model. These include learning rate, training test split ratio, and multiple epochs. The best-performing parameter ratios are employed in this experiment, while other ratios are used for evaluation purposes.

4.2.4. Model classification report

A classification report is utilized to evaluate the accuracy of predictions generated by a classification algorithm. One can determine the proportion of accurate and inaccurate predictions by using its parameters, which include true positive, true negative, false positive, and false negative. Figures 4.3 and 4.4 below show the classification report for RGB and Gary scale images respectively.

Classification report of RGB:

	precision	recall	f1-score	support
Enset bacterial leaf spot disease	0.98	0.99	0.98	100
Enset bacterial wilt diseases	0.99	1.00	0.99	100
Enset healthy leaf	0.99	1.00	1.00	100
Enset insect pest disease	0.98	0.99	0.99	100
Enset Mosaic Virus	0.99	0.99	0.99	100
accuracy			0.99	500
macro avg	0.99	0.99	0.99	500
weighted avg	0.99	0.99	0.99	500

Figure 4.3 Classification report for RGB images with selected parameters.

The accuracy, precision, recall, and f1_score values for the RGB image dataset are displayed in the classification report above.

Classification report Gray Scale:

	precision	recall	f1-score	support
Enset bacterial leaf spot disease	0.94	0.94	0.95	100
Enset bacterial wilt diseases	0.93	0.94	0.95	100
Enset healthy leaf	0.97	0.95	0.97	100
Enset insect pest disease	0.98	0.95	0.97	100
Enset Mosaic Virus	0.96	0.96	0.95	100
accuracy			0.94	500
macro avg	0.97	0.97	0.97	500
weighted avg	0.97	0.97	0.97	500

Figure 4.4 Classification report on grayscale images with selected parameters.

4.2.5. Transfer Learning Experiment

To compare results with our newly developed model, we used pre-trained models, MobileNetV3Small and EfficientNetB7. To fit pre-trained models, we enlarged our dataset from 128x128x3 to 224x224x3. In addition, we included an output layer with a SoftMax activation function for both models, a dense layer with 64 neuron and relu activation functions, and a flattening layer.

4.2.6. EfficientNetB7 model to experiment

Using 65 epochs and other parameters like those in the developed model, the EfficientNet pre-trained model achieves 93.08% accuracy. The hyperparameters that we employed in our proposed model are also employed in this one. We saved this model in a Keras file format. since deep learning-based models use this file format, which is the newest. There are more than 3 million trainable parameters in the EfficientNetB7. However, the model's parameters decrease to about 1,939,120 when we apply this information to our new issue. These methods can reduce computational complexity, costs, and training time. Figure 4.6 shows the MobileNetV3Small transferable learning model architecture.

Layer (type)	Output Shape	Param #
EfficientNetB7 (Functional)	(None, 7, 7, 576)	1939120
flatten_1 (Flatten)	(None, 28224)	0
dense (Dense)	(None, 64)	2806725
dense_1 (Dense)	(None, 5)	325

Total params: 3146117
Trainable params: 2806725
Non-trainable params: 1939120

Figure 4.5 EfficientNetB7 model to experiment using local data

4.2.7. MobileNetV3 Small Experiment

As Figure 4.7 shows, the MobileNetV3 Small has a total of more than 2 million trainable parameters. However, the model's parameters decrease to about 1,806,725 when we apply this information to our new situation. These methods can reduce computational complexity, costs, and training time.

Layer (type)	Output Shape	Param #
MobilenetV3small (Functional)	(None, 7, 7, 576)	939120
flatten_1 (Flatten)	(None, 28224)	0
dense (Dense)	(None, 64)	1806400
dense_1 (Dense)	(None, 5)	325

Total params: 2745845
Trainable params: 1806725
Non-trainable params: 939120

Figure 4.6 MobileNetV3 Small Experiment Using Our Local Data

Using 65 epochs and other parameters like those in the developed model, this pre-trained model demonstrates an accuracy of 95.32%. Since TensorFlow supports mobile and embedded devices, we saved the created model once it was trained.

When we compared the accuracy of the proposed model to the pre-trained model using our local dataset, the proposed model performed more accurately. In addition to that the proposed EnsetDNet model may have undergone fine-tuning on the local dataset, allowing it to adapt its weights and parameters to better fit the specifics of the data. The pre-trained model, on the other hand, may not have been fine-tuned for the local dataset, leading to lower accuracy compared with our novel model.

Overall, the superior performance of the EnsetDNet model compared to the pre-trained model on the local dataset can be attributed to a combination of factors such as data relevance, domain specificity, fine-tuning, and model architecture.

4.2.8. Evaluating the models' performances

Figure 4.8 below shows comparison charts of pre-trained models like MobileNetV3Small and EfficientNetB7 and developed EnsetDNet based on accuracy.

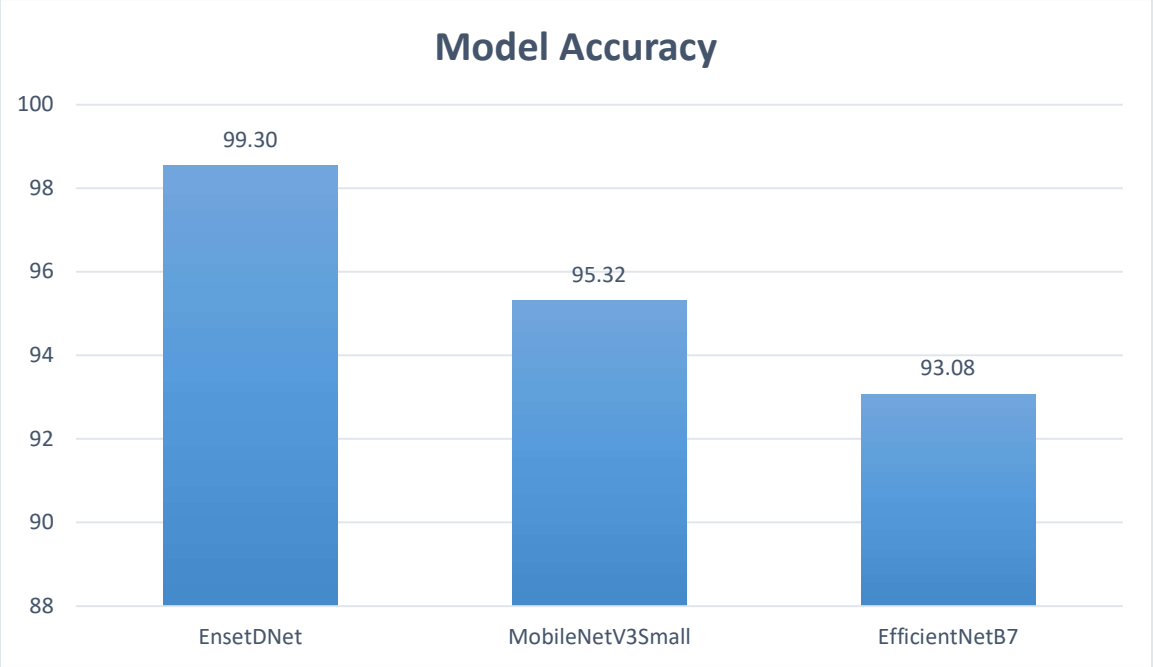

Figure 4.7 Shows the evaluating model's performances.

4.3. Discussion

To compare the performance of the pre-trained EfficientNetB7 and MobileNetV3Small, we developed the proposed EnsetDNet model. All the studies used the same hardware and system specifications, which we covered in the research methodology's section on hardware requirements.

We used 5,000 images in our complete dataset to train, validate, and test the models for five different classes. In our investigation, 1000 photos were used for each class. However, it can only detect issues with the leaves and stems; diseases of the roots are not included.

The results obtained from the experiments conducted on the dataset are quite promising. The dataset was split into 70% for training and 30% for testing, with an epoch size of 65. The model achieved an impressive validation accuracy of 99.30% and a training accuracy of 98.91%, showcasing the effectiveness of the training process.

The batch size used was 32, which allowed for efficient processing of the data during training. A learning rate of 0.001 was utilized to optimize the model's performance and improve convergence during training. Two different activation functions were implemented in the model - Softmax and Relu. Softmax activation function was used for the output layer to produce probability distribution over multiple classes, while Relu activation function was used in the hidden layers to introduce non-linearity into the model.

The optimizer used in the model was Nadam, which is a combination of Nesterov accelerated gradient and Adam optimizer. This optimizer helped in efficiently updating the network weights and converging faster during training.

The novel model EnsetDNet achieved an impressive validation accuracy of 99.30%, surpassing the performance of the pre-trained models EfficientNetB7 and MobileNetV3Small. EfficientNetB7 achieved a validation accuracy of 93.08% using the original data, while MobileNetV3Small achieved a validation accuracy of 95.32%.

The superior performance of the EnsetDNet model can be attributed to its architecture and training process. The model was specifically designed and optimized for the given dataset, allowing it to learn the features and patterns that are most relevant to the task at hand. In contrast, the pre-trained models may not have been as well-suited to the dataset, leading to lower accuracy levels. The results highlight the effectiveness of the novel model EnsetDNet in achieving high accuracy and outperforming pre-trained models.

The EnsetDNet model demonstrates the power of custom model design and training in achieving high levels of accuracy on complex tasks. Further research and experimentation with novel model architectures and hyperparameters may lead to even greater advancements in the field of machine learning and artificial intelligence.

Overall, the results of the model training and testing reflect a well-designed and well-trained deep-learning model. By carefully selecting the data split, hyperparameters, activation functions, and optimizer, the model was able to achieve exceptional performance on the dataset. Further improvements could potentially be made by fine-tuning the hyperparameters or experimenting with different architectures, but the current results are certainly impressive.

CHAPTER FIVE

5. Conclusion and Recommendation

5.1. Conclusion

In this study, we described Enset, its leaf and stem diseases, computer vision, image processing, and the use of deep learning techniques like CNN and transfer learning to detect and classify these diseases. The study's methodology is described. On its dataset, the outcomes, detection rates, and research methodology were also provided and described for both the pre-trained and novel models. A dataset of locally gathered Enset photos taken with a digital camera was used in this research. The best outcome that deep learning can detect in Enset leaves is demonstrated by the testing results. Lastly, this thesis illustrates the properties of the model's selected and displayed parameters and hyperparameters.

The quality and quantity of enset production are impacted by many issues related to its various diseases. Diseases caused by fungi, viruses, insects, and bacteria provide the most threats to the product yield. A low-cost, easily operated technological solution is required to identify the issues early on. We have created a model using CNN and image processing techniques together with a deep learning approach to enable early disease identification. To identify and classify the healthy and unhealthy classes, we developed the EnsetDNet model by collecting photos from the farms. With a 99.30% accuracy rate, this model was created and produced to the highest standards.

We used 310,917 total learnable weights and 0 total bias as the parameters in the experiments to construct this model. ReLu activation function for the hidden layer, SoftMax activation function for the fully connected layer, min-batch gradient descent optimization algorithm, sparse categorical cross-entropy loss function, 3x3 filters, stride 1, epochs 65, batch size 32, learning rate 0.001, and images in one batch/iterations are 125 in each epoch for both RGB and grayscale images are the hyperparameters that are used in our experiments.

Five maximum pooling layers, three fully connected layers, including output layers, and five convolutional layers are incorporated. 126x126x32=508,032 neurons make up the first convolutional layer; 61x61x64 = 238,144 neurons make up the second; 28x28x64 = 50, 178 neurons make up the third; 12x12x128 = 18,432 neurons make up the fourth; and 4x4x128 = 2,048 neurons make up the fifth convolutional layer. There are 64 and 512 neurons in fully linked layers. Five neurons in the output layer in our study classify five classes.

5.2. Recommendation and Future Work

In a vast and complex real-world area, the researcher would like to suggest that the results of this study would be more accurate if they were applied to different crops pests, and diseases that can be distinguished from one another by the types of crops. To advance this research, particularly in Ethiopia, farmers, pathologists, agricultural extension agents, and other governmental and nongovernmental organizations must be available to assist the agricultural sector in transitioning to intelligent farming and agricultural practices. Finally, we advise agriculture colleges and institutes to compile all required crop datasets containing both single and dual disease symptoms to assist researchers in studying machine learning, artificial intelligence, and deep learning in agriculture. This will enable farmers to transition from conventional to intelligent farming systems.

Crop disease detection is a broad field that involves examining different leaf classes and crop sections. Our next study should thus focus heavily on identifying the pests and diseases that harm other plants as well as the Enset crop. The fundamental components of our future work will be:

- Extra improvements to its classes and dataset for numerous leaf, stem, and root diseases.
- The researcher intends to use deep learning to produce real-time apps for web, mobile, and embedded devices in the future.
- Making far more progress in identifying diseases in all plants used for food and profit.
- Establishing a close link with precision agriculture and smart agricultural technology to enhance sustainability through disease detection.
- Creating a mobile application with the ability to take pictures of enset leaves, analyze them with a developed model, and offer management advice for diseases.
- Developing a real-time detection system that can assess photos of enset leaves in real-time and deliver a prompt analysis of the presence or absence of disease.

5.3. Contribution

By designing and developing a novel model that employs deep learning techniques like CNN to more accurately diagnose Enset stem and leaf diseases, our study benefits both the scientific community and the public. To effectively accomplish these goals, we ran several tests using the suggested model and pre-trained models. For the benefit of farmers, researchers, and professionals, this study provides the following findings:

- A novel CNN-based model for detecting Enset stem and leaf diseases has been created.
- When the researcher tested our new model, the outcome was excellent.
- Pre-trained models like MobileNetV3Small and EfficientNetB7 have been compared with developed models.
- Its optimal performance over pre-trained models was discussed by the researcher.
- Developed a deep learning model using convolutional neural networks (CNNs) for detecting and classifying Enset diseases.
- Collected and annotated a large dataset of images of Enset plants affected by different diseases for training the model.
- Proposed a novel approach for data augmentation to improve the model's performance on limited data.
- Conducted experiments to evaluate the model's accuracy and compared its performance with traditional machine learning algorithms.
- Collaborated with agricultural experts and farmers to validate the model's effectiveness in detecting and classifying Enset diseases in real-world scenarios.

Reference

[1] T. Dilebo, T. Feyissa, Z. Asfaw, and A. Zewdu, "On-farm diversity, use pattern, and conservation of enset (Ensete ventricosum) genetic resources in southern Ethiopia," *J. Ethnobiol. Ethnomed.*, vol. 19, no. 1, pp. 1–18, 2023, doi: 10.1186/s13002-022-00569-x.

[2] B. Garedew, A. Ayiza, B. Haile, and H. Kasaye, "Habtamu Kasaye. Indigenous Knowledge of Enset (Ensete ventricosum (Welw.) Cheesman) Cultivation and Management Practice by Shekicho People, Southwest Ethiopia," *Belachew Garedew, Aklilu Ayiza*, vol. 5, no. 1, pp. 6–18, 2017, doi: 10.11648/j.jps.20170501.12.

[3] M. Sahle, O. Saito, and S. Demissew, "Exploring the multiple contributions of enset (Ensete ventricosum) for sustainable management of home garden agroforestry system in Ethiopia," *Curr. Res. Environ. Sustain.*, vol. 3, p. 100101, 2021, doi: 10.1016/j.crsust.2021.100101.

[4] T. Y. BIZA, "Enset (Enset ventricosum) Plant Disease and pests Identification Using Image Processing and Deep Convolutional Neural Network.," pp. 0–11, 2021, [Online]. Available: https://repository.ju.edu.et/handle/123456789/6206

[5] R. J. Quinlan, M. B. Quinlan, S. Dira, M. Caudell, A. Sooge, and A. A. Assoma, "Vulnerability and resilience of Sidama Enset and Maize Farms in Southwestern Ethiopia," *J. Ethnobiol.*, vol. 35, no. 2, pp. 314–336, 2015, doi: 10.2993/etbi-35-02-314-336.1.

[6] G. Sakkarvarthi, G. W. Sathianesan, V. S. Murugan, A. J. Reddy, P. Jayagopal, and M. Elsisi, "Detection and Classification of Tomato Crop Disease Using Convolutional Neural Network," *Electron.*, vol. 11, no. 21, Nov. 2022, doi: 10.3390/electronics11213618.

[7] K. Patil, "Leaf Disease Detection using Deep Learning Algorithm," *Int. J. Eng. Adv. Technol.*, vol. 9, no. 3, pp. 3172–3176, 2020, doi: 10.35940/ijeat.c5965.029320.

[8] G. Chopra and P. Whig, "Analysis of Tomato Leaf Disease Identification Techniques," *J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 98–103, 2021, doi: 10.36596/jcse.v2i2.171.

[9] S. Jana, A. R. Begum, and S. Selvaganesan, "Design and Analysis of Pepper Leaf Disease

Detection Using Deep Belief Network," *Eur. J. Mol. Clin. Med.*, vol. 7, no. 9, pp. 1724–1731, 2020, [Online]. Available: https://ejmcm.com/article_5384.html

[10] A. ÖZCAN and E. DÖNMEZ, "Bacterial Disease Detection for Pepper Plant by Utilizing Deep Features Acquired from DarkNet-19 CNN Model," *DÜMF Mühendislik Derg.*, pp. 573–579, Sep. 2021, doi: 10.24012/dumf.1001901.

[11] N. K. Trivedi *et al.*, "Early detection and classification of tomato leaf disease using high-performance deep neural network," *Sensors*, vol. 21, no. 23, 2021, doi: 10.3390/s21237987.

[12] G. Gezahegn and F. Mekbib, "Evaluation of Enset (*Ensete ventricosum* (Welw.) Cheesman) Clone Suckers to Bacterial Wilt Disease Pathogen under Greenhouse Condition," vol. 6, no. 21, pp. 51–56, 2016, [Online]. Available: www.iiste.org

[13] G. Yemata, "Ensete ventricosum: A Multipurpose Crop against Hunger in Ethiopia," *Sci. World J.*, vol. 2020, 2020, doi: 10.1155/2020/6431849.

[14] A. Tsegayc, "Identification a nd Analysis of Nutritional Components of Enset (Ensete velltricoslIIll) Landraces Suitable for Corm Production in Southern Ethiopia," no. 2, 2015.

[15] M. Aytenfsu and B. Haile, "Distribution and Management of Bacterial Wilt (Xanthomonas Campestris pv. Musacearum) of Enset (Ensete Ventricosum) in Ethiopia: a Review," *Online) Int. J. Res. Agric. For.*, vol. 7, no. 2, pp. 40–53, 2020.

[16] T. Addis, F. Azerefegne, and G. Blomme, "Density and distribution on enset root mealybugs on enset," *African Crop Sci. J.*, vol. 16, no. 1, pp. 67–74, 2010, doi: 10.4314/acsj.v16i1.54344.

[17] H. Gudisa Megersa and D. Tolessa Lemma, "Integrated Major Pest Management Systems of Enset (<i>Ensete ventricosum </i>(Welw.) Cheesman) in Ethiopia," *Am. J. Entomol.*, vol. 6, no. 2, p. 14, 2022, doi: 10.11648/j.aje.20220602.11.

[18] M. W. Gardner and J. B. Kendrick, "Bacterial spot of tomato," *J. Agric. Res.*, vol. 21, pp. 123–156, 1921.

[19] A. Adane, “Emerged Plant Virus Disease in Ethiopian Agriculture: Causes and Control Options,” *Ethiop. J. Agric. Sci.*, vol. 29, no. 1, pp. 39–55, 2019.

[20] T. Daba and M. Shigeta, “Enset (Ensete Ventricosum) Production in Ethiopia: Its Nutritional and Socio-Cultural Values,” *Agric. Food Sci. Res.*, vol. 3, no. 2, pp. 66–74, 2016, doi: 10.20448/journal.512/2016.3.2/512.2.66.74.

[21] V. Wiley and T. Lucas, “Computer Vision and Image Processing: A Paper Review,” *Int. J. Artif. Intell. Res.*, vol. 2, no. 1, p. 22, 2018, doi: 10.29099/ijair.v2i1.42.

[22] D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, and S. Pandya, “Cnn7.Pdf,” pp. 1–28, 2021.

[23] S. Joseph, “Image Processing Techniques and Its Applications : an Overview,” *Ijariie*, vol. 4, no. 3, pp. 2168–2174, 2018, [Online]. Available: http://ijariie.com/AdminUploadPdf/IMAGE_PROCESSING_TECHNIQUES_AND_ITS_APPLICATIONS__AN_OVERVIEW_ijariie8745.pdf

[24] R. Archana and P. S. E. Jeevaraj, *Deep learning models for digital image processing: a review*, vol. 57, no. 1. Springer Netherlands, 2024. doi: 10.1007/s10462-023-10631-z.

[25] V. G. Krishnan, J. Deepa, P. V. Rao, V. Divya, and S. Kaviarasan, “An automated segmentation and classification model for banana leaf disease detection,” *J. Appl. Biol. Biotechnol.*, vol. 10, no. 1, pp. 213–220, 2022, doi: 10.7324/JABB.2021.100126.

[26] D. Kaur and Y. Kaur, “International Journal of Computer Science and Mobile Computing Various Image Segmentation Techniques: A Review,” *Int. J. Comput. Sci. Mob. Comput.*, vol. 3, no. 5, pp. 809–814, 2014.

[27] S. Niyas, S. J. Pawan, M. Anand Kumar, and J. Rajan, “Medical image segmentation with 3D convolutional neural networks: A survey,” *Neurocomputing*, vol. 493, pp. 397–413, 2022, doi: 10.1016/j.neucom.2022.04.065.

[28] V. Anand, “Performance of Induction Motor and BLDC Motor and Design of Induction Motor driven Solar Electric Vehicle (IM-SEV),” *Int. J. Adv. Res. Sci. Commun. Technol.*, vol. 6, no. 1, pp. 1046–1053, 2021, doi: 10.48175/568.

[29] Z. Q. Zhao, P. Zheng, S. T. Xu, and X. Wu, “Object Detection with Deep Learning: A Review,” *IEEE Trans. Neural Networks Learn. Syst.*, vol. 30, no. 11, pp. 3212–3232, 2019, doi: 10.1109/TNNLS.2018.2876865.

[30] M. Zareapoor and S. K. R, “Feature Extraction or Feature Selection for Text Classification: A Case Study on Phishing Email Detection,” *Int. J. Inf. Eng. Electron. Bus.*, vol. 7, no. 2, pp. 60–65, 2015, doi: 10.5815/ijieeb.2015.02.08.

[31] K. L. Narayanan *et al.*, “Banana Plant Disease Classification Using Hybrid Convolutional Neural Network,” *Comput. Intell. Neurosci.*, vol. 2022, 2022, doi: 10.1155/2022/9153699.

[32] A. Hassan, M. Refaat, and A. Hemeida, “Image classification based deep learning: A Review,” *Aswan Univ. J. Sci. Technol.*, vol. 2, no. 1, pp. 11–35, 2022, doi: 10.21608/aujst.2022.259887.

[33] X. Zhao, L. Wang, Y. Zhang, X. Han, M. Deveci, and M. Parmar, *A review of convolutional neural networks in computer vision*, vol. 57, no. 4. Springer Netherlands, 2024. doi: 10.1007/s10462-024-10721-6.

[34] B. Póczos, S. 17, R. Salakhutdinov, J. Bengio, G. Hinton, and Y. Lecun, “Advanced Introduction to Machine Learning, CMU-10715 Deep Learning Credits”, [Online]. Available: https://www.cs.cmu.edu/~epxing/Class/10715/lectures/DeepArchitectures.pdf

[35] A. Voulodimos, N. Doulamis, A. Doulamis, and E. Protopapadakis, “Deep Learning for Computer Vision: A Brief Review,” *Comput. Intell. Neurosci.*, vol. 2018, 2018, doi: 10.1155/2018/7068349.

[36] M. Tan and Q. V. Le, “EfficientNet: Rethinking model scaling for convolutional neural networks,” *36th Int. Conf. Mach. Learn. ICML 2019*, vol. 2019-June, pp. 10691–10700, 2019.

[37] B. Wang and A. Rezaei sofla, “Solution for sports image classification using modified MobileNetV3 optimized by modified battle royal optimization algorithm,” *Heliyon*, vol. 9, no. 11, p. e21603, 2023, doi: 10.1016/j.heliyon.2023.e21603.

[38] R. Sun, “Optimization for deep learning: theory and algorithms,” pp. 1–60, 2019, [Online]. Available: http://arxiv.org/abs/1912.08957

[39] R. V, “Crop Diseases and Pest Detection using Deep Learning and Image Processing Techniques,” *Int. J. Res. Appl. Sci. Eng. Technol.*, vol. 9, no. VI, pp. 372–380, 2021, doi: 10.22214/ijraset.2021.34915.

[40] M. G. Selvaraj *et al.*, “AI-powered banana diseases and pest detection,” *Plant Methods*, vol. 15, no. 1, pp. 1–11, 2019, doi: 10.1186/s13007-019-0475-z.

[41] V. Chaudhari and M. P. Patil, “Detection and Classification of Banana Leaf Disease Using Novel Segmentation and Ensemble Machine Learning Approach,” *Appl. Comput. Syst.*, vol. 28, no. 1, pp. 92–99, 2023, doi: 10.2478/acss-2023-0009.

[42] K. Karadağ, M. E. Tenekeci, R. Taşaltın, and A. Bilgili, “Detection of pepper fusarium disease using machine learning algorithms based on spectral reflectance,” *Sustain. Comput. Informatics Syst.*, vol. 28, no. 2018, 2020, doi: 10.1016/j.suscom.2019.01.001.

[43] D. Ashebir and G. Tadesse, “BWENet: Detection and Grading of Bacterial Wilt Using Deep Convolutional Neural Network,” *Indian J. Sci. Technol.*, vol. 15, no. 22, pp. 1100–1111, 2022, doi: 10.17485/ijst/v15i22.634.

[44] “Anaconda.” https://github.com/WeilerWebServices/Anaconda

[45] J. G. A. Barbedo, “Factors influencing the use of deep learning for plant disease recognition,” *Biosyst. Eng.*, vol. 172, pp. 84–91, 2018, doi: 10.1016/j.biosystemseng.2018.05.013.

[46] K. Maharana, S. Mondal, and B. Nemade, “A review: Data pre-processing and data augmentation techniques,” *Glob. Transitions Proc.*, vol. 3, no. 1, pp. 91–99, 2022, doi: 10.1016/j.gltp.2022.04.020.

[47] Y. Ho and S. Wookey, “The Real-World-Weight Cross-Entropy Loss Function: Modeling the Costs of Mislabeling,” *IEEE Access*, vol. 8, pp. 4806–4813, 2020, doi: 10.1109/ACCESS.2019.2962617.

[48] L. Alzubaidi *et al.*, *Review of deep learning: concepts, CNN architectures, challenges, applications, future directions*, vol. 8, no. 1. Springer International Publishing, 2021. doi: 10.1186/s40537-021-00444-8.

APPENDICES

Appendices I: The Python code that shows the EnsetDNet Model

```
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dropout
from keras.layers import Dense, Dropout
import logging
import datetime
import json
model = Sequential()
#Adds a convolutional layer with 32 filters and a 3x3 kernel, ReLU activation, and input shape of (128, 128, 3).
#Adds a max-pooling layer with a 2x2 pool size.
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
# Adds another convolutional layer with 128 filters and a 3x3 kernel, ReLU activation.
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
#Adds a flattening layer to convert the 2D output to a 1D vector for the fully connected layers.
model.add(Flatten())

#Adds a dense layer with 64 neurons and ReLU activation.
#Adds the output layer with 5 neurons (assuming 5 classes) and softmax activation for multiclass classification.
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(5, activation='softmax'))
model.add(Dropout(0.5))
learning_rate = 0.001
optimizer = tf.keras.optimizers.Nadam(learning_rate)
model.compile(optimizer= 'Nadam', loss='categorical_crossentropy', metrics=['accuracy'])
# Creates an ImageDataGenerator for preprocessing images without data augmentation, rescaling pixel values to the range [0, 1].
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2,
                             horizontal_flip=False,
                             vertical_flip=False,
                             rotation_range=False)
train_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
```

Appendices II: The Python code that shows Data augmentation

Augmentation Techniques

```
    # Data Augmentation
from flask import Flask, render_template, Response
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator

app = Flask(__name__)

data_dir = 'models/original/'
output_dir = 'models/original Outpout/'

threshold = 1000000

@app.route('/')
def index():
    return render_template('agument.html', data_dir=data_dir, output_dir=output_dir)

@app.route('/augment_images')
def augment_images():
    data_generator = ImageDataGenerator(rescale=1./255, rotation_range=20, width_shift_range=0.2, height_shift_range=0.2, shear_range=0.2, zoom_range=0.2,
horizontal_flip=True, fill_mode='nearest')

    total_images = sum(len(files) for _, _, files in os.walk(data_dir))
    processed_images = 0

    def generate_augmented_images():
        nonlocal processed_images
        for root, _, files in os.walk(data_dir):
            output_subdir = os.path.join(output_dir, 'augmented_' + os.path.relpath(root, data_dir))
            if not os.path.exists(output_subdir):
                os.makedirs(output_subdir)

        for root, _, files in os.walk(data_dir):
            output_subdir = os.path.join(output_dir, 'augmented_' + os.path.relpath(root, data_dir))


        for root, _, files in os.walk(data_dir):
            output_subdir = os.path.join(output_dir, 'augmented_' + os.path.relpath(root, data_dir))

            subdir_size = sum(os.path.getsize(os.path.join(root, f)) for f in files)
            if subdir_size < threshold:
                iterations = 20
                rotation_range = 20
                width_shift_range = 0.3
                height_shift_range = 0.3
                zoom_range = 0.3
            else:
                iterations = 5
                rotation_range = 20
                width_shift_range = 0.2
                height_shift_range = 0.2
                zoom_range = 0.2

            data_generator = ImageDataGenerator(rescale=1./255, rotation_range=rotation_range, width_shift_range=width_shift_range,
height_shift_range=height_shift_range, shear_range=0.2, zoom_range=zoom_range, horizontal_flip=True, vertical_flip=True, brightness_range=(0.8, 1.2),
fill_mode='nearest')

            for img_name in files:
                img_path = os.path.join(root, img_name)
                img = tf.keras.preprocessing.image.load_img(img_path)
                x = tf.keras.preprocessing.image.img_to_array(img)
                x = np.expand_dims(x, axis=0)
                i = 0
                try:
                    for batch in data_generator.flow(x, batch_size=1, save_to_dir=output_subdir, save_prefix='augmented_' + img_name.split('.')[0],
save_format='jpeg'):
```

Appendices II: The Python code that converts RGB into Gray Scale

```python
import os
from PIL import Image
from tqdm import tqdm

def convert_to_grayscale(source_folder, destination_folder):
    if not os.path.exists(destination_folder):
        os.makedirs(destination_folder)

    image_paths = []

    for subdir, dirs, files in os.walk(source_folder):
        for file in files:
            if file.lower().endswith(('.png', '.jpg', '.jpeg', '.tiff', '.bmp', '.gif')):
                image_paths.append(os.path.join(subdir, file))

    for image_path in tqdm(image_paths, desc='EnsetDNet Converting images'):
        dest_path = image_path.replace(source_folder, destination_folder)
        dest_subdir = os.path.dirname(dest_path)
        if not os.path.exists(dest_subdir):
            os.makedirs(dest_subdir)
        with Image.open(image_path) as img:
            grayscale_img = img.convert('L')
            grayscale_img.save(dest_path)

source_dir = 'models/original'
destination_dir = 'models/grayOriginal'

convert_to_grayscale(source_dir, destination_dir)
```

Appendices III: The Python code for MobileNetv3 Small

```
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV3Small

#Load the MobileNetV3
mobile_NetV3 = MobileNetV3Small(weights='imagenet')

# Load the EfficientNet model
MobileNetV3Small_model = MobileNetV3Small(weights='imagenet', include_top=False, input_shape=(224, 224, 3))
for layer in MobileNetV3Small_model.layers:
    layer.trainable = True

model_MobileNetV3Small = tf.keras.Sequential()
model_MobileNetV3Small.add(MobileNetV3Small_model)
model_MobileNetV3Small.add(tf.keras.layers.Flatten())
model_MobileNetV3Small.add(tf.keras.layers.Dense(64, activation='relu'))
model_MobileNetV3Small.add(tf.keras.layers.Dense(5, activation='softmax'))

model_MobileNetV3Small.compile(optimizer='nadam', loss='categorical_crossentropy', metrics=['accuracy'])
# Load the original custom model
model_original = load_model('model_original.keras')

# Data generators and training/validation setup (code omitted for brevity)
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2,
                             horizontal_flip=False,
                             vertical_flip=False,
                             rotation_range=False)
train_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)

validation_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
test_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)
model_MobileNetV3Small.fit(train_generator, epochs=65, validation_data=validation_generator)

# Train the original custom model
model_original.fit(train_generator, epochs=65, validation_data=validation_generator)

# Evaluate the models on the test set
test_loss_MobileNetV3Small, test_acc_MobileNetV3Small = model_MobileNetV3Small.evaluate(test_generator)
# test_loss_original, test_acc_original = model_original.evaluate(test_generator)

print(f'MobileNetV3Small Test Accuracy: {test_acc_MobileNetV3Small}')
print(f'MobileNetV3Small Test Loss: {test_loss_MobileNetV3Small}')
```

Appendices IV: The Python code for EfficientNet B7

```
import tensorflow as tf
from tensorflow.keras.models import load_model
from tensorflow.keras.applications import EfficientNetB7
from tensorflow.keras.preprocessing.image import ImageDataGenerator

#Load the EfficicientNet B7
efficientnet_model = EfficientNetB7(weights='imagenet')

# Load the EfficientNet model
efficientnet_model = EfficientNetB7(weights='imagenet', include_top = False, input_shape=(224, 224, 3))
# efficientnet_model = EfficientNetB7(weights='imagenet', include_top = False, input_shape=(224, 224, 3))

for layer in efficientnet_model.layers:
    layer.trainable = True

efficientnet_model = tf.keras.Sequential()
# EfficientNetB7.add(efficientnet_model)
efficientnet_model.add(tf.keras.layers.Flatten())
efficientnet_model.add(tf.keras.layers.Dense(64, activation='relu'))
efficientnet_model.add(tf.keras.layers.Dense(5, activation='softmax'))

efficientnet_model.compile(optimizer='nadam', loss='categorical_crossentropy', metrics=['accuracy'])

# Load the original custom model
model_original = load_model('model_original.keras')

# Data generators and training/validation setup (code omitted for brevity)
datagen = ImageDataGenerator(rescale=1./255, validation_split=0.2,
                             horizontal_flip=False,
                             vertical_flip=False,
                             rotation_range=False)

train_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='training'
)
validation_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical',
    subset='validation'
)
test_generator = datagen.flow_from_directory(
    'models/original',
    target_size=(128, 128),
    batch_size=32,
    class_mode='categorical'
)
# Train the EfficientNet model
efficientnet_model.fit(train_generator, epochs= 65, validation_data=validation_generator)

# Train the original custom model
model_original.fit(train_generator, epochs=65, validation_data=validation_generator)

# Evaluate the models on the test set
test_loss_EfficientNetB7, test_acc_EfficientNetB7 = efficientnet_model.evaluate(test_generator)
test_loss_original, test_acc_original = model_original.evaluate(test_generator)

print(f'EfficientNetB7 Test Accuracy: {test_acc_EfficientNetB7}')
print(f'EfficientNetB7 Test Loss: {test_loss_EfficientNetB7}')
```

Appendices V: History of Developed Model Fitting

```
Epoch 1/65
125/125 [==============================] - 157s 820ms/step - loss: 1.4917
- accuracy: 0.3650 - val_loss: 1.3320 - val_accuracy: 0.4627
Epoch 2/65
125/125 [==============================] - 163s 869ms/step - loss: 1.1392
- accuracy: 0.5628 - val_loss: 0.8868 - val_accuracy: 0.6990
Epoch 3/65
125/125 [==============================] - 153s 816ms/step - loss: 0.8593
- accuracy: 0.6940 - val_loss: 0.6682 - val_accuracy: 0.7650
Epoch 4/65
125/125 [==============================] - 178s 945ms/step - loss: 0.6866
- accuracy: 0.7590 - val_loss: 0.5844 - val_accuracy: 0.7872
Epoch 5/65
125/125 [==============================] - 150s 800ms/step - loss: 0.5953
- accuracy: 0.7853 - val_loss: 0.5818 - val_accuracy: 0.7822
Epoch 6/65
125/125 [==============================] - 160s 851ms/step - loss: 0.5230
- accuracy: 0.8118 - val_loss: 0.4153 - val_accuracy: 0.8525
Epoch 7/65
125/125 [==============================] - 165s 903ms/step - loss: 0.4603
- accuracy: 0.8340 - val_loss: 0.4231 - val_accuracy: 0.8485
Epoch 8/65
125/125 [==============================] - 151s 806ms/step - loss: 0.4091
- accuracy: 0.8523 - val_loss: 0.3294 - val_accuracy: 0.8798
Epoch 9/65
125/125 [==============================] - 145s 774ms/step - loss: 0.3824
- accuracy: 0.8650 - val_loss: 0.3206 - val_accuracy: 0.8857
Epoch 10/65
125/125 [==============================] - 148s 786ms/step - loss: 0.3481
```

```
- accuracy: 0.8727 - val_loss: 0.2765 - val_accuracy: 0.8965
Epoch 11/65
125/125 [==============================] - 149s 792ms/step - loss: 0.3111
- accuracy: 0.8945 - val_loss: 0.2155 - val_accuracy: 0.9217
Epoch 12/65
125/125 [==============================] - 147s 781ms/step - loss: 0.2824
- accuracy: 0.9005 - val_loss: 0.2807 - val_accuracy: 0.8955
Epoch 13/65
125/125 [==============================] - 149s 794ms/step - loss: 0.2452
- accuracy: 0.9135 - val_loss: 0.2769 - val_accuracy: 0.8987
Epoch 14/65
125/125 [==============================] - 152s 809ms/step - loss: 0.2310
- accuracy: 0.9165 - val_loss: 0.2212 - val_accuracy: 0.9192
Epoch 15/65
125/125 [==============================] - 149s 794ms/step - loss: 0.2114
- accuracy: 0.9255 - val_loss: 0.3676 - val_accuracy: 0.8873
Epoch 16/65
125/125 [==============================] - 150s 800ms/step - loss: 0.2155
- accuracy: 0.9275 - val_loss: 0.1938 - val_accuracy: 0.9317
Epoch 17/65
125/125 [==============================] - 177s 943ms/step - loss: 0.1825
- accuracy: 0.9360 - val_loss: 0.1780 - val_accuracy: 0.9345
Epoch 18/65
125/125 [==============================] - 172s 916ms/step - loss: 0.1559
- accuracy: 0.9455 - val_loss: 0.2193 - val_accuracy: 0.9237
Epoch 19/65
125/125 [==============================] - 151s 801ms/step - loss: 0.1952
- accuracy: 0.9337 - val_loss: 0.1768 - val_accuracy: 0.9417
Epoch 20/65
125/125 [==============================] - 147s 783ms/step - loss: 0.1515
- accuracy: 0.9485 - val_loss: 0.1403 - val_accuracy: 0.9518
```

```
Epoch 21/65
125/125 [==============================] - 147s 784ms/step - loss: 0.1545
- accuracy: 0.9467 - val_loss: 0.1319 - val_accuracy: 0.9538
Epoch 22/65
125/125 [==============================] - 147s 783ms/step - loss: 0.1732
- accuracy: 0.9427 - val_loss: 0.1074 - val_accuracy: 0.9648
Epoch 23/65
125/125 [==============================] - 147s 784ms/step - loss: 0.1371
- accuracy: 0.9503 - val_loss: 0.1190 - val_accuracy: 0.9563
Epoch 24/65
125/125 [==============================] - 147s 781ms/step - loss: 0.1171
- accuracy: 0.9575 - val_loss: 0.0888 - val_accuracy: 0.9650
Epoch 25/65
125/125 [==============================] - 145s 774ms/step - loss: 0.1304
- accuracy: 0.9542 - val_loss: 0.1152 - val_accuracy: 0.9615
Epoch 26/65
125/125 [==============================] - 148s 786ms/step - loss: 0.1101
- accuracy: 0.9605 - val_loss: 0.1044 - val_accuracy: 0.9620
Epoch 27/65
125/125 [==============================] - 149s 792ms/step - loss: 0.1218
- accuracy: 0.9612 - val_loss: 0.1391 - val_accuracy: 0.9512
Epoch 28/65
125/125 [==============================] - 146s 775ms/step - loss: 0.1223
- accuracy: 0.9587 - val_loss: 0.0945 - val_accuracy: 0.9628
Epoch 29/65
125/125 [==============================] - 145s 771ms/step - loss: 0.1013
- accuracy: 0.9642 - val_loss: 0.0810 - val_accuracy: 0.9750
Epoch 30/65
125/125 [==============================] - 146s 775ms/step - loss: 0.0918
- accuracy: 0.9652 - val_loss: 0.0753 - val_accuracy: 0.9743
```

```
Epoch 31/65
125/125 [==============================] - 146s 778ms/step - loss: 0.0999
- accuracy: 0.9668 - val_loss: 0.2651 - val_accuracy: 0.9212
Epoch 32/65
125/125 [==============================] - 147s 780ms/step - loss: 0.0953
- accuracy: 0.9653 - val_loss: 0.0608 - val_accuracy: 0.9783
Epoch 33/65
125/125 [==============================] - 146s 776ms/step - loss: 0.1055
- accuracy: 0.9643 - val_loss: 0.1181 - val_accuracy: 0.9565
Epoch 34/65
125/125 [==============================] - 146s 775ms/step - loss: 0.0747
- accuracy: 0.9745 - val_loss: 0.0551 - val_accuracy: 0.9817
Epoch 35/65
125/125 [==============================] - 146s 775ms/step - loss: 0.0959
- accuracy: 0.9678 - val_loss: 0.1199 - val_accuracy: 0.9560
Epoch 36/65
125/125 [==============================] - 145s 771ms/step - loss: 0.1001
- accuracy: 0.9692 - val_loss: 0.0618 - val_accuracy: 0.9778
Epoch 37/65
125/125 [==============================] - 144s 766ms/step - loss: 0.0866
- accuracy: 0.9728 - val_loss: 0.0450 - val_accuracy: 0.9850
Epoch 38/65
125/125 [==============================] - 145s 773ms/step - loss: 0.0801
- accuracy: 0.9737 - val_loss: 0.0630 - val_accuracy: 0.9785
Epoch 39/65
125/125 [==============================] - 144s 769ms/step - loss: 0.0742
- accuracy: 0.9762 - val_loss: 0.0694 - val_accuracy: 0.9765
Epoch 40/65
125/125 [==============================] - 146s 779ms/step - loss: 0.0840
- accuracy: 0.9735 - val_loss: 0.0531 - val_accuracy: 0.9798
```

```
Epoch 41/65
125/125 [==============================] - 145s 774ms/step - loss: 0.0764
- accuracy: 0.9748 - val_loss: 0.0399 - val_accuracy: 0.9863
Epoch 42/65
125/125 [==============================] - 145s 772ms/step - loss: 0.0713
- accuracy: 0.9768 - val_loss: 0.0512 - val_accuracy: 0.9823
Epoch 43/65
125/125 [==============================] - 144s 766ms/step - loss: 0.0627
- accuracy: 0.9783 - val_loss: 0.0696 - val_accuracy: 0.9753
Epoch 44/65
125/125 [==============================] - 144s 767ms/step - loss: 0.0651
- accuracy: 0.9765 - val_loss: 0.0332 - val_accuracy: 0.9898
Epoch 45/65
125/125 [==============================] - 145s 765ms/step - loss: 0.0654
- accuracy: 0.9775 - val_loss: 0.0514 - val_accuracy: 0.9825
Epoch 46/65
125/125 [==============================] - 145s 771ms/step - loss: 0.0671
- accuracy: 0.9788 - val_loss: 0.0477 - val_accuracy: 0.9833
Epoch 47/65
125/125 [==============================] - 144s 769ms/step - loss: 0.0624
- accuracy: 0.9787 - val_loss: 0.0315 - val_accuracy: 0.9908
Epoch 48/65
125/125 [==============================] - 145s 771ms/step - loss: 0.0625
- accuracy: 0.9797 - val_loss: 0.0413 - val_accuracy: 0.9852
Epoch 49/65
125/125 [==============================] - 146s 777ms/step - loss: 0.0609
- accuracy: 0.9792 - val_loss: 0.0514 - val_accuracy: 0.9840
Epoch 50/65
125/125 [==============================] - 144s 768ms/step - loss: 0.0594
- accuracy: 0.9795 - val_loss: 0.0420 - val_accuracy: 0.9858
```

```
Epoch 51/65
125/125 [==============================] - 154s 820ms/step - loss: 0.0822
- accuracy: 0.9750 - val_loss: 0.0571 - val_accuracy: 0.9820
Epoch 52/65
125/125 [==============================] - 143s 761ms/step - loss: 0.0683
- accuracy: 0.9767 - val_loss: 0.0379 - val_accuracy: 0.9858
Epoch 53/65
125/125 [==============================] - 146s 777ms/step - loss: 0.0567
- accuracy: 0.9798 - val_loss: 0.0279 - val_accuracy: 0.9908
Epoch 54/65
125/125 [==============================] - 149s 793ms/step - loss: 0.0565
- accuracy: 0.9830 - val_loss: 0.0697 - val_accuracy: 0.9817
Epoch 55/65
125/125 [==============================] - 149s 792ms/step - loss: 0.0665
- accuracy: 0.9780 - val_loss: 0.0545 - val_accuracy: 0.9818
Epoch 56/65
125/125 [==============================] - 146s 776ms/step - loss: 0.0499
- accuracy: 0.9843 - val_loss: 0.1749 - val_accuracy: 0.9427
Epoch 57/65
125/125 [==============================] - 147s 780ms/step - loss: 0.0560
- accuracy: 0.9805 - val_loss: 0.0619 - val_accuracy: 0.9788
Epoch 58/65
125/125 [==============================] - 145s 765ms/step - loss: 0.0567
- accuracy: 0.9797 - val_loss: 0.0463 - val_accuracy: 0.9833
Epoch 59/65
125/125 [==============================] - 144s 765ms/step - loss: 0.0526
- accuracy: 0.9822 - val_loss: 0.0917 - val_accuracy: 0.9692
Epoch 60/65
125/125 [==============================] - 144s 767ms/step - loss: 0.0507
- accuracy: 0.9848 - val_loss: 0.0414 - val_accuracy: 0.9860
```

```
Epoch 61/65
125/125 [==============================] - 144s 766ms/step - loss: 0.0655
- accuracy: 0.9785 - val_loss: 0.0416 - val_accuracy: 0.9863
Epoch 62/65
125/125 [==============================] - 144s 766ms/step - loss: 0.0596
- accuracy: 0.9803 - val_loss: 0.0271 - val_accuracy: 0.9915
Epoch 63/65
125/125 [==============================] - 144s 768ms/step - loss: 0.0636
- accuracy: 0.9805 - val_loss: 0.0380 - val_accuracy: 0.9880
Epoch 64/65
125/125 [==============================] - 143s 763ms/step - loss: 0.0593
- accuracy: 0.9815 - val_loss: 0.0805 - val_accuracy: 0.9738
Epoch 65/65
125/125 [==============================] - 143s 764ms/step - loss: 0.0413
- accuracy: 0.9892 - val_loss: 0.0121 - val_accuracy: 0.9930
```