Image-Based Coffee Plant Disease Detection using Transfer Learning

Kedir Gemechu Waritu

A Thesis Submitted to Department of Computer Science

College of Computing

Presented in Partial Fulfillment of the Requirements for the Master's in Computer Science

Office Of Graduate Studies

Madda Walabu University

October, 2023

Bale-Robe, Ethiopia

Image-Based Coffee Plant Disease Detection using Transfer Learning

Kedir Gemechu Waritu

Advisor: Dr. Sinatyehu Hirpassa (Ph.D.)

A Thesis Submitted to Department of Computer Science

College of Computing

Presented in Partial Fulfillment of the Requirements for the Master's in Computer Science

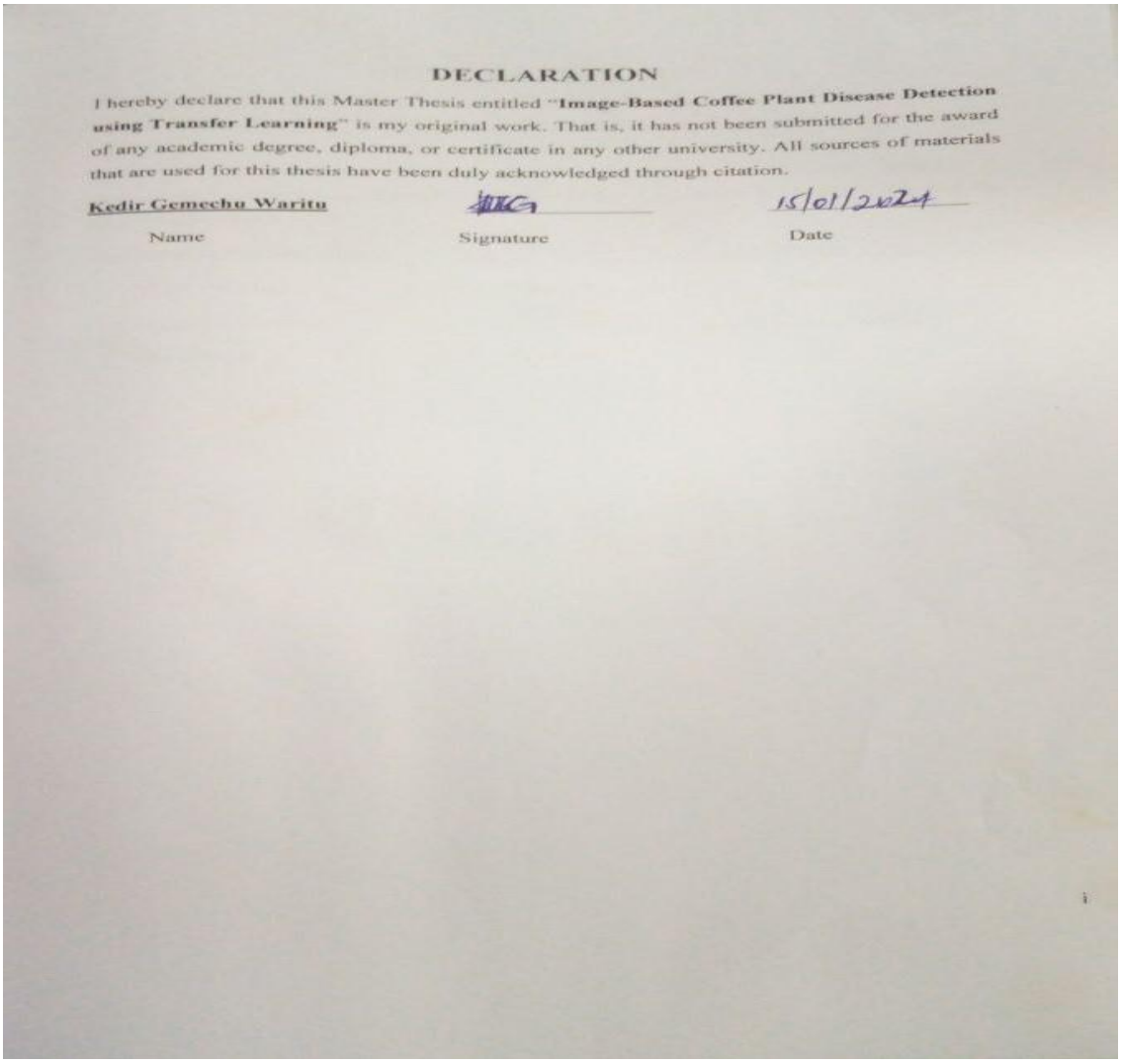
Office Of Graduate Studies

Madda Walabu University

October, 2023

Bale-Robe, Ethiopia

DECLARATION



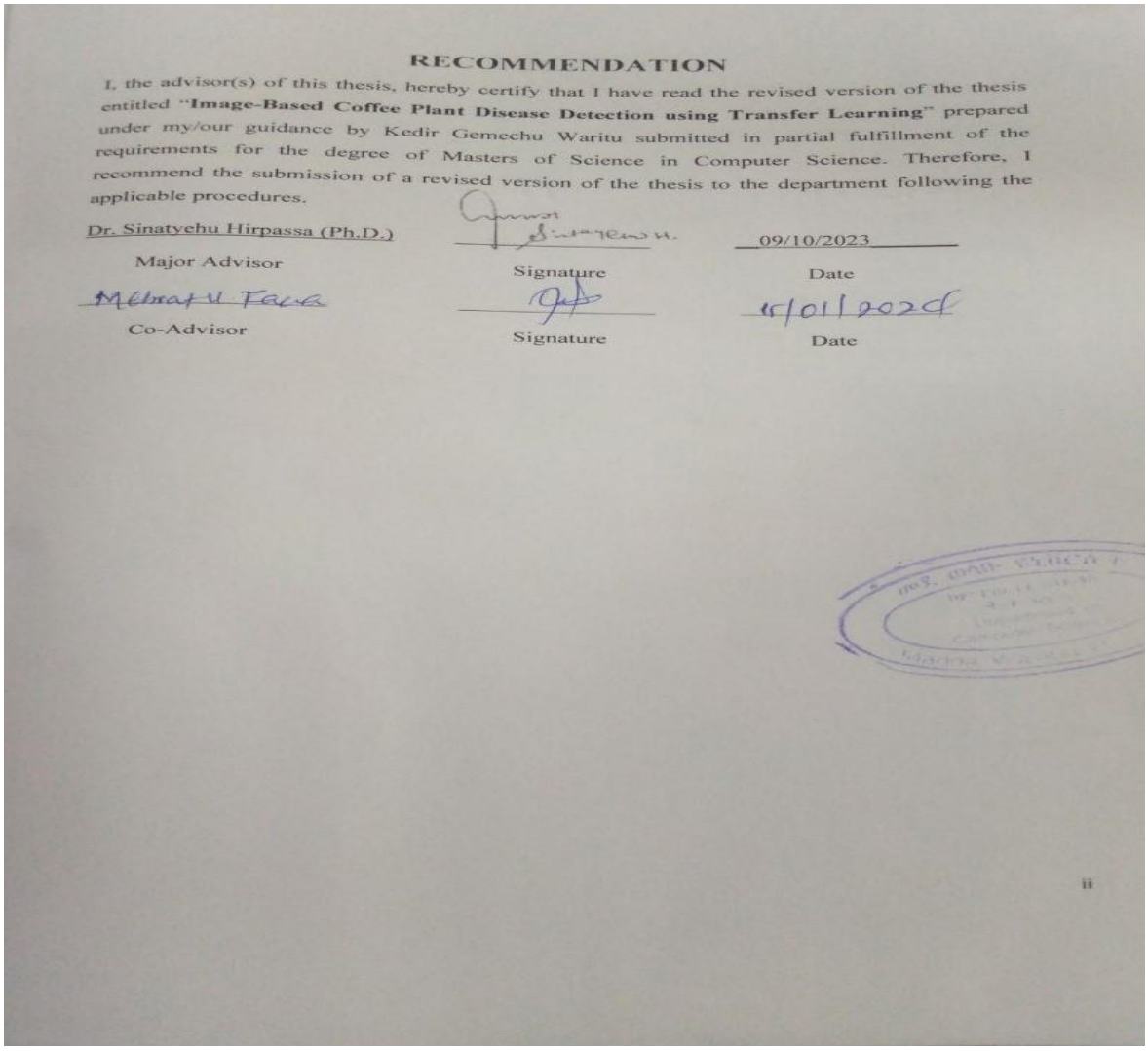
DECLARATION

I hereby declare that this Master Thesis entitled "**Image-Based Coffee Plant Disease Detection using Transfer Learning**" is my original work. That is, it has not been submitted for the award of any academic degree, diploma, or certificate in any other university. All sources of materials that are used for this thesis have been duly acknowledged through citation.

Kedir Gemechu Waritu | | 15/01/2024

Name | Signature | Date

RECOMMENDATION



RECOMMENDATION

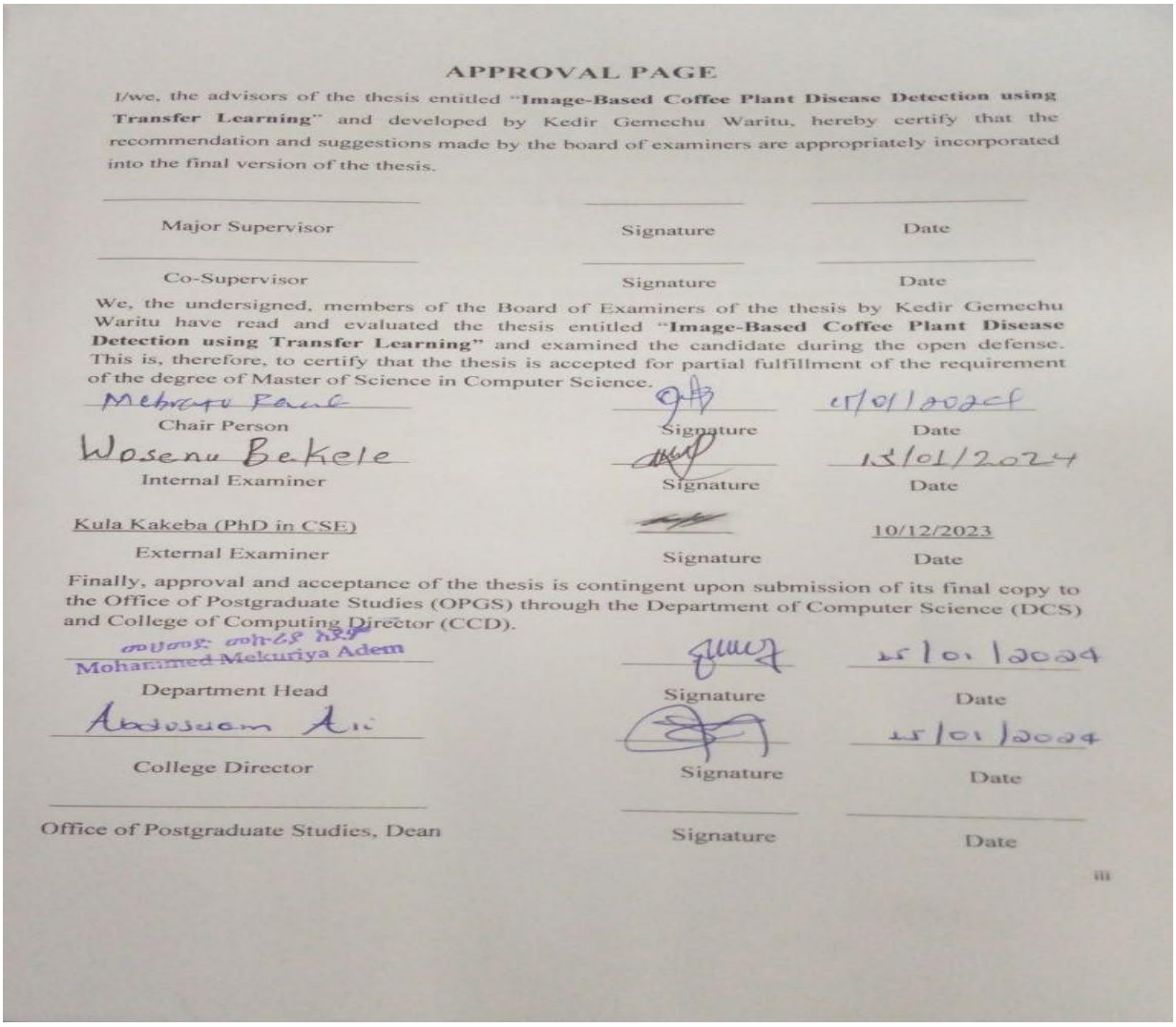
I, the advisor(s) of this thesis, hereby certify that I have read the revised version of the thesis entitled "**Image-Based Coffee Plant Disease Detection using Transfer Learning**" prepared under my/our guidance by Kedir Gemechu Waritu submitted in partial fulfillment of the requirements for the degree of Masters of Science in Computer Science. Therefore, I recommend the submission of a revised version of the thesis to the department following the applicable procedures.

Dr. Sinatyehu Hirpassa (Ph.D.) — Major Advisor — Signature — 09/10/2023 — Date

Mekonnen H. Tana — Co-Advisor — Signature — 15/01/2024 — Date

ii

APPROVAL PAGE



APPROVAL PAGE

I/we, the advisors of the thesis entitled "**Image-Based Coffee Plant Disease Detection using Transfer Learning**" and developed by Kedir Gemechu Waritu, hereby certify that the recommendation and suggestions made by the board of examiners are appropriately incorporated into the final version of the thesis.

Major Supervisor	Signature	Date
Co-Supervisor	Signature	Date

We, the undersigned, members of the Board of Examiners of the thesis by Kedir Gemechu Waritu have read and evaluated the thesis entitled "**Image-Based Coffee Plant Disease Detection using Transfer Learning**" and examined the candidate during the open defense. This is, therefore, to certify that the thesis is accepted for partial fulfillment of the requirement of the degree of Master of Science in Computer Science.

Mebratu Fanta Chair Person	Signature	15/01/2024 Date
Wosenu Bekele Internal Examiner	Signature	13/01/2024 Date
Kula Kakeba (PhD in CSE) External Examiner	Signature	10/12/2023 Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department of Computer Science (DCS) and College of Computing Director (CCD).

Mohammed Mekuriya Adem Department Head	Signature	15/01/2024 Date
Abdusuam Ali College Director	Signature	15/01/2024 Date
Office of Postgraduate Studies, Dean	Signature	Date

iii

ACKNOWLEDGMENT

I am very grateful to Allah (Waaqa Uumaa) for His blessings and guidance through the thesis-writing process. I am particularly grateful to Dr. Sinatyehu Hirpassa, my adviser, for his important advice, essential comments, and continuous support. His knowledge and experience were critical to the achievement of this thesis. I am especially thankful for our several communication conversations, which have always pushed me to think further about my study.

Finally, I would like to express my sincere gratitude to my family, especially my brother Aliyi, and my friends for their love and encouragement. I could not have completed this thesis without their unwavering support. I am truly grateful for all of the help and guidance they have provided me throughout this journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYMS AND ABBREVIATIONS

AI	Artificial Intelligence
ANN	Artificial Neural Network
BN	Batch Normalization
BES	Brown Eye Spot
CBD	Coffee Berry Disease
CLR	Coffee Leaf Rust
CNN	Convolutional Neural Network
CNTK	Cognitive Toolkit
CWD	Coffee Wilt Disease
DL	Deep Learning
FC	Fully Connected
FP	False Positive
FN	False Negative
GAN	Generative adversarial network
GPU	Graphics Processing Unit
KNN	K-Nearest Neighbor
PLS	Phoma Leaf Spot
RBF	Radial Basis Function
RGB	Red, Green, Blue
RMSprop	Root Mean Squared Propagation.
SGD	Stochastic Gradient Descent.
SOM	Self-Organizing Map

SVM	Support Vector Machines
TP	True Positive
TPU	Tensor Processing Unit
TN	True Negative
USD	United States dollar
VGG19	Visual Geometry Group 19

ABSTRACT

Agriculture stands as the cornerstone of Ethiopia's economy, accounting for a staggering 80-85% of its GDP. Among the nation's agricultural treasures, coffee holds a special place, serving as a vital export commodity and a source of daily sustenance for countless farmers. The contribution of coffee to Ethiopia's economy is immense, generating substantial foreign exchange earnings and fueling the livelihoods of rural communities. Given the pivotal role of coffee in Ethiopia's economic landscape, ensuring its quality and safeguarding it from diseases takes paramount importance. However, the current practice of relying on manual disease identification by experts faces several limitations. This method is not only time-consuming and labor-intensive but also lacks scalability, making it impractical to cover all coffee-growing regions effectively. Timely detection of coffee diseases is crucial for minimizing damage and preventing significant economic losses. Early intervention can halt the spread of diseases, preserving the health of coffee plants and ensuring a bountiful harvest. Therefore, the development of an automated disease identification system is an urgent necessity for the Ethiopian coffee industry. This paper proposes an automatic coffee disease identification system using transfer learning and recommends treatments in Afan Oromo. The system extracts important features from coffee plants and uses them to classify the plant into five categories: coffee berry disease (CBD), coffee leaf rust (CLR), Phoma leaf spot (PLS), brown eye spot (BES), and healthy coffee. To achieve the best results during the classification of such diseases, we compared training a deep learning model from scratch and using transfer learning with a pre-trained ResNet50 model to classify coffee diseases. The ResNet50 model achieved a training accuracy of 97.6% and a test accuracy of 95%, significantly outperforming the model trained from scratch, which achieved an accuracy of 89.3%.

The system was trained on a dataset of 5010 coffee plant images and achieved an accuracy of 95% with the ResNet50 architecture and 92% with the VGG19 architecture. These results suggest that deep learning algorithms have the potential to be used for automatic and accurate coffee disease identification.

Keywords: *Coffee disease, CNN, Transfer Learning, Resnet50, VGG 19, Inception v3*

CHAPTER ONE

1 INTRODUCTION

1.1 Background of the Study

One of the most significant commodities traded globally is coffee. Ethiopia produces the most coffee in Sub-Saharan Africa and ranks fifth globally after Brazil, Vietnam, Colombia, and Indonesia, producing around 7–10% of the world's total coffee production [1].

Ethiopia is one of the developing countries, and its economy depends on agriculture. Agriculture products are essential for both humans and animals in Ethiopia. The country is suitable for crop production, such as coffee. Coffee is an essential commodity for the Ethiopian economy, and we must increase its productivity and quality. Disease detection in plants is a major task for farmers. Plants are affected by various types of viruses and bacterial attacks, which can lead to decreased coffee production, which in turn can affect the country's economy [2]. Farmers visually inspect their farmland to detect and diagnose diseases. However, this method is time-consuming, expensive, and sometimes inconvenient. Automatic detection using Transfer learning techniques provides fast and accurate results.

Farmers who produce coffee plants face significant economic losses every year due to various diseases that can affect their coffee plants. There are different types of diseases in coffee plant production. If farmers detect these diseases early and apply appropriate treatment, they can save a lot of waste and prevent economic losses for the country. Therefore, we can use deep learning to identify the type of disease in coffee and recommend the appropriate course of action for coffee producers.

Currently, there are many types of diseases reported in Ethiopia caused by fungi, bacteria, nematodes, and viruses. These diseases include Cercospora leaf spot, phoma leaf spot, coffee berry disease, coffee wilt disease, and other common diseases that attack coffee. For example, Brown Eye Spot (BES), a fungal disease caused by Cercospora coffeicola, causes circular brown spots on leaves with gray-brown centers and red-brown edges. Yellow halos may encircle the

spots [3]. Coffee Berry Disease (CBD), caused by the fungal infection Colletotrichum kahawae, causes tiny, water-soaked lesions on young, developing berries. These lesions soon become dark brown or black and become somewhat sunken [4]. Coffee leaf rust (CLR) is a fungal disease caused by Hemileia vastatrix. It is a devastating disease that can destroy coffee crops, especially at their peak growing levels [5]. Hemileia vastatrix, the pathogen that causes coffee leaf rust, infects the leaves of coffee plants, leaving yellow patches or powdery lumps on the underside of the leaves. This decreases the plant's capacity to photosynthesize, which can lead to considerable output reductions. Coffee Wilt Disease (CWD) is a fungal infection caused by Fusarium xylarioides [6]. The leaves turn yellow, fold, and bend inward as early signs of CWD. After the yellow spots or powdery blisters appear on the leaves of coffee plants, the leaves become limp, dry up, turn brown, and fall off. Eventually, the trees become completely leafless. Other insects are often attracted to the diseased trees, which can further damage the plants.

There are different traditional mechanisms to diagnose Coffee diseases by chemical analysis or visual observation. However, Traditional techniques of detecting coffee disease are expensive, inconsistent, error-prone, time-consuming, require trained professionals and specialized tools, and are inefficient [7]. Automated detection systems for coffee diseases often employ transfer learning due to its several advantages over traditional machine learning approaches.

Data Scarcity: Collecting and labeling large datasets of coffee plant images, especially those with various disease symptoms, can be challenging and time-consuming. Transfer learning allows researchers to leverage pre-trained models, such as those developed on large image datasets like ImageNet, to extract relevant features from coffee plant images without the need for extensive training data.

Feature Extraction Efficiency: Transfer learning enables the system to efficiently extract meaningful features from coffee plant images, even with limited training data. This is because pre-trained models have already learned to identify general patterns and features that are common across different image domains.

Improved Generalizability: Transfer learning helps improve the generalizability of the model, meaning it can perform well even when encountering new coffee plant images with variations in lighting, background, and disease symptoms. This is crucial for real-world applications, where

the system should be able to detect diseases across different coffee varieties and growing conditions.

Reduced Training Time and Computational Resources: Transfer learning significantly reduces the training time and computational resources required compared to training a model from scratch.

Transfer learning offers a powerful approach for automated detection of coffee diseases, addressing the challenges of data scarcity, feature extraction, generalizability, and computational efficiency. By leveraging pre-trained models and fine-tuning for specific tasks, transfer learning enables the development of effective and practical disease detection systems for the Ethiopian coffee industry. Therefore, I am motivated to develop a system that easily detects diseases in coffee leaf and bean images and recommends farmers aiming to assist coffee producers to increase their productivity and quality of their commodity using image process and Transfer learning techniques to support experts and farmers.

1.2 Motivation of the Study

Automatic detection of plant diseases is now widely used in the agricultural sector all over the world. Because agricultural products account for 80 to 85% of the Ethiopian economy, this technology is still an insufficient, interesting area of research, and we would have been motivated to work with Transfer learning for image disease identification and recommend its treatment.

As a result, various plants affected by fungi, bacteria, or damaged by different diseases have been a popular research area in recent years. Many researchers internally and externally work in this area. But I haven't seen more research that satisfies me According to its value on the Coffee plant. Coffee diseases cause a significant issue and result in economic and agricultural industry loss. In Ethiopia, Coffee is produced by large-scale cultivated families, primarily for domestic utilization and nearby markets for both local consumption and export. The current endeavors made by the Ethiopian government and many non-governmental organizations to increase value in production chains of Coffee could expand the production of many valuable. Currently, technological improvements have many advantages for agricultural observations and implementations, The area of artificial intelligence is one of these technologies Besides that

image processing and Deep learning are emerging technologies that can be advantageous for Ethiopian agriculture, especially in plant disease detection.

1.3Statement Of Problem

The most valuable tropical crop in terms of export earnings is coffee, which is also an essential source of money for those countries where it is grown. Farmers have to grow and manage coffee production and quality to supply the needs of the global coffee industry. Coffee leaf disease is a destructive disease that can have significant impacts on coffee quality and production, resulting in lower yields and worse-quality beans [8]. For developing countries, agriculture is one of the most important economic sectors, both for foreign exchange earnings and for ensuring the food sustainability of their citizens [9].

Ethiopia ranks fifth among coffee-producing countries in the world. It contributed 29 – 31% of the export earnings of the nation, sustaining 4.7 million small-holder farmers directly, absorbing 25% employment opportunity, generating $780 million from export amount 184,000 tons in 2014/2015 and 10% of government revenue[10]. Despite the greatest economic impact; it is threatened by several diseases like Coffee Berry Disease (CBD), Coffee Wilt Disease (CWD) and Coffee Leaf Rust (CLR) are major once [10].

The diseases affecting coffee plants are a critical factor in decreasing the productivity and quality of coffee plants. Coffee is essential for the country's biological, social, and economic well-being. However, the disease has stopped the country from fully achieving the crop's genetic and ecological potential [11].

Kumlachew[11], reported 29.9% of national average crop losses to total harvestable coffee yield due to CBD, and the increased intensity was associated with disease management and altitude. Coffee wilt disease (CWD) caused significant economic losses in Ethiopia, with an estimated national incidence of 27.9% and severity of 3%, resulting in monetary losses of more than 3.7 million USD. The management of CWD is more difficult because, unlike coffee berry disease, coffee trees affected by the coffee wilt disease cannot be saved more. Coffee leaf rust was first observed in Ethiopia in 1934, although it has yet to reach pandemic proportions or result in the destruction of Arabica coffee in the nation [11]. Coffee leaf rust (CLR) is a widespread disease that has varied degrees of severity in Ethiopia's coffee-growing areas. The country's average number of infected coffee trees was estimated to be 12.9% in 1980 but had more than tripled to

36.3% by 1990. Kumlachew[11], reported in Hararghe, the severity of coffee leaf rust (CLR) has been observed to be as high as 27%. This might be because of the existence of vulnerable coffee varietals, aggressive fungal races, or the type of coffee manufacturing equipment used.

Transfer learning involves training a base network on a large dataset, such as ImageNet, and then transferring the learned features to a second network that is trained on a smaller dataset for a specific task. This can be useful when the data for the specific task is limited, or when it is computationally expensive to train a network from scratch. Fine-tuning a CNN from scratch takes time, is computationally expensive, and requires a large amount of data. For these reasons, researchers usually begin with a pre-trained CNN trained on a big dataset such as ImageNet. This gives a suitable starting point for the CNN's weights, which may subsequently be fine-tuned to a given job while using less time, data, and computational resources [12]. To fine-tune a CNN for a new classification task, we can first remove the final fully-connected (FC) layer and replace it with a new FC layer with the number of neurons equal to the number of classes to be classified. We then train only this new FC layer on our dataset, using the features extracted from the final layer of the pre-trained CNN.

In addition to replacing the classifier, we can also fine-tune the weights of the previous layers of a CNN. To do this, we "freeze" the weights so that they do not change, except for those in the layers that we are fine-tuning. We can potentially fine-tune the entire network, but this increases the risk of overfitting. Therefore, we need to find a balance between the amount of fine-tuning and the size of our dataset.

Currently, farmers and Coffee experts in Ethiopia look at their farmland with their eyes to detect and diagnose coffee leaf diseases. But this way can be time-consuming, expensive, and sometimes inconvenient. Detecting and diagnosing coffee plant diseases early and accurately is essential to prevent their spread. This task should be automated using technology, rather than relying on manual labor. Much agricultural research of countries is held up to disease detection but they can’t recommend what action they can take after their farmland suffers from disease. Ethiopia lost financially due to coffee disease according to researchers.

Haymanot [13], Abnet [14], Eyobed [15], and Abraham Debasu et al. [16] conducted studies on coffee disease detection. However, we were unable to use their studies for several reasons. One reason is that they did not cover all coffee diseases. Additionally, their datasets were small. there

is still no research that addresses all four diseases. Some studies have focused on CLR, while others have focused on CBD or CWD. However, no study has yet been conducted that combines all four diseases into a single model and recommends their treatment. Therefore, Coffee disease detection is still an active area of research, with the goal of improving accuracy and reliability.

1.4Research Question

This research seeks to answer the following research questions.

RQ1. To what extent does the CNN base transfer learning classifier classify the Coffee plant disease?

RQ2. Which Augmentation technique is more effective on model performance?

RQ3. Which hyperparameters (number of layers, learning rate of the optimizer, batch size, and number of epochs) are better for the designed model?

RQ4. Which transfer learning is best for image-based coffee leaf and bean disease detection?

1.5 Objective Of Study

1.5.1 General Objective of study

The general objective of this study is to develop an image-based coffee plant disease detection model using transfer learning**.**

1.5.2 Specific Objectives of study

The main specific objective of this research is as follows:

- To develop a model that can detect Coffee disease and recommend what action can producers take after they get the disease.
- To investigate which data augmentation technique is most effective for improving the performance of a model.
- To examine which set of hyperparameters is most effective for improving the performance of a model.
- To identify and select which transfer learning is most effective for classifying coffee leaf and bean diseases.

1.6 Scope And Limitation

Arabica coffee originated in the highland forests of Ethiopia, where it was discovered growing wild in the 15th century. The coffee trees grow in these high-altitude, cool, and moist conditions. Arabica coffee is known for its delicate flavor and aroma, and it is the world's most popular type of coffee. Therefore, this research is focused on identifying diseases in Arabica coffee, which is the most popular type of coffee grown in Ethiopia. This study focuses on the most common coffee leaf and coffee bean diseases in Ethiopia, including Cercospora leaf spot (brown eye spot), coffee leaf rust, phoma leaf spot disease, and coffee berry disease.

This study focuses on coffee leaf and bean diseases, excluding coffee root and stem diseases. Collecting images of coffee roots and stems was therefore outside the scope of this study. Both infected and healthy coffee leaf and bean images were used to train and develop the model. A CNN-based transfer learning architecture was deployed. This study develops a model to detect coffee leaf diseases like Brown Eye Spot (BES), Coffee Leaf Rust (CLR), Phoma Leaf Spot (PLS), and coffee bean disease of coffee berry disease (CBD).

The main limitation of this study is that most plant diseases exhibit signs and symptoms on the surface of the leaf or berry, which can be captured using image processing techniques. However, some diseases may manifest symptoms that are not visually detectable. For example, some diseases may cause internal damage to the plant that cannot be seen from the outside. Additionally, the severity of a disease can also affects its appearance, making it difficult to identify. Additionally, the data used in this study was collected during a time of year when the diseases of interest were not actively displaying symptoms. Despite these limitations, this study provides a valuable starting point for developing a more comprehensive image processing-based system for detecting coffee plant diseases. Future research should focus on developing techniques to identify diseases that cause internal damage and to collect data during times of year when the diseases are actively displaying symptoms.

1.7Purpose Of the Study

As the world is developing with technology, I believe that the farmers of our country should take advantage of this opportunity. Deep neural network using transfer learning is recent technologies that can be used to detect coffee diseases from coffee leaf and bean images, and they can help us

enhance our living conditions. I do not doubt that this work is very important for our country's agricultural professionals and farmers. If the agricultural sector of our country is developed with technology, the country can register rapid growth continuously so I was forced to do such work. The study developed a model that detected the disease automatically and recommend what action can producers take, so it has benefits in terms of producing quality of coffee.

The finding encourages other researchers to use image processing and deep learning to detect coffee disease. Expected results of this thesis is:

- **Automate coffee plant disease detection model:** The proposed model will replace the use of microscopes and lab-based methods for detecting coffee plant disease, making the process faster, more efficient, and accessible to a wider range of users.
- **Recommendation:** The model will suggest recommended treatment methods based on the signs and symptoms of the detected disease, helping domain users to make informed decisions about how to manage their plant.
- **Benefit government and private agricultural organizations**: The proposed model can be used by government and private agricultural organizations to develop targeted interventions and policies to improve coffee production and reduce crop losses.

1.8 Organization of the Thesis

This research work is divided into five chapters:

Chapter 1: Introduction This chapter provides a background on coffee plant disease detection and introduces the proposed methodology. It also presents an overview of the entire thesis.

Chapter 2: Literature Review This chapter reviews previous work on coffee plant disease detection, with a focus on deep learning techniques. It also identifies the gaps in the existing literature that the proposed research aims to address.

Chapter 3: Methodology This chapter describes the proposed methodology in detail, including the datasets used, the features extracted, and the deep learning model architecture.

Chapter 4: Results and Discussion This chapter presents the results of the proposed methodology and discusses the findings. It also compares the proposed model to existing state-of-the-art methods.

Chapter 5: Conclusion and Future Work This chapter summarizes the main contributions of the thesis and outlines directions for future research.

CHAPTER TWO

2 LITERATURE REVIEW

2.1 Introduction

This chapter reviews the literature on coffee plant disease, image processing, deep learning, and their applications in real-world scenarios. Image capture is the first step in digital image processing. After image capture, there are several steps that must be followed to achieve the desired goal of a machine vision system. This chapter also discusses the diseases that affect coffee leaves and beans, as well as their symptoms. The focus of this chapter is on image processing techniques for disease identification and a review of the relevant literature.

2.2 Coffee

There are over 125 species of coffee plants in the world, but only three are commercially important: Arabica, Robusta, and Liberian are the three primary varieties of coffee. Arabica is the most widely traded, accounting for over 60% of world commerce. Robusta is the second most popular, accounting for the majority of the remainder. Liberian is the least widely used currency, representing for less than 1% of world commerce [17]. Coffee is a globally consumed beverage that is cultivated in over 50 countries and consumed by more than one-third of the world's population [6], [18]. Every day an estimated 2.25 billion coffee cups are consumed throughout the world. With an estimated 319,500 metric tons of coffee produced in the year 2018-19, of which almost 80% is exported throughout the globe. The amount of revenue generated from the coffee industry is projected to be at USD 987m in the year 2020 and its market is projected to increase annually by 7.4%[19].

Coffee farmers in Africa and around the world face a constant threat from a variety of pests and diseases. Many of these are minor and have little impact on crop yield or quality. However, some pests and diseases, such as coffee berry disease, coffee leaf rust, and coffee wilt disease, can be very serious and have a devastating impact on individual farmers and the economies of countries and regions that rely heavily on coffee exports [5], [16], [19]. Coffee wilt disease is a fungal disease that has been present in Africa since the 1920s. In the 1990s, there were renewed and widespread outbreaks of the disease, which caused significant losses in Uganda and the

Democratic Republic of Congo. Once the disease is established on a farm, it is very difficult to control [5].

2.3 Ethiopian Coffee and Disease

Coffee originated in Ethiopia, specifically in the Kaffa region. All coffee grown in Ethiopia is of the Arabica variety. The majority of coffee is produced in the Oromia Region (63.7%) and the Southern Nations, Nationalities, and Peoples' Region (34.4%), with smaller amounts produced in the Gambela Region and around the city of Dire Dawa [20].

Ethiopia is the fifth largest coffee producer in the world. It contributes 29 – 31% of the export earnings of the nation, sustaining 4.7 million small-holder farmers directly, absorbing 25% employment opportunity, generating $780 million from export amount 184,000 tons in 2014/2015 and 10% of government revenue [10]. Despite the greatest economic impact; it is threatened by several diseases like Coffee Berry Disease (CBD), Coffee Wilt Disease (CWD) and Coffee Leaf Rust (CLR) are major once[10].

Ethiopia has two major coffee-producing systems. Sun coffee and forest coffee. Forest coffee is cultivated under the shadow of trees and is further classified into two types: Forest coffee is a traditional method that employs naturally growing stands of coffee trees in the forest. Farmers in this system play little role in the growth of the coffee trees. Semi-forest coffee is a more intensive system that includes management interventions such as tree cutting, understory clearing, and weed control. Farmers in this system grow coffee seedlings as well. Sun coffee is cultivated outside, without the protection of trees. This approach is typically small-scale, with coffee plants planted densely in a regular-sized area. Sun coffee is often grown at high elevations, ranging from 1700 to 2100 meters, and is frequently planted near homes. This is typical of coffee growing in Ethiopia's Harar coffee zone.

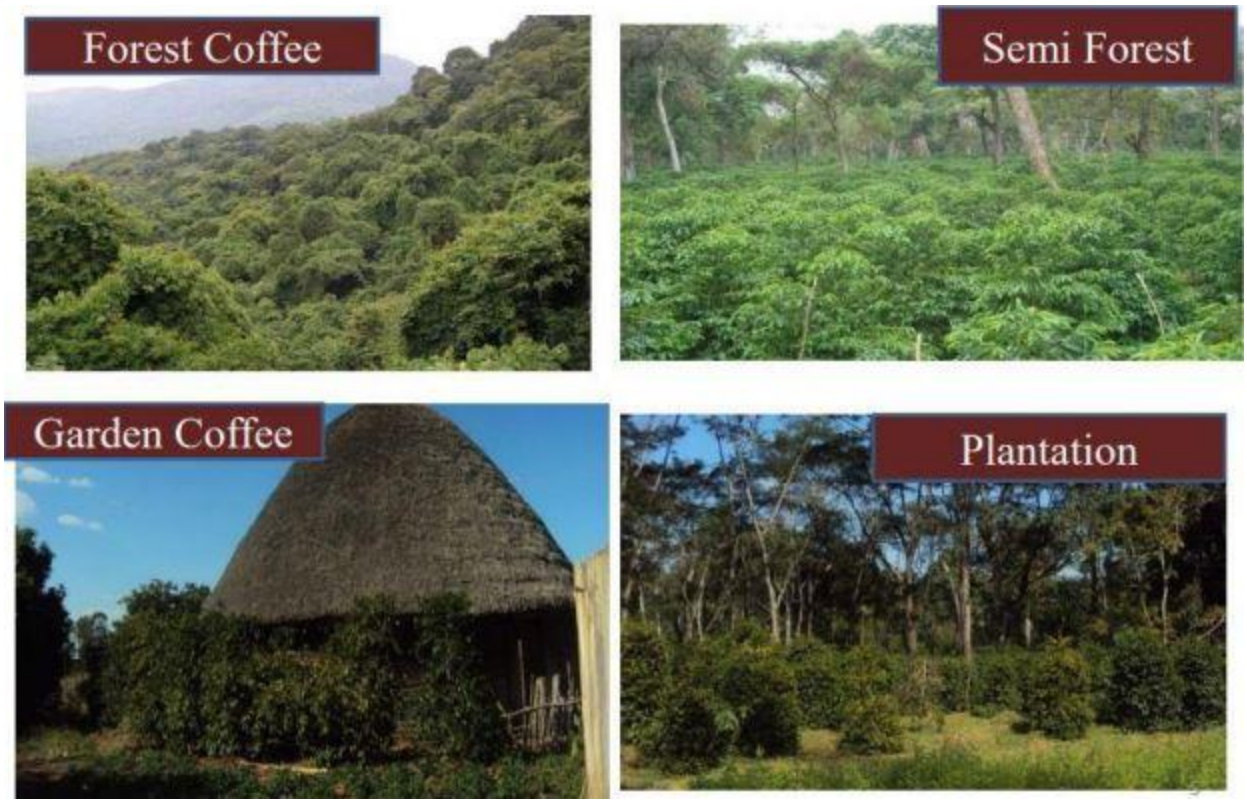


Figure 2.1 The coffee production system in Ethiopia[10]

Researchers studied the occurrence of diseases in four different rainforest regions in Ethiopia: Harenna in the Bale Mountains, and Bonga, Berhane-Kontir, and Yayu in the southwest.

2.3.1 Coffee Diseases

A plant disease is a condition that disrupts or modifies the normal life processes of a plant. Diseases can be caused by living organisms, such as fungi, bacteria, and viruses, or by non-living environmental conditions, such as soil compaction, wind, frost, and soil salt damage [21].

Coffee is prone to several diseases that reduce its yield & and quality. A plant disease is a condition that disrupts or changes the normal way that a plant works.

2.3.2 Coffee Leaf Rust (CLR)

Coffee leaf rust (CLR), a fungus caused by Hemileia vastatrix, may infect all three major coffee plant species: Arabica (Coffea arabica), Robusta (Coffea canephora), and Liberian (Coffea liberica). CLR is a disease that primarily affects coffee plants, causing chlorotic lesions or yellow

powdery pustules on the underside of the leaves, reducing the plant's photosynthetic area [5]. Coffee leaf rust (CLR) first appeared in Sri Lanka in the 1860s and has since spread to most coffee-growing regions of the world, becoming endemic in major producing countries. It is present in every coffee-growing country in Africa [10]. Coffee leaf rust was first detected in Ethiopia in 1934, although it never reached pandemic proportions, resulting in the elimination of Arabica coffee in the country [22].

Figure 2.2 Image of coffee leaf rust disease

Coffee leaf rust (CLR) is found primarily in low altitudes below 1700 meters above sea level, but has also been reported in high-altitude areas.

CLR Symptoms

- Orange patches on the lower surface of the leaves
- Pale yellow spots that appear first and later turn yellow/orange
- The entire leaf may become infected

2.3.3 Coffee Wilt Disease (CWD)

Coffee wilt disease (CWD) is a fungal disease caused by Fusarium xylarioides [6]. The first symptoms of coffee wilt disease (CWD) are yellowing, folding, and curling of the leaves. The

leaves then become limp, dry up, turn brown, and fall off. Eventually, the entire tree becomes leafless. CWD has been known to occur in Africa for over 70 years, having first been reported in the Central African Republic in 1928 [5]. Coffee wilt disease (CWD) was initially discovered in Ethiopia (Kaffa area) [22]. [4] reported that A vascular wilt disease caused by this pathogen is the primary cause of coffee tree death in Ethiopia.

Coffee wilt disease (CWD), caused by the fungus Gibberella xylarioides, affects an estimated 28% of coffee plants in Ethiopia [18]. Coffee wilt disease (CWD), caused by a soil-borne fungus, is difficult to treat with chemicals. Farmers may need to leave affected fields fallow for several years or plant other crops. CWD reduces coffee production (yield) at the farm level by 37% (from 1482 to 932 kg per sample farm), leading to a 67% decline in income (from 5038 to 1651 birr). In Ethiopia, CWD causes annual national crop losses of 3360 tonnes, valued at US$ 3,750,976 [17]. Coffee wilt disease (CWD) is the most difficult disease to handle in Ethiopian coffee production, producing large economic losses and making management difficult. It is a severe danger to coffee output and must be addressed immediately.

Figure 2.3 Image of coffee wilt disease

CWD Symptoms

- The first symptoms are seen as leaves curling inwards. The leaves may also wilt and feel dry to the touch. Yellowing may or may not occur.
- Sudden leaf fall may occur within a few days of the first symptoms.
- Primaries/bearing branches remain bare after the leaf falls. A few leaves may sometimes remain at the top of the main stems.

- Black or brown to violet streaks/bands are observed on the wood when the bark is peeled off the stem.
- Affected trees remain firmly rooted to the ground and do not topple on pushing.
- Affected trees never recover. Even if they produce suckers, these later dry up and also die.

2.3.4 Coffee Beery Disease (CBD)

Coffee berry disease was first reported in Kenya, close to the border with Uganda in 1922.[4], confirmed the occurrence of coffee berry disease for the first time in Ethiopia in 1971; The frequency of coffee berry disease was determined in several locations of Ethiopia at various times and in various coffee production methods by various experts. Recently[11], reported 29.9% of the national average crop losses to total harvestable coffee yield due to CBD, and the increased intensity was associated with disease management and altitude. A fungus causes coffee berry disease, which creates dark, sunken lesions on green berries that may drop from the plant.

Figure 2.4 Image of coffee berry disease

Coffee berry disease (CBD) is caused by a fungus that may infect coffee plants at any altitude and in every coffee-growing region. During the rainy season, the fungus spreads quickly on the tree's bark, and berries are especially vulnerable to infection. 4-20 weeks after growing [23].

CBD Symptoms

- Dark brown blotches on flowers
- Small, dark, sunken patches on green berries

- Dark, sunken patches with black dots on ripe berries
- Dark brown coloration on the maturing bark
- Infected berries may shed or remain on trees in black shriveled condition

2.3.5 Brown eye spot disease (BES)

Brown eye spot (BES), a fungal disease that flourishes in warm, humid environments, is found in all coffee-growing regions across the world. While it is the most frequent coffee arabica disease, it typically causes production losses in this species. It can, however, create defoliation and decrease the vitality of seedlings and young plants [14].

Figure 2.5 Image of Brown eye spot disease

BES Symptoms

- On the leaves, there are tiny brown specks that are more noticeable on the upper surface.
- The spots occur within the leaf, mostly between the veins, and also at the margins.
- The spots can grow up to 15 mm in diameter; they have light brown or sometimes light grey centers, surrounded by a wide, dark brown ring, and a yellow margin.

2.3.6 Phoma Leaf Spot (PLS)

Phoma leaf spot, a fungal disease caused by Phoma tarda, is one of the most serious coffee diseases in Ethiopia, causing severe crop quality and production losses. Symptoms include discoloration of tips and branches, dead rosettes, dried-up berries, and leaf blotches. Control options include planting in places less prone to cold winds, erecting windbreaks, and administering balanced fertilizer with nitrogen, calcium, and micronutrients. Chemical control may also be essential [24].

Figure 2.6 Image of phoma leaf spot disease

2.3.7 Healthy Coffee Leaf

Coffee is an important agricultural crop in the world economy, especially in Ethiopia, where it provides a significant source of money as well as cultural significance. Ethiopia is Africa's top producer and exporter of coffee, accounting for 22% of the country's commodity exports [15].

Figure 2.7 Image of healthy coffee

2.4 Common Coffee Disease in Ethiopia with their Causative Pathogen

The following table describes common coffee diseases in Ethiopia with their Causative pathogens according to Kifle [10].

Table 2.1 common coffee disease with their causative pathogen[10]

S/N	Disease	Causative Pathogen
1	Coffee Berry Disease	Colletotrichum kahawai
2	Coffee Wilt Disease	Gibberella xylarioides
3	Coffee Leaf rust	Hemilleia vastatrix
4	Brown eye spot	Cercospora coffeicola
5	Phoma leaf spot	Phoma tarda (Stewart)
6	Thread Blight	Corticium koleroga
7	Bacterial Blight of Coffee	Pseudomonas syringae
8	Brown blight	Colletotrichum gloeosprioids
9	Bean discolorations	Pseudomonas syringae
10	Leaf blight	Ascochyta tarda
11	Fruit rot	Fusarium sp.
12	Root rot	Armillaria mellea

13	Damping off	Pythium spp.
14	Coffee bean mold	Aspergillus and Penicillium
15	Coffee Bark disease	Fusarium stilbenoids

2.5 Artificial Intelligence (AI)

Artificial intelligence (AI) is the ability of machines to think and act like humans. It is used to create machines that can perform tasks such as speech recognition, learning, planning, and problem-solving. AI is essential for robotics, which is the field of creating intelligent machines that can interact with the world around them. AI addresses the important questions of what knowledge is needed to think, how to represent that knowledge, and how to use it. Robotics challenges Artificial Intelligence by forcing it to deal with real objects in the real world [25].

2.6 Machine learning

Machine learning, a branch of artificial intelligence, has grown in popularity in recent years. Several researchers are now exploring and applying machine learning methods to a variety of applications, including image and speech recognition. Machine learning is not a new area; it dates back to the mid-1950s through the mid-1960s [26]. Early machine learning research concentrated on the development of self-organizing and adaptive systems, which are unsupervised learning systems that can learn from data without the intervention of a person. In the mid-1970s, concept learning became the focus of machine learning research. For the next decade, the focus of machine learning research was concept learning. However, the discipline developed dramatically in the 1980s, moving from learning single ideas to learning numerous concepts and from experimenting with various learning methods to generating new learning methods. This improvement in both breadth and depth signaled the start of a new period of rapid development in machine learning. The revival of neural networks was a significant driving force in this breakthrough, resulting in the creation of new machine-learning methods and systems. This move from theoretical research to practical applications has continued to the current day, and machine learning is now one of computer science's most significant and fast-increasing topics [27]. Machine learning is divided into several categories.

2.6.1 Supervised Learning

Supervised learning, a classic machine learning algorithm, includes logistic regression and reverse neural networks. The input data in supervised learning is called the training dataset. The basis function models can be algebraic or probability functions. The model is trained using an iterative process, and the training data results are known to the functions. Supervised learning algorithms work by first making predictions based on a training dataset. The algorithm then iteratively updates its predictions until they match the known outcomes in the training dataset. This makes supervised learning algorithms well-suited for classification and regression tasks. From those machine learning types, my research is also supervised learning[27], [28].

2.6.2 Unsupervised Learning

Unsupervised learning is a type of machine learning where the algorithm is not given any labeled training data. Instead, the algorithm must learn from the data on its own. This means that the results produced by unsupervised learning algorithms are uncertain, as the algorithm is not sure what it is looking for. Unsupervised learning can be divided into three main categories: Unsupervised learning methods, such as clustering, anomaly detection, and competitive learning, can be used to identify patterns and structures in data without the requirement for labeled data. Clustering methods, for example, may be used to organize data points into clusters based on their similarity. Typical unsupervised learning methods include the K-means algorithm, the Apriori algorithm, and the SOM algorithm [27], [28], [29].

2.6.3 Semi-supervised Learning

Semi-supervised learning is a practical sort of machine learning that can train models with less labeled data than other machine learning methods, making it more efficient. It learns the associations between distinct characteristics by combining labeled data (data with known results) with unlabeled data (data without known outcomes) and generating a classification or regression model. Semi-supervised learning may be applied to classification and regression problems [30].

2.7 Deep Learning

DL refers to “deeper” neural networks that use various convolutions to produce a hierarchical representation of data. Deep learning is a relatively new, cutting-edge technology for image processing and data analysis that has a lot of promise and potential. Deep learning has been successfully utilized in a variety of sectors, and it has recently entered the agricultural arena[31].

Deep learning outperforms frequently used image processing approaches in terms of accuracy. DL is made up of several different components (e.g., convolutions, pooling layers, fully connected layers, gates, memory cells, activation functions, encode/decode schemes, etc.), depending on the network architecture used (i.e., Unsupervised Pre-trained Networks, Convolutional Neural Networks, Recurrent Neural Networks, Recursive Neural Networks)[32].

Deep learning is the popular area of machine learning which efforts to learn high-level abstractions in data by utilizing hierarchical architectures[33]. Hidden layers are composed of a set of neurons responsible for training the unique set of features based on the output layer of the previous layer. Data complexity and abstraction rise in direct proportion to the number of hidden layers.

As we see from the figure any Deep neural network will consist of three types of layers:

- **The Input Layer:** It receives all the inputs and the last layer is the output layer which provides the desired output
- **The Hidden Layer:** Hidden layers are the layers in a neural network that are between the input layer and the output layer.
- **The Output Layer:** which gives the result

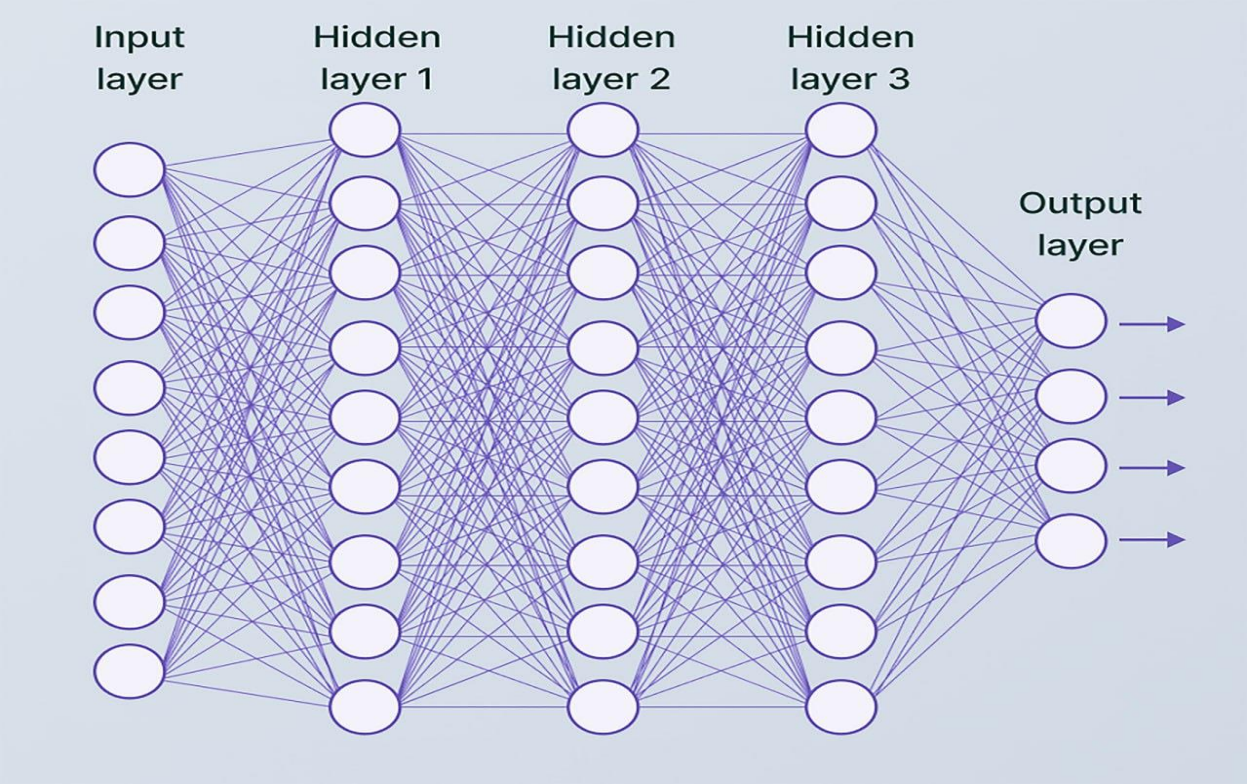


Figure 2.8 Deep learning neural network[33]**.**

2.8 Neural Network

A neural network is made up of layers of neutrons. These neutrons were motivated by research into biological nerve systems. In other words, neural networks are an attempt at creating machines that work in a similar way to the human brain by building these machines using components that behave like biological neurons. The function of that neural network is to produce an output result when presented with input data[34].

Neural networks are trained by feeding them a set of data and telling them what the correct output should be for each input. The network then adjusts the weights of its connections until it can produce the correct output for each input. Once trained, a neural network can be used to make predictions or classifications on new data.

2.8.1 Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) are a type of machine learning algorithm that is specifically designed to process pixel data. They are widely used in image recognition, image

clustering, classification, and object detection. CNNs have been significantly improved since their introduction in 1989, with the main advances coming from the rearrangement of processing units and the design of new blocks. Most CNN architectural advancements have focused on depth and spatial exploitation. CNNs can be divided into seven categories based on the type of architectural changes used: spatial exploitation, depth, multi-path, width, feature-map exploitation, channel boosting, and attention-based CNNs [35].

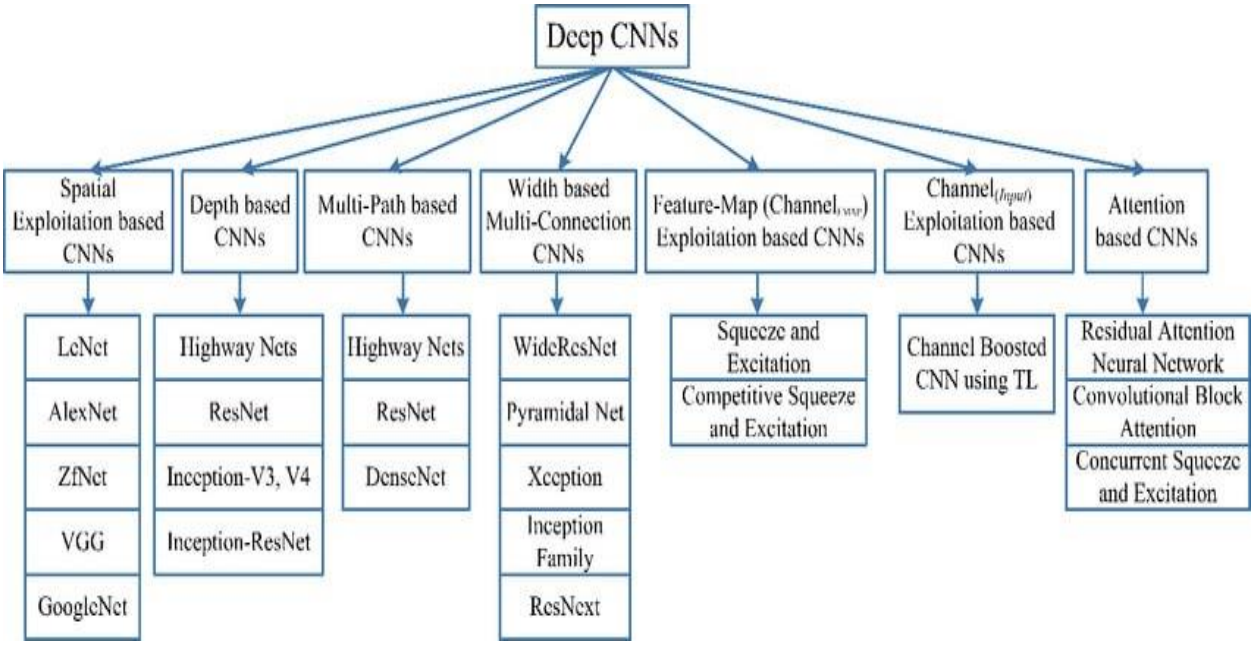


Figure 2.9 Taxonomy of CNN architectures[35]**.**

CNNs (Convolutional Neural Networks) are a type of deep learning model that has become increasingly popular for image recognition tasks. This is due to three key advantages: first, Automatic feature extraction: CNNs can learn features directly from raw image data, without the need for manual feature engineering. This makes them more efficient and effective than other types of deep learning models. Secondly, State-of-the-art accuracy: CNNs have achieved state-of-the-art results on a wide range of image recognition tasks, such as facial recognition, object detection, and image classification. Third, Transfer learning: CNNs can be retrained for new recognition tasks by using pre-trained weights from existing networks. This enables researchers to build on the work of others and develop new models more quickly and easily [36]. Depending on the application, researchers can create a CNN from scratch or transfer learning by using a pre-trained CNN model with an existing dataset. CNNs are deep learning models that are very

dependent on the size and quality of the training data. However, when trained on a large and well-prepared dataset, CNNs can surpass human performance on image recognition tasks. CNNs require relatively less preprocessing than other image-processing algorithms. This is because CNNs can learn features directly from raw image data. A CNN connection pattern mimics the organization of the human visual brain. A CNN consists of multiple layers, including input layers, output layers, and hidden layers. There is no limit to the number of hidden layers that can be used in a CNN [37].

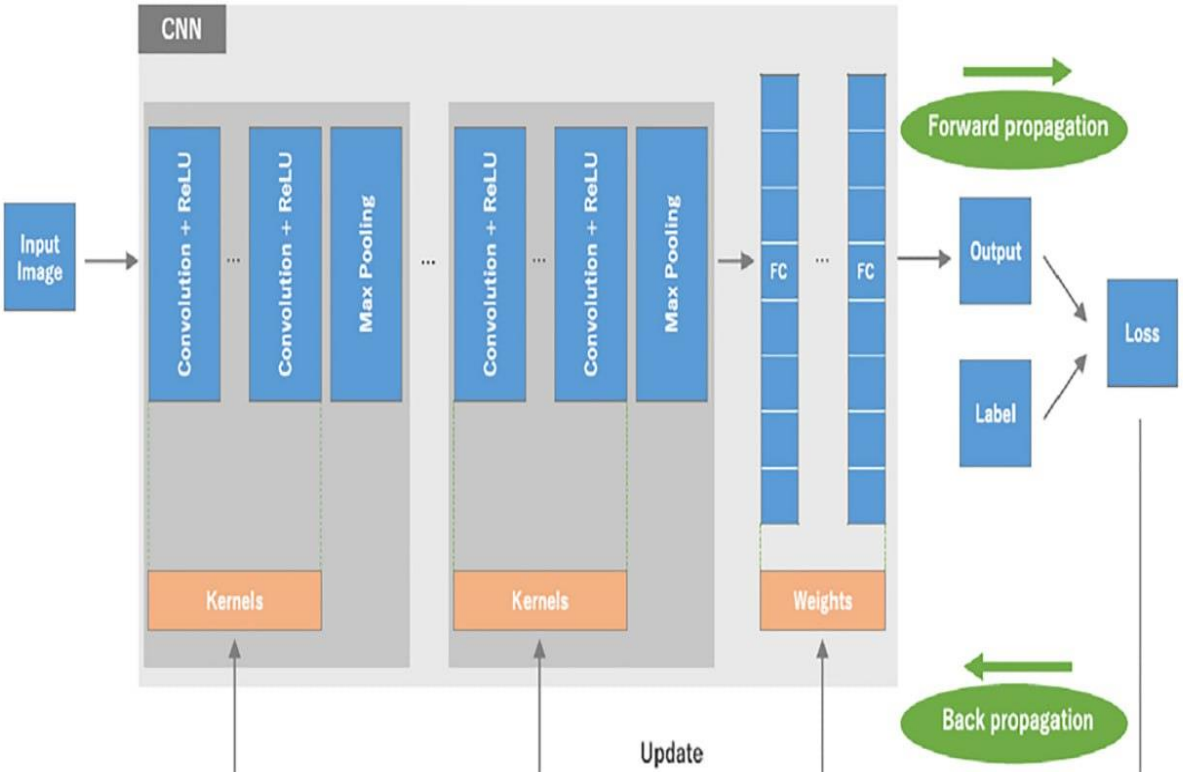


Figure 2.10 An overview of CNN architecture and the training process[37].

Each layer in a CNN model is arranged in three dimensions based on the input image: height, width, and depth, or a × a × b, where height (a) equals width and depth (b). The depth is also known as the channel number. For example, in an RGB image, the depth (b) is equal to three. Several kernels (filters) accessible in each convolutional layer are designated by k and, like the input image, have three dimensions (c × c × d). However, c must be less than a, and d must be equal to or less than b. Furthermore, the kernels serve as the basis for the local connections,

which use identical parameters (bias **bk** and weight **Wk**.) to generate k feature maps **hk** with a size of (a – c - 1) each and are convolved with input, as previously described. The convolution layer, like NLP, produces a dot product between its input and the weights, but the inputs are undersized portions of the initial picture size [38]. Next, we extract the following equation by adding nonlinearity or an activation function to the convolution-layer output:

$$\mathbf{h^k = f (w^k * x + b^k)}$$

The next step is to down sample each feature map in the sub-sampling layers. This results in a decrease in network parameters, which speeds up the training process and allows for the treatment of the overfitting problem. The pooling function (e.g., max or average) is applied to a neighboring region of size l x l, where l is the kernel size, for all feature maps. Finally, like in a conventional neural network, the fully connected (FC) layers receive the mid- and low-level characteristics and generate the high-level abstraction. The categorization scores are produced by the last layer [for example, support vector machines (SVMs) or SoftMax] [37].

Advantages of using CNNs

The following are the primary benefits of employing CNNs over other neural networks in image processing:

- **Use weight sharing.** This means that the same parameters are used to learn different features, which reduces the number of trainable parameters and helps the network avoid overfitting.
- **Jointly learn feature extraction and classification.** This means that the network learns both how to extract features from images and how to classify those features at the same time. This results in a model that is both accurate and efficient.
- **Are scalable.** CNNs can be easily scaled to large datasets and complex tasks by adding more layers and neurons.

CNN layers

The CNN architecture consists of several layers (or so-called multi-building blocks). Each layer in the CNN architecture, including its function, is described in detail in the figure below.

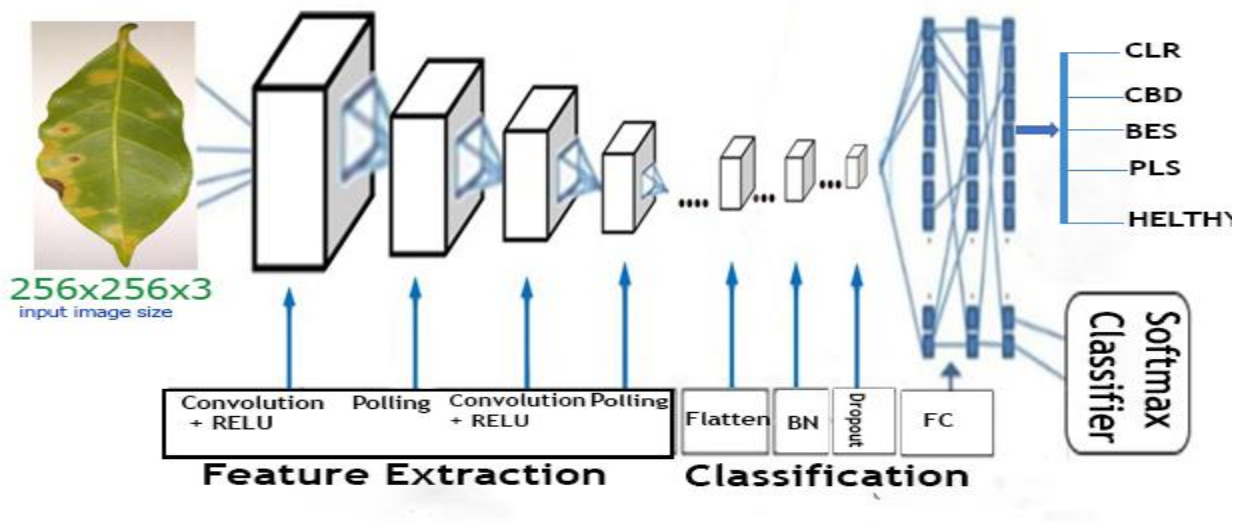


Figure 2.11 CNN layers

Convolutional layer: The most important component of a CNN architecture is the convolutional layer. It is made up of a number of convolutional filters (kernels) that are applied to the input image to generate the output feature map. The input image is represented as an N-dimensional tensor, where N is the number of channels (e.g., 3 for RGB images) [39].

Kernel definition: A kernel is a small grid of numbers that is used to extract features from an image. The values in the kernel are called kernel weights. At the beginning of training, the kernel weights are initialized with random values. Then, the weights are adjusted at each training iteration until the kernel learns to extract significant features from the images. For example, a kernel might be designed to identify horizontal edges, vertical edges, or corners. By applying the kernel to the image and looking at the output, CNN can learn to identify these patterns in other images.

Stride: It is a parameter of the neural network that adjusts the amount of movement over the image. Stride specifies how many pixels do; move the convolutional filter horizontally and vertically at each step. It controls how the filter convolves around the volume of input. Usually, the stride is equal to one (uncommonly two, and even more two). When the stride is 1 then one pixel at a time, it passes the filters. When the stride is 2 (or unusually 3 or more, but in practice

this is rare) then as it slides them around the filters leap 2 pixels at a time this will create spatially smaller production volumes.

Convolutional Operation: Initially, the CNN input format is given. The vector format is the standard neural network's input, whereas the multichannel image is the CNN's input. The grayscale image format, for instance, is single-channel, but the RGB image format is three-channel. To better understand the convolutional process, consider a 7 x 7 gray-scale image with 3 x 3 random weight-initialized kernels. First, the kernel moves horizontally and vertically over the whole image. In addition, the dot product of the image being used and the kernel are computed, in which their respective values are multiplied and then summed to provide a single scalar value calculated concurrently. The operation is then repeated until there is no more sliding possible. It should be noted that the generated dot product values indicate the output's feature map. The diagram below depicts the major calculations performed at each stage. The light red hue in this picture indicates the 3 x 3 kernels, whereas the light blue color represents a similar-sized portion of the input image. Both are multiplied, and the total of the resultant product values (shown in light green) indicates an entry value in the output feature map.

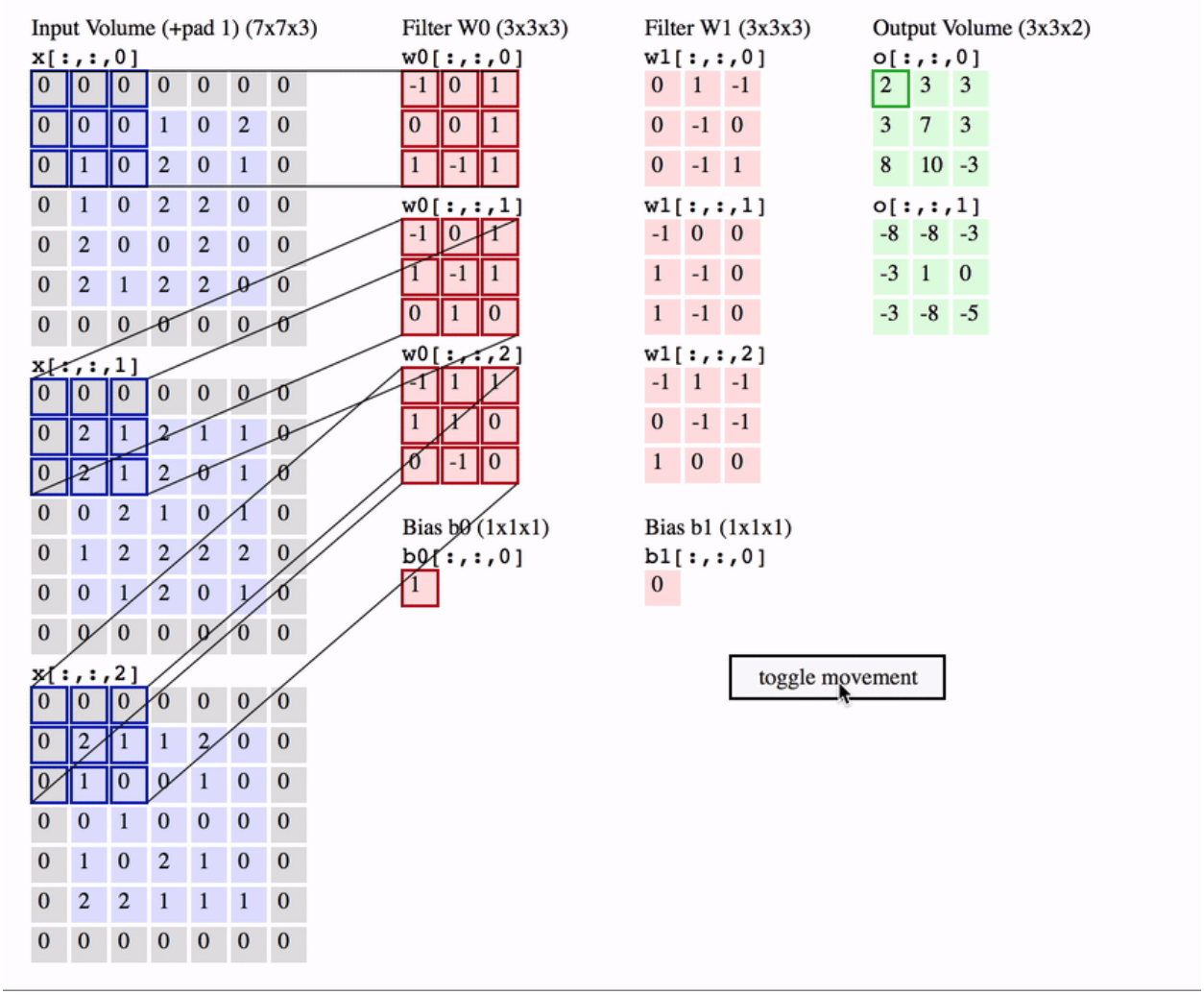


Figure 2.12 How the convolution layer works in CNN[39]

Pooling Layer: Pooling layers are used to down sample feature maps produced by convolutional layers. This means that pooling layers shrink the size of the feature maps while preserving the most important information. Pooling layers are similar to convolutional layers in that they have a stride and a kernel size. However, pooling layers do not have any weights. There are many different types of pooling layers, but the most common are max pooling, min pooling, and average pooling. Max pooling takes the maximum value of the pixels in a pooling region, min pooling takes the minimum value, and average pooling takes the average value [40]. The pooling layer may sometimes decrease a CNN's overall performance because it simply identifies the position of features in the input image rather than their presence or absence. This may lead the CNN model to miss important information.

Activation Function (non-linearity): Activation functions, which map input to output, are at the heart of all neural networks. The weighted sum of the neuron input and its bias (if present) is used to compute the input value. This implies that the activation function determines whether or not to fire a neuron in response to a certain input by generating the associated output.

Non-linear activation layers are used after all weighted layers (learnable layers such as fully connected layers and convolutional layers) in convolutional neural networks (CNNs). This causes the mapping from input to output to become non-linear, allowing CNNs to learn complicated things. Differentiable activation functions are also required for error backpropagation, the technique used to train CNNs.

Sigmoid: This activation function takes real numbers as input and outputs only values between zero and one. The sigmoid function curve is S-shaped and is described by Equation.

$$\mathbf{F(x)_{sigm}} = \frac{1}{1 + e^{-x}}$$

Tanh: It is similar to the sigmoid function in that it accepts real numbers as input, but its output is limited to values between -1 and 1. It is mathematically represented as follows.

$$\mathbf{F(x)_{tenh}} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ReLU: In the CNN context, this is the most frequently used Activation function. It changes the input's complete values to positive integers. The key advantage of ReLU over the others is its lower computational load. It is mathematically represented as follows.

$$\mathbf{F(x)_{ReLU} = max\ (0,\ x)}$$

Fully Connected Layer: A CNN architecture fully connected (FC) layer is often positioned at the end. Each neuron in this layer is connected to all neurons in the layer before it. This layer functions as the CNN classifier. As a feedforward ANN, it uses the same fundamental approach as a typical multilayer perceptron neural network. The FC layer receives input from the previous pooling or convolutional layer in the form of a vector produced by flattening the feature maps. The final CNN output is represented by the FC layer output.

2.9 Transfer Learning in Deep Learning

In recent years, researchers have created a large number of CNN models from scratch and transfer learning. ImageNet is a research effort that developed a massive collection of annotated and labeled images. ImageNet is already used to train pre-trained models such as InceptionV1, InceptionV2, VGG-16, VGG-19, ResNet, Dense Net, Mobile Net, and Google Net. In deep learning, CNN is mostly used in image processing. Transfer learning is mostly applied in deep and machine learning for the implementation of different strategies. In deep learning, the transfer learning technique can be defined as the process of reusing a model that has already been trained for a specific task to perform a similar or related task[41]. In transfer learning a pre-trained model which is created by someone else to solve a similar problem.

2.9.1 When to use transfer learning

Transfer learning of deep neural networks is mostly used in the case of:

- **Lack of data:** A specific target task can be solved using a pre-trained model for a similar source task with a limited data set
- **Speed:** decreases training time and allows for building various solutions.

2.9.2 Transfer Learning Approaches

Transfer learning approaches applied in deep CNN in the case of:

- **Direct Use of Pre-Trained Models:** The simplest strategy is to solve a target task by directly applying a model from a source task.
- **Leveraging Feature Extraction from Pre-Trained Models:** In this approach discarding the last fully connected output layer and then making the pretrained neural network for feature extractor. In this transfer learning it is used to apply new datasets to solve different problems.
- **Fine-Tuning Last Layers of Pre-Trained Models:** This is applied by training the output classifier and then fine-tuning weights of the last block layers of the pre-trained model. The earlier layers of the network (CNN) are made frozen, whereas the last ones are freed up for tuning. This method helps to perform full training on the pre-trained model and modify the parameters at the last layers. Modification is applied at the last

layers because the earlier layers in a network capture more general features while later ones are very dataset-specific. We use this approach for this thesis.

2.10 Loss Function

In this study, the categorical cross-entropy loss function is used for multi-class classification, which involves using a SoftMax classifier to calculate the probability of each target label. The label with the highest probability is then predicted as the output class for a given input image [42].

```
model.compile(optimizer="Adam", loss="categorical_crossentropy", metrics=["accuracy"])
```

2.11 Optimizers

When training a deep learning neural network, Optimizer algorithms are used to modify each neuron's weight parameters to achieve low values of loss. The algorithms Adam, Adamax, AdaGrad, Adadelta, RMSprop, SGD, etc. are only a few examples.

2.11.1 Adam

This Optimizer combines two current popular methods: RMSprop, which performs well in online and non-stationary situations, and AdaGrad, which performs well with sparse gradients. Evidence from empirical studies demonstrates that Adam outperforms other stochastic optimization methods in practical applications. The fact that this Optimizer is computationally effective, memory-light, and invariant to the diagonal rescaling of the gradients makes it a good choice for problems involving large amounts of data and/or parameters.

2.11.2 AdaGrad

The most obvious enhancement to SGD is AdaGrad. Based on the historical gradient from some prior iteration, AdaGrad dynamically modifies the learning rate. Comparing AdaGrad to SGD, one key advantage is that it does away with the need to manually adjust the learning rate.

2.11.3 RMSprop

The idea behind RMSprop is to simply focus on the gradients in a window over a period rather than aggregating all historical gradients, and then calculate the second-order cumulative

momentum using the exponential moving average. This Optimizer was created to address AdaGrad's dramatically declining learning rates.

2.12 Learning Rate

The learning rate is an important factor in the training process. The models determine the rate of learning. It regulates the amount of proportional error that occurs after each update of the model's weights [43]. This happened after each batch of training samples. If the learning rate is high due to more significant weight alterations, or steps, the loss on the training dataset varies throughout training epochs [43]. When the learning rate is extremely low, the training tempo slows and may eventually come to a halt due to significant training errors [44].

2.13 Batch Normalization

To reduce internal covariate shifts and greatly speed up deep learning network training, batch normalization is advised. The means and variances of the layer inputs are fixed throughout the normalization process to achieve this. Additionally, batch normalization regularizes the model and reduces the need for Dropout [45].

2.14 Batch Size

The amount of training samples used in a single iteration is referred to as the batch size. The gradients will be calculated for each batch, and the network's weights will be updated as necessary. The training typically continues until the validation performance degrades.

2.15 Dropout

To avoid the problem of overfitting during training, a specific type of convolutional neural network layer called dropout is used in deep learning. It is performed by dynamically deleting random connections between nodes in particular levels of a neural network. Dropout, also known as their dropout from the network, is the temporary removal of some hidden and visible neural network units[46].

The straightforward idea of dropout is used to prevent overfitting. Every time you perform the exercise during the training period, a neuron with probability p is temporarily "dropped" or "disabled"[46]. It is measured that the dropout rate, or hyperparameter p, typically ranges between 0.5 and 50% of the neurons.

2.16 Popular CNN models

In recent years, researchers have created a large number of CNN models from scratch via transfer learning. A CNN's design is crucial for enhancing its performance for various applications. Since 1989, CNN designs have undergone several changes, including structural reformulation, regularization, parameter optimization, and so forth. However, rearranging processing units and inventing new blocks has resulted in the most substantial boost in CNN performance. The most creative breakthroughs in CNN designs have been in the field of network depth in particular. Alex Net model[47], ResNet model[48], Google Net model[49], VGG Net model[50].

This thesis uses a convolutional neural network (CNN)-based transfer learning model to detect and classify coffee diseases from images of coffee plants by inputting images of both healthy and infected plants. CNNs are composed of a series of sequential layers, each of which transforms one volume of activation to another using different functions. CNNs are used for classification because they can learn to extract features from images that are relevant to the task at hand.

2.17 Related Works

There has been a significant amount of work done so far to solve the coffee disease problem by local and foreign researchers. However, each work is designed to address different coffee plant diseases with different algorithms.

Abraham Debasu et al.[16], have developed a system to identify coffee leaf rust, coffee wilt, and coffee berry diseases from the leaves of the coffee plant. The system uses a digital camera and fluorescent light to capture an image of the leaf, and then uses machine learning algorithms to classify the disease. The researchers used a variety of machine learning algorithms for classification, including artificial neural networks, k-nearest neighbors, radial basis functions, Naive Bayes, and a hybrid of self-organizing maps. They also used a variety of image segmentation techniques, including Otsu, fuzzy c-means, k-means, Gaussian distribution, and combinations of k-means and Gaussian distribution. The system achieved an accuracy of 90% in experimental testing. However, the researchers found that the system's accuracy was lower for disease features extracted from the berry of the plant. They recommend further research and improvements to improve the system's ability to identify coffee berry diseases. One of the

drawbacks of the research is they consider only three types of disease but I want to consider five types of disease including Brown Eye Spot (BES), Phoma leaf spot (PlS).

Haymanot[13], have developed a convolutional neural network (CNN) model to detect coffee diseases such as Cercospora leaf spot, coffee phoma disease, and coffee berry disease. The model was trained on a dataset of 5886 images of coffee leaves, and achieved a training accuracy of 96.1% and a test accuracy of 69.1%. The researchers plan to improve the model's performance by training it on a larger dataset and using other pre-trained deep learning models such as ResNet. One drawback of her work is that she only considered three types of disease. I want to consider five types of disease, including Coffee Wilt Disease (CWD) and coffee leaf rust (CLR). I plan to use her recommendation as input for further investigation.

Abdulaziz[51], compares three CNN architectures (ResNet18, ResNet34, and ResNet50) for automatic plant disease identification with limited data. The study uses two datasets for training and testing: the Plant Village dataset and the coffee leaf dataset. The Plant Village dataset contains 54,305 leaf images of 14 crop species and 26 diseases, distributed among 38 crop-disease pairs. The coffee leaf dataset contains images of coffee leaves with three common diseases: Cercospora leaf spot, coffee phoma disease, and coffee berry disease. This dataset contains clear images of individual plant leaves and each image contains only one leaf. The study develops four approaches Transfer Learning (Baseline and Baseline++), Metric Learning Using a Triplet Network, and Deep Adversarial Metric Learning. The study achieves a very high accuracy of 99% for new classes when the source and target domain data are captured under the same condition. The study also achieves a reasonable accuracy of 81% for novel datasets that are captured under different conditions.

Abnet[14], The author developed a system to automatically identify diseases and pests from the leaves and berries of coffee plants using image processing and machine learning. The system uses two color spaces (L*a*b and YCbCr) and a texture filter segmentation algorithm to identify and segment the damaged regions of the plant. The system then uses a feedforward artificial neural network (ANN) to classify the diseases and pests. The author also compared the performance of the ANN classifier to the results obtained from support vector machine (SVM) and k-nearest neighbor (KNN) classification algorithms. The ANN classifier achieved the

highest accuracy, followed by the SVM and KNN classifiers. Finally, ANN achieved the best classification performance with an accuracy of 91.9%.

Eyobed[15], designed a coffee leaf disease detection using Deep learning. In this study model is designed based on CNN from scratch and CNN transfer learning (ResNet50 and mobile Net) to perform coffee disease detection. They collected 1120 images from the Wolaita Sodo agricultural research center Augmented their data and used a total of 3360 images for training and testing. They consider only three diseases brown eye spot, coffee wilt disease, and coffee leaf rust. They achieved a good accuracy of around 99.89% by using CNN transfer learning using ResNet 50, 97.01% using Mobile Net, and 98.55 using CNN from scratch.

Table 2.2 summary of related work

Reference	**Contribution**	**Limitation**
[14]	He applied feedforward artificial neural network (ANN), SVM, and K-nearest neighbor classification algorithms to classify coffee plant diseases and pests. ANN achieved better accuracy at 91.9 %.	He used a few data sets of 2,400 images and got that accuracy and I want to increase the dataset and modify his accuracy. He only considered three types of disease. But I want to consider more.
[13]	Imaging and machine learning techniques (CNN) She Achieved 96.1 % training accuracy and 69.1% testing accuracy.	The study focuses on only three major types of coffee disease (PIS, CBD, and BES). She doesn't consider the major diseases in coffee like CLR and CWD She developed a model with a low testing accuracy of 69.1% which I want to modify.
[16]	He used Imaging and machine learning techniques. ANN, KNN, Naïve, and a hybrid of Self-organizing Map SOM and Radial basis function (RBF). He Achieved 90 % using SOM and	The study focuses on only three major types of coffee disease (CLR, CBD, and CWD) The study doesn't mention the testing class from which class tested the model.

	RBF with texture color features	This means there are no healthy classes.
[15]	He compares CNN from scratch and CNN-based transfer learning He achieved a high 99.85% accuracy using CNN-based transfer learning	He used a small amount of dataset 1,120. He augments data and gets 3360. The study focuses on only three major types of coffee disease (CLR, CWD, and BES
[51]	Employed four approaches: transfer learning, triplet networks, & and deep adversarial metric learning achieved a high accuracy of 99%	They only compare only ResNet18, ResNet34, and ResNet50 Architecture.

2.18 Summary

Automatic coffee disease detection is a promising approach for identifying diseases in coffee plants. Machine learning and deep learning algorithms, such as artificial neural networks (ANNs) and convolutional neural networks (CNNs), have been used with great success. However, Due to the significant economic impact of coffee plant diseases, there is a pressing need to develop more accurate and efficient models that can be used to detect diseases in all parts of the coffee plant. Most previous studies on coffee disease detection have used small datasets from internet searches or publicly available databases. They have also considered a fragmented number of diseases, with one researcher using two types of diseases, another using three, and another using four. Additionally, most studies have focused on coffee leaves, rather than coffee beans. Finally, many studies have not properly categorized the diseases according to the affected coffee plant part.

This thesis presents a novel deep learning approach to detecting coffee diseases in both coffee leaves and coffee beans. The model is trained to identify Brown eye spot (BES), Coffee leaf rust (CLR), Phoma leaf spot (PLS), and Coffee berry disease (CBD). The model is developed using transfer learning, which is a technique that involves reusing a pre-trained model to solve a new problem. The proposed model is evaluated on a large dataset of coffee leaves and coffee beans

that is carefully categorized by coffee plant experts according to the affected coffee plant part. The model achieves high accuracy on all four diseases.

The proposed model has the potential to be used to develop a web service that allows coffee farmers to upload images of their coffee leaves and coffee beans and receive a diagnosis. This could help coffee farmers to identify and treat diseases early, which could lead to increased yields and profits.

CHAPTER THREE

3 RESEARCH METHODOLOGIES

3.1 Introduction

In this chapter, the experimental methodology that is used to complete this thesis work is defined clearly. Therefore, to develop the proposed model this study has passed through the following activities including data collection and preparation, a tool used to implement and test the model, and the evaluation techniques to measure the predictive performance of the model.

3.2 System Architecture

To design the Coffee leaf and bean disease detection model, we use different drawing tools, digital image processing, and deep learning techniques (CNN). The proposed Architecture is depicted in Figure 3.1. The first step is image data collection for both training and testing. second, the whole dataset is preprocessed means the dataset is resized to and normalized to *1/255 to convert decimal values of images between 0 and 1. Third, to increase the size of the dataset, the data augmentation techniques are applied by rotating the images into different degrees, horizontal flipping, vertical and horizontal shifting, resizing, cropping, padding, etc. It helps to address issues like overfitting and data scarcity, and it makes the model robust with better performance. fourth, Feature extraction from images is very effective at finding patterns that are difficult to detect with traditional methods. Finally, we train our CNN base transfer learning classifier and, the model classifies the image as healthy, CBD, CLR, BES, PlS, and Healthy. The overall model architecture is depicted in Figure 3.1.

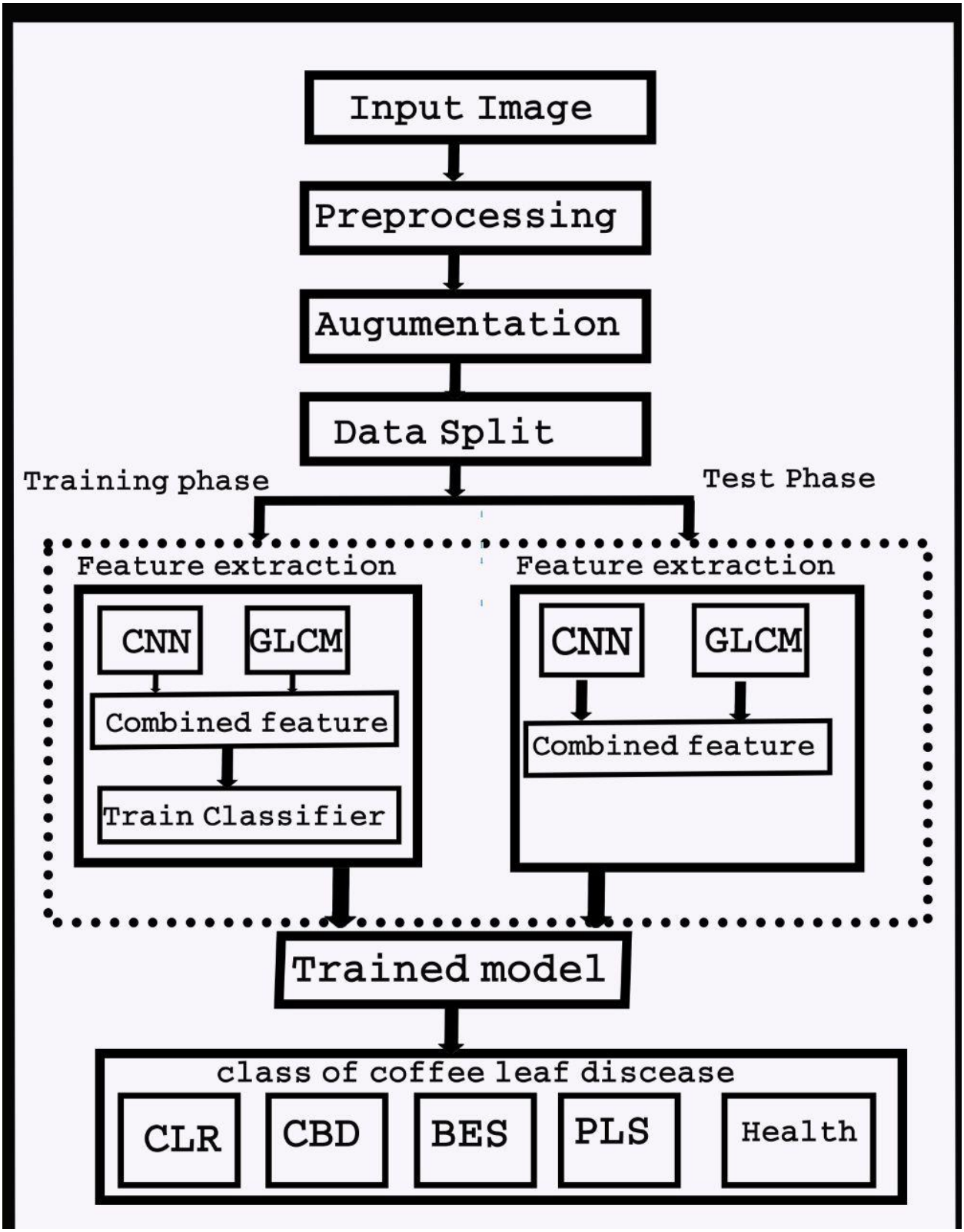


Figure 3.1 Proposed System Architecture

3.3 Dataset collection and preparation

The dataset for this thesis was collected from the Wondoganat Agricultural Research Center, Awada branch, located in Yirgalem, Ethiopia. The dataset contains images of coffee leaves and beans with labeled disease labels. The images were collected at different stages of disease progression and represent a variety of coffee plant diseases. After collection, the images were reviewed and verified by plant science experts at the Wondoganat Agricultural Research Center. The final dataset contains five classes:

- Coffee leaf rust
- Coffee berry disease
- Brown eye spot
- Phoma leaf spot
- Healthy coffee plants

The dataset is balanced, meaning that there is an approximately equal number of images for each disease class. This is important to ensure that the deep learning model trained on the dataset is able to learn the characteristics of each disease class effectively. The dataset is also representative of the different types of coffee diseases and the different stages of disease progression. This is important to ensure that the deep learning model trained on the dataset is able to generalize to real-world coffee disease detection scenarios.

3.4 Data pre-processing

Image preprocessing is a step we take to convert an image from its rough form into a form that our model is ready to use for training and testing. It can also be defined as a method of performing some important operations on an image to get an enhanced and high-quality image to extract the most useful information from it. The preprocessing phase includes image resizing to (224x224, 256x256, 299x299), background noise removal, and conversion of images into arrays using NumPy and Kera's. Image Pre-processing is an important step in many Machine learning applications. Because preprocessing reduces distortion and removes noise from the input images,

the quality of the final image is improved. This study used preprocessing techniques to convert RGB images to grayscale, downsize images, remove noise, and improve image quality. The image preprocessing technique is a method of processing images so that the outcome is significantly better than the original picture for a particular use. Before being fed into a neural network or deep learning system, image data is processed. Pre-processing includes tasks like cropping, scaling, and color conversion.

OpenCV (Open-source Computer Vision) is used to resize images and convert them to NumPy arrays. The RGB images taken by the digital camera are the originals. These RGB images are made up of three main hues (Red, Green, and Blue). As a result, because the color range is between 0 and 255, it is difficult to develop the system application. As a result, it is critical to transform RGB colors into a range between 0 and 1, which allows various applications to be easily implemented.

3.5 Data Augmentation

Data augmentation, also known as data synthesis, is the process of creating new samples by applying task-specific modifications to existing data. It has been used as a regularizer to prevent overfitting by increasing the number of training samples and balancing classes in the data by simply augmenting under-represented classes. GAN, a new advancement in neural networks, has also been utilized as a data synthesis approach by synthesizing data from random noise to improve classifier performance [52].

The most common and successful method currently used for data augmentation is to do conventional affine and elastic Transformations including rotating or reflecting the original image, zooming in and out, shifting, applying distortion, and altering the color scheme to create new images. Despite their many benefits, basic classical procedures are sometimes insufficient to considerably increase the accuracy of neural networks or to solve the overfitting problem.

Data augmentation can be performed before or after the data is given into the model, or it can be done during the training process. This thesis employs Keras libraries to improve data during network training. The Image_Data_Generator class is used during training to produce new images from the original images. This is accomplished by randomly transforming the images.

The modified images are then used to train the model. We use the Image-Data Generator augmentation technique for this thesis but we will see some Augmentation techniques also.

3.5.1 Geometric transformations for data augmentation

In this section, we will see some basic geometric transformations applied to images to increase the training dataset. These are not by far the only techniques available, refer to [53], for the complete list. An essential concept when applying this transformation is that the transformations applied to labeled data do not change the semantic meaning of the label. For example, taking a case in computer vision, the scaled, rotated, flipped, and translated image of a coffee would still be an image of a coffee, and thus it is possible to apply these transformations to produce additional training data while preserving the semantic meaning of the label.

Flipping

Images are added by flipping the original image vertically or horizontally. Depending on the dataset, this may not be a label-preserving transformation.

Rotation

Rotation augmentations are performed by rotating the picture to the right or left on an axis ranging from 1 to 359 degrees. The rotation degree parameter and problem type have a large impact on the safety of rotation augmentations.

Scale

It can be resized up or down. The ultimate image size will be greater than the initial image size as you scale outward. Most image frameworks extract a piece of the new picture that is the same size as the old image.

3.5.2 Data Augmentation using Kera's Image-Data Generator

Image data augmentation is a technique that creates new images from existing ones. To do that, you make some small changes to them, such as adjusting the brightness of the image, rotating the image, or shifting the subject in the image horizontally or vertically. Augmenting the data for the proposed model to increase the number of images. The Image-Data Generator is used in the study to implement augmentation. To complement the data, we use an image data generator. The images that have been enhanced with preprocessed data are as follows.

Table 3.1 Augmentation techniques that have been used in the proposed model

Image_Data_Generator Augmentation Techniques We used.	Values
Rotation range	30
Width shift range	0.15
Height shift range	0.15
Shear range	0.25
Zoom range	0.25
Horizontal flip	True
Fill mode	Nearest
Cval	125

3.5.3 Generative adversarial networks for data augmentation

In this section, we will see the GAN approaches to data augmentation. GANs were first introduced by[54]. It is a type of generative model, which means it produces new data based on the training image. After the first introduction, several studies were done and fascinating results were achieved from images to video generation. We tried for our research but we could not Augment using GAN for our work due to requiring a lot of images and a lot of computational resources to produce real images.

Generator and discriminator

A GAN is made of two neural networks that compete against each other, the generator and the discriminator (see below figure). The generator, as the name implies, generates data similar to the training data while the discriminator checks the validity of the created data, i.e., tries to identify whether the data is from the generator or the real data. In other words, the generator tries to fool the discriminator, while the discriminator tries to prevent this from happening[55].

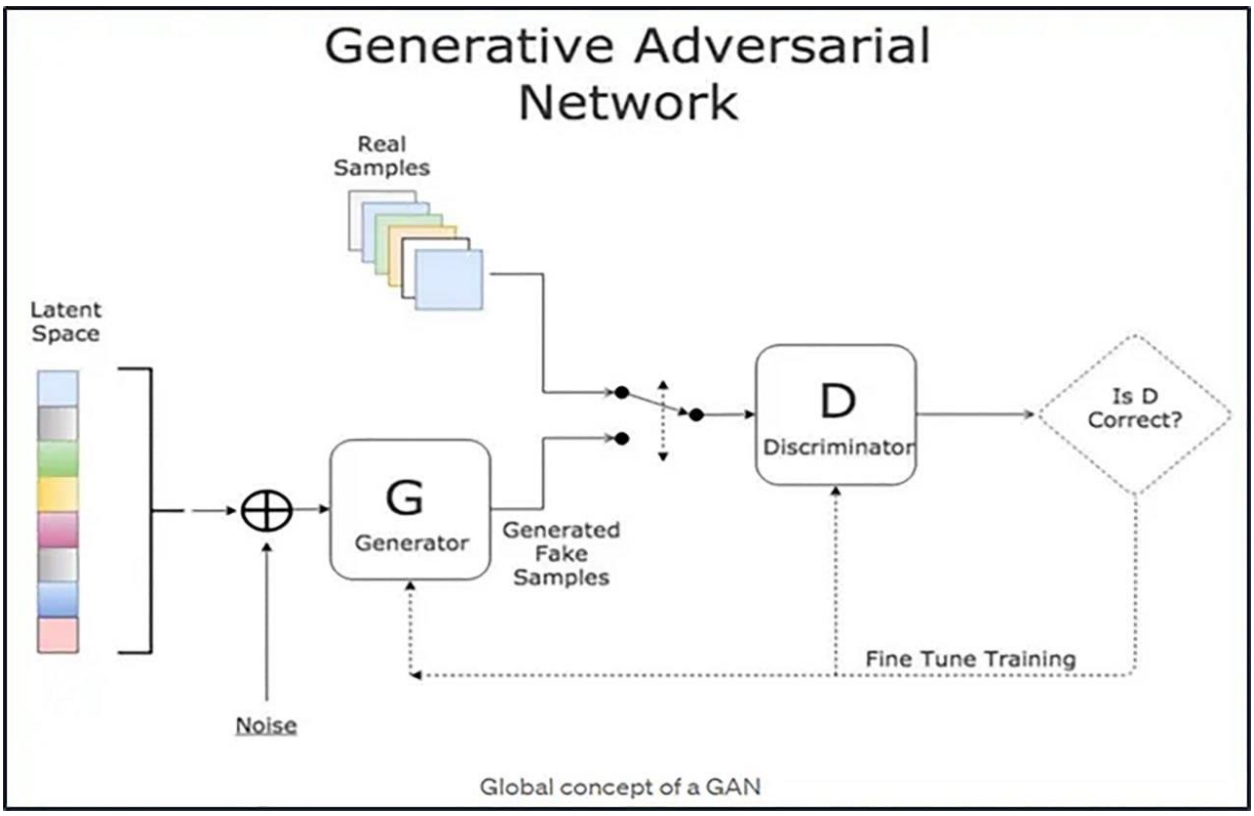


Figure 3.2 Concept of GAN[55]

3.6 Data splitting after augmented images for Proposed Model

The study uses Image_Data_Generator augmentation techniques to increase the size of the dataset. The augmented dataset is then automatically divided into three parts using Python code: 80% for training, 10% for testing, and 10% for validation. The validation dataset is a subset of the training dataset. The model is trained on the training dataset and validated on the validation dataset. This helps to ensure that the model is not overfitting to the training data. Finally, the performance of the model is tested on a separate test dataset that has not been seen by the model before.

3.7 Feature Extraction

Feature extraction is the process of extracting expressive features from derived data or images, such as color, shape, size, and texture. This would provide non-redundant information, which would make learning and categorization activities easier. These retrieved features are important for obtaining useful information from images and are utilized for object recognition and disease

identification using a set of predefined categories. Feature extraction is the part where the network learns to detect different high-level features from the input images. The image stores pixel values, and some characteristics are shown in each pixel value. Color, texture, and shape are some general characteristics of the picture. In this research work, we used CNN algorithms for feature extraction.

Convolutional Neural Networks are used to extract features and classify objects. The neural network is important for feature extraction to perform image classification. The components of data that you care about that are fed over the network are known as features. The features of an object in the case of image recognition are groups of pixels, such as edges and points that the network analyzes for patterns.

As[55], and[56], state, CNN is a strong method for extracting the best features from a huge dataset automatically. The proposed model in this study is based on domain experts for labeling the dataset and the physical appearances of the Coffee leaves and beans. To train and test our proposed model, we employed CNN-based Transfer Learning. To extract features, different layers of CNN are stacked on top of each other.

Classification

Classification operation is done in fully connected layers. In that fully connecting layer, the features are combined to create a model. Finally, a SoftMax activation function is used to convert the input feature vector into target classes. These connected layers are based on the SoftMax activation function to compute the class's scores. The input of the SoftMax classifier is a vector of features resulting from the learning process and the output is a probability that an image belongs to a given class. The identification component classifies sample image of coffee leaves and beans into disease groups based on image attributes retrieved from the image. The component includes feature set preparation and the detection of Coffee leaf and bean disease to accomplish this. We initially constructed a feature set utilizing CNN techniques based on the features retrieved using feature extraction components. After that, we trained a classifier model. The trained classifier is then used to determine the kind of disease present in coffee leaves and beans Recommend what action can be taken.

3.8 Evaluation Methods

Various performance metrics have been used to assess the performance of the model. The performance of any machine learning model is determined using measures such as true positive rate, false-positive rate, true negative rate, and false-negative rates[57]. The most widely used metrics are confusion matrix, accuracy, precision, recall, and f1-score.

Confusion Matrix

It describes the prediction results of each class, and it is represented by a matrix of the true class labels versus the predicted class labels. It contains the number of truly predicted instances and misclassified instances for each class. It has four dimensions True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN).

Understand TP, TN, FP, and FN in terms of coffee disease and health.

True Positive (TP): You predicted positive and it's true. For example, you predicted that a coffee has disease and is.

True Negative (TN): You predicted negative and it's true. For example, you predicted that a coffee is not diseased, and is not.

False Positive (FP): You predicted positive and it's false. For example, you predicted that a coffee is diseased but is not.

False Negative (FN): You predicted negative and it's false. For example, you predicted that a coffee is not diseased but is.

Accuracy: the percentage of true positives (both true positives and true negatives) along with the whole population.

$$\textbf{Accuracy} = \frac{(\boldsymbol{TP+TN})}{(\boldsymbol{TP+TN+FP+FN})}$$

Precision: is the percentage of true positives along with the whole positives. Mathematically, it is expressed as:

$$\textbf{Precision} = \frac{\boldsymbol{TP}}{\boldsymbol{P}}$$

Recall or sensitivity: is the percentage of true positives along with the whole true or right data. It is a quantitative description of how well the model avoids false negatives. It also called this type of measurement a true positive rate or hit rate.

$$\textbf{Recall} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP}+\boldsymbol{FN})}$$

Error Rate: The error rate is the percentage of entries that are wrongly categorized. It is determined by dividing the total number of inaccurate predictions by the total number of observations in the dataset.

$$\textbf{Error rate} = \frac{\boldsymbol{FP}+\boldsymbol{FN}}{(\boldsymbol{FP}+\boldsymbol{FN}+\boldsymbol{TP}+\boldsymbol{TN})}$$

F1-score: is the average accuracy and recall weighted. Precision and recall both contribute equally to the F1-score.

$$\textbf{F1-score} = \frac{\mathbf{2}*(\boldsymbol{Precision}*\boldsymbol{recall})}{(\boldsymbol{precision}+\boldsymbol{recall})}$$

3.9 Tools Used

3.9.1 Software Tools

Python: Python is a programming language that is easy to read and write, and it can be used to create a wide variety of programs, from small scripts to large-scale applications. Python is a good choice for machine learning because it has many libraries and tools that are specifically designed for this task. Python's design philosophy emphasizes code readability, which is why it uses significant whitespace. This means that the amount of space between keywords and identifiers is important for the syntax of the code. Python's language constructs and object-oriented approach also aim to help programmers write clear and logical code. There are two main versions of Python: Python 2 and Python 3. Python 2 is no longer supported, so it is recommended to use Python 3 for all new projects. For this study, we will be using Python 3. This is because Python 2 will no longer be available after 2020, and mass Python 3 adoption is the clear direction of the future.

Tensor Flow: TensorFlow is a free and open-source machine learning library that can be used for a variety of tasks, but is especially well-suited for training and deploying deep neural networks. It is used for both research and production at Google. TensorFlow is a complete platform for machine learning, with a wide range of tools, libraries, and community resources that make it easy for researchers to advance the state-of-the-art in ML and for developers to build and deploy ML-powered applications. TensorFlow was developed by the Google Brain team and released to the public in 2015.

Kera's: Keras is a free and open-source deep learning framework for Python. It is designed to make it easy to experiment with and build deep neural networks. Keras is built on top of other popular machine learning libraries, such as TensorFlow, Theano, and Cognitive Toolkit (CNTK). However, as of version 2.4, only TensorFlow is supported. Keras is known for being user-friendly, modular, and extensible. It is also the most popular deep learning framework among the top 5 winning teams on Kaggle and is used by leading organizations such as Google, Square, Netflix, Huawei, and Uber.

Anaconda: Anaconda is a free and open-source distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

Matplotlib: Matplotlib is a library for creating plots and charts in Python. It is similar to MATLAB in terms of its functionality, but it is free and open-source. Pyplot is a module in Matplotlib that provides a MATLAB-like interface. It allows you to create plots and charts in Python using a simple, easy-to-learn syntax. Matplotlib can be used to create a wide variety of plots, including line plots, bar charts, histograms, scatter plots, and 3D plots. It can also be used to create interactive plots that allow users to zoom, pan, and select data points. Matplotlib is a powerful tool for data visualization, and it is widely used in science, engineering, and other fields.

NumPy: NumPy is a Python library for scientific computing. It provides efficient ways to work with large, multidimensional arrays and matrices. Google Collab is a free cloud service that

provides access to GPU and CPU resources for machine learning and data science. It is based on Google's TensorFlow AI framework. Google made Collab free to the public in 2017, possibly to make it a standard for teaching machine learning and data science. It may also be a way to build a customer base for Google Cloud APIs. Google Collab offers the following.

3.9.2 Hardware Tools

We used the HD 360 x 360 pixels digital camera of Samsung to capture the sample images from the field. This study used Google Collab to implement the CNN algorithm for preprocessing data, extracting features, and training and testing models. Google Collab is a cloud service that provides free access to GPU and CPU resources for machine learning and data science. It is a good choice for this study because it meets the requirements of the CNN algorithm, which include a GPU, a processing speed of 2.8 GHz, and 12 GB of internal memory.

3.10 Coffee Leaf Disease Detection Deployment

We chose to use Flask to deploy our deep learning model because it is easy to use, supports integrated unit testing, and is faster than other tools such as Django. Additionally, the researcher has experience with the Python web framework. Flask is a Python web framework that allows us to build web applications. It supports extensions to add additional functionality to applications. Flask is a good choice for deploying deep learning models because it is lightweight and flexible. It is also easy to learn and use, even for developers with limited experience in web development. Additionally, Flask supports a wide range of extensions, including extensions for machine learning and data science. This makes it easy to add additional functionality to your web application, such as the ability to make predictions using your deep learning model.

CHAPTER FOUR

4 EXPERIMENTAL RESULT AND DISCUSSION

4.1 Introduction

This chapter discusses the experiments conducted and the results obtained while testing the proposed system architecture for coffee leaf and bean disease detection and classification. The experiments demonstrate the effectiveness of transfer learning for this task. We described the dataset used in training and testing for this research, tools used for implementation, and experimental results for the model, and presented the results compared with different experiments. We used CNN from scratch, CNN-based transfer learning and the last experiment is the comparison of the proposed model of CNN from scratch and CNN-based transfer learning like Resnet 50, VGG 19, Inception v3 and We have used four performance measure metrics to evaluate the performance of the proposed model. These are accuracy, precision (positive predictive value), recall (True positive rate or Sensitivity), and F1-score. We also used a confusion matrix to measure the performance of each class.

4.2 Dataset Used

We have collected an image dataset from the Wondoganat Agricultural Research Awada branch Located in Yirgalem. After collecting the images, resizing the images, then filtering these images and using data augmentation to reduce the overfitting problems due to the scarcity of data. We have used 5010 images for this research, from these data 80% of the dataset is for training data (fed into the model to train upon), 10% for validation data, and 10% for testing data. The dataset split ratio is (80:10:10).

4.3 Algorithm Selection

This thesis uses a deep learning algorithm called a convolutional neural network (CNN) for coffee leaf and bean disease detection. CNN models are well-suited for image processing because they can automatically extract features from images. CNN models have several layers, including convolution layers, pooling layers, activation layers, and dense layers. The convolution layers extract features from the input images, the pooling layer is used for downsampling and the activation layers help to improve the learning process. The model took images of coffee leaves and beans as input, preprocessed the data, and then categorized the images into five classes:

Healthy, CBD (coffee berry disease), CLR (coffee leaf rust), PLS (phoma leaf spot), and BES (brown eye spot).

4.4 Comparison of CNN Model

This study compares CNN from scratch with three pre-trained architectures for Coffee leaf and bean disease detection: VGG 19, InceptionV3, and ResNet50. The study specifically compares the CNN from scratch with three pre-trained architectures by Hyperparameter tuning the number of parameters and layers.

4.4.1 Convolutional Neural Network (CNN)

Convolutional neural networks (CNNs) are a type of deep learning algorithm that are specifically designed to process pixel data. They are one of the most popular algorithms for image recognition, image clustering, classification, and object detection. CNNs work by extracting features from images. This is done by applying a series of convolution operations to the image. Convolution operations are a way of detecting patterns in images. The extracted features are then passed through a series of layers, which learn to classify the images. The layers in a CNN are typically arranged hierarchically, with each layer learning to detect more complex features. CNNs are very effective at image classification tasks.

4.4.2 VGG-19

VGG-19 is a convolutional neural network (CNN) with 19 layers, making it one of the deepest CNNs ever built. It was developed by Karen Simonyan and Andrew Zisserman at the University of Oxford in 2014[50]. The network was trained on a dataset of over 1 million images from the ImageNet database. VGG-19 has been pre-trained to classify up to 1,000 different Classes. The network was trained on colored images with a resolution of 224x224 pixels. The name VGG stands for **Visual Geometry Group**, which is the research group at the University of Oxford where Simonyan and Zisserman work. The number 19 refers to the number of layers in the network. VGG-19 is a deep CNN, which means that it has many layers. Deep CNNs can learn more complex features from images than shallow CNNs. VGG-19 is a powerful CNN that has

been used for a variety of image recognition tasks. It has been used to classify images, segment images, and detect objects in images. VGG-19 is also a popular choice for transfer learning, which is a technique for using a pre-trained CNN for a new task.

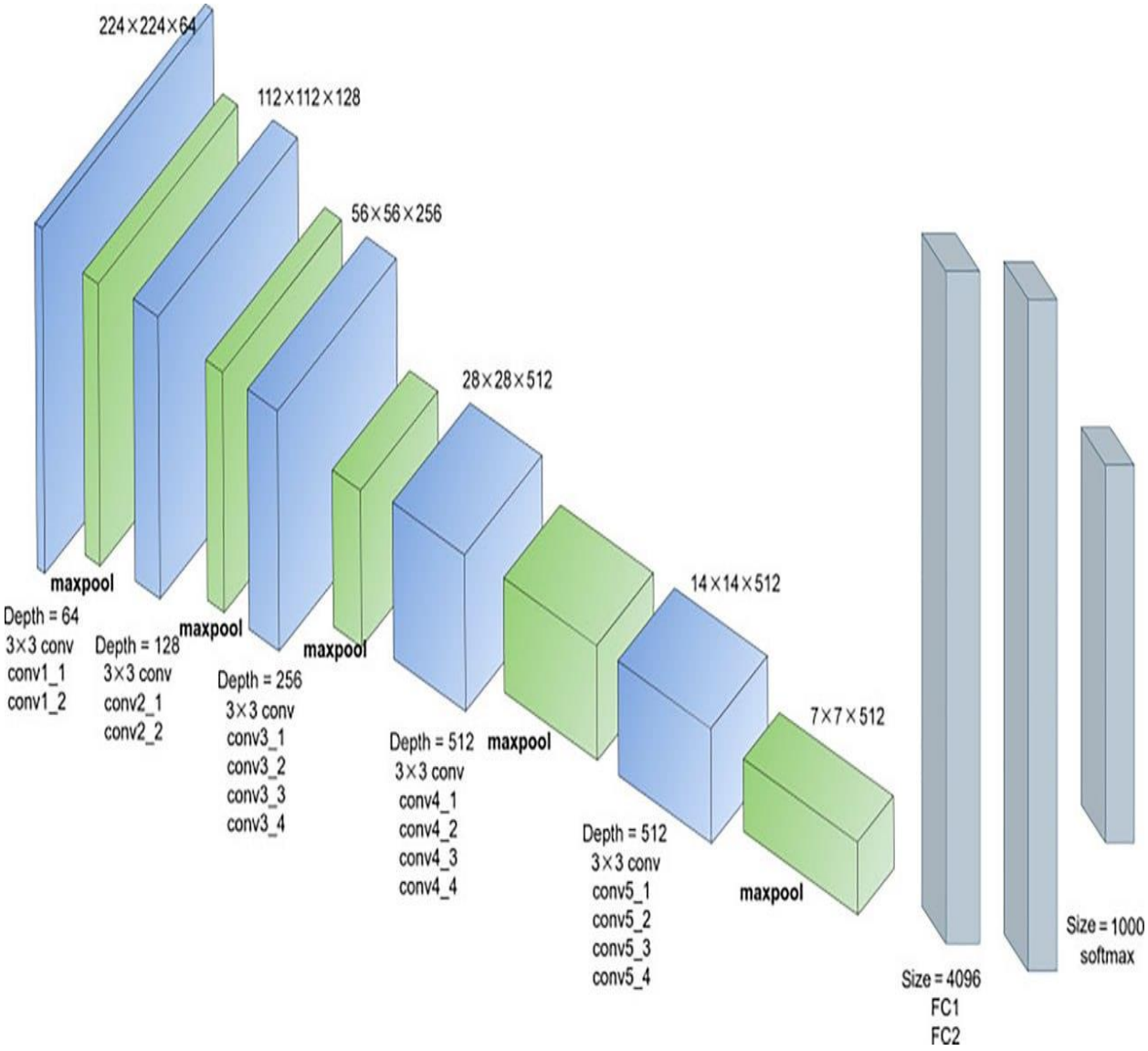


Figure 4.1 VGG-19 architecture[50]

Here is a brief architecture of VGG-19:

- **Input:** The input to VGG-19 is a 224x224x3 RGB image.

- **Convolutional layers:** VGG-19 has 16 convolutional layers. Each convolutional layer has a kernel size of 3x3 and a stride of 1. The convolutional layers are followed by a max pooling layer with a kernel size of 2x2 and a stride of 2.
- **Fully connected layers:** VGG-19 has 3 fully connected layers. The first fully connected layer has 4096 neurons, the second fully connected layer has 4096 neurons, and the third fully connected layer has 1000 neurons. The 1000 neurons represent the 1000 different object categories in the ImageNet dataset.
- **SoftMax layer:** The final layer of VGG-19 is a SoftMax layer, which is used to classify the input image into one of the 1000 object categories.

4.4.3 ResNet 50 (Residual Network)

ResNet 50 is a convolutional neural network (CNN) with 50 layers. It was developed by Microsoft in 2015 and is a popular choice for image classification tasks. ResNet 50 was trained on the ImageNet dataset, which contains over 1 million images of 1000 different object categories. The name ResNet stands for **Residual Network**. Residual networks are a type of CNN that uses **skip connections** to help with training deep networks. Skip connections allow the network to learn residual functions, which are functions that add to the input to the network. This makes it easier for the network to learn complex features from images. ResNet 50 is a powerful CNN that has been used for a variety of image recognition tasks. It has been used to classify images, segment images, and detect objects in images. ResNet 50 is also a popular choice for transfer learning, which is a technique for using a pre-trained CNN for a new task.

ResNet50 and **VGG19** are both convolutional neural networks (CNNs) that are used for image classification tasks. ResNet50 has 50 layers, while VGG19 has 19 layers. This means that ResNet50 is a deeper network than VGG19. However, ResNet50 outperforms VGG19 on image classification tasks, even though it has lower complexity. This is because ResNet50 uses **skip connections**, which help the network learn complex features from images. ResNet50 has been improved several times since it was first introduced. The latest version, ResNet50V2, has been shown to outperform the original ResNet50 on a variety of image classification tasks. There are also several other versions of ResNet, including ResNet101, ResNet152, ResNet50V2,

ResNet101V2, and ResNet152V2. These networks have more layers than ResNet50, and they can achieve even higher accuracy on image classification tasks.

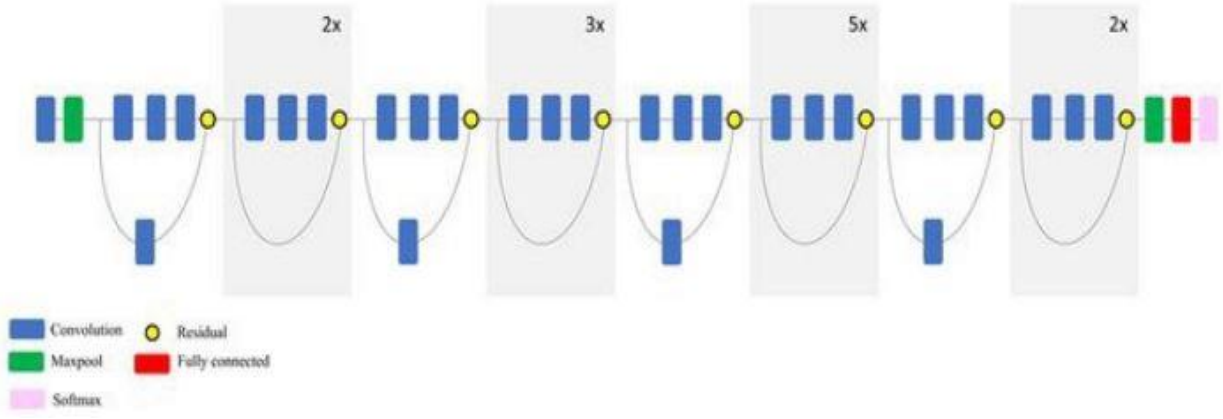


Figure 4.2 Resnet 50 architecture

Here is a brief architecture of ResNet50:

- **Input:** The input to ResNet50 is a 224x224x3 RGB image.
- **Residual blocks:** ResNet50 has 34 residual blocks. Each residual block consists of two convolutional layers, followed by a skip connection. The skip connection adds the input to the output of the convolutional layers. This allows the network to learn residual functions, which are functions that add to the input to the network.
- **Fully connected layers:** ResNet50 has 3 fully connected layers. The first fully connected layer in this neural network has 2048 neurons. This means that each neuron in this layer is connected to all 2048 neurons in the previous layer. The second fully connected layer has 1024 neurons, and the third fully connected layer has 1000 neurons. This means that the number of neurons decreases as the neural network progresses, which helps to prevent overfitting.
- **SoftMax layer:** The last layer is a SoftMax layer, which outputs the probabilities of the image belonging to each of the 1000 object categories.

4.4.4 Inceptionv3 (Google Net)

InceptionV3 is a convolutional neural network (CNN) with 50 layers. It was developed by Google in 2015 and is a popular choice for image classification tasks. InceptionV3 was trained on the ImageNet dataset, which contains over 1 million images of 1000 different object categories. The name Inception comes from the way that the network is designed. InceptionV3 uses a technique called **inception modules**, which allow the network to learn features at different scales. This makes InceptionV3 more powerful than other CNNs that only learn features at a single scale. InceptionV3 is a powerful CNN that has been used for a variety of image recognition tasks. It has been used to classify images, segment images, and detect objects in images.

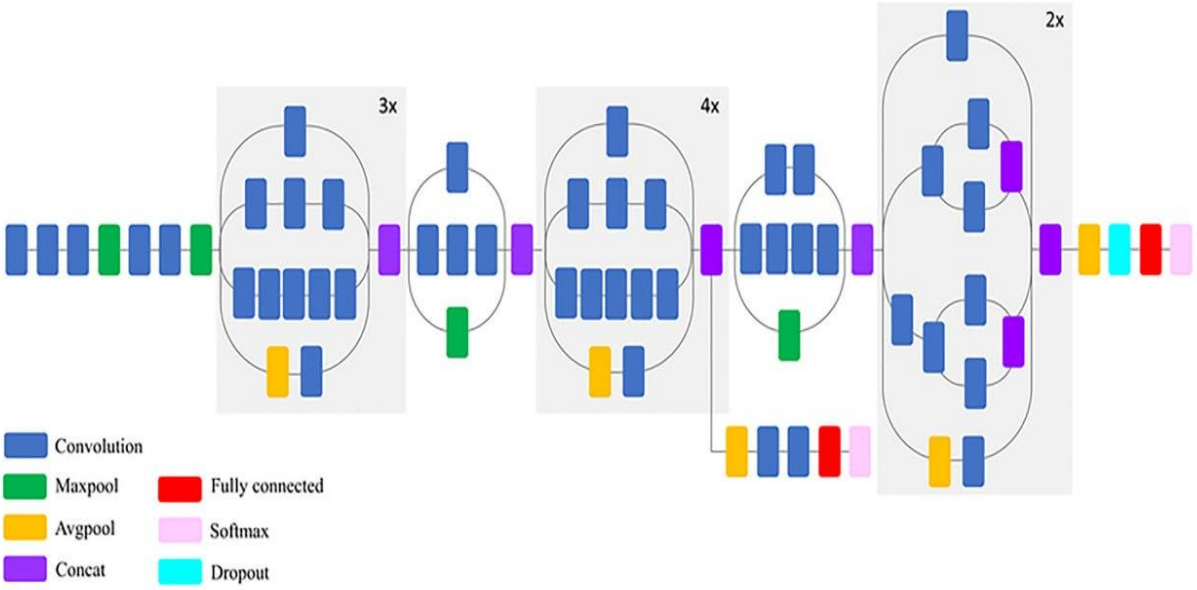


Figure 4.3 InceptionV3 architecture

Here is a brief architecture of InceptionV3:

- **Input:** The input to InceptionV3 is a 299x299x3 RGB image.

- **Inception modules:** InceptionV3 has 29 inception modules. Each inception module consists of several convolutional layers, followed by a pooling layer. The convolutional layers in an inception module are arranged in a way that allows the network to learn features at different scales.
- **Fully connected layers:** InceptionV3 has 3 fully connected layers. The first fully connected layer has 4096 neurons, the second fully connected layer has 1024 neurons, and the third fully connected layer has 1000 neurons.
- **SoftMax layer:** which outputs the probabilities of the image belonging to each of the 1000 object categories.

4.5 Hyper-parameter Tuning

The parameters of the pre-trained models that were chosen were fine-tuned to make them suitable for the task at hand and consistent with the dataset. The table below depicts the hyperparameters used in this study in order to fine-tune the CNN architecture.

In all models, the weights were initialized with the ImageNet pre-trained weights. The dense layer was removed and replaced with new convolutional layers, including batch normalization and dropout layers. Batch normalization helps to increase the speed of learning and acquire higher accuracy values by normalizing the activations of the convolutional layers. Dropout randomly sets a fraction of the neurons in each layer to zero, which helps to prevent overfitting.

The results showed that the fine-tuned models outperformed the original pre-trained models on the validation dataset. This suggests that fine-tuning can be an effective way to improve the performance of CNNs on a specific task.

- **Learning rate:** The learning rate is a hyperparameter that controls how much the model's weights are updated during training. A lower learning rate will result in a slower convergence, but it may also be less likely to overfit the training data.
- **Batch size:** The batch size is the number of training examples that are used to update the model's weights at each iteration. A larger batch size can result in faster training, but it may also require more memory.

- **Epochs:** An epoch is a complete pass through the training data. The number of epochs is a hyperparameter that controls how many times the model will be trained on the data.

Table 4.1 hyperparameter tuning

S/No	Hyperparameter	The way we find the best parameter
1	**Input size**	**224x224, 256x256, 299x299**
2	Batch-size	**16, 32, 64, 128**
3	Epoch	**5, 10, 20 for the pre-trained model and up to 100 for CNN**
4	Optimizers	**Adam, SDG, RMSprop**
5	Learning-rate	**1e-1 up to 1e-5**
6	Dropout	**[0-1]**
7	Activation function	**ReLU, SoftMax**
8	Batch-normalization	**Yes**
9	Weight	**ImageNet**

4.6 Training and Testing Models

Experiment on VGG 19 Model

In VGG 19 model we conducted different experiments, first, we split the dataset into 80:10:10, with 224 x 224 pixels, 256 x 256 pixels, and 299 x 299 pixels images then we checked the accuracy. We used train and test the dataset throughout all experiments 80:10:10 means we used 80% for training, 10% for validation, and 10% for testing. When developing the VGG 19 model there is no standard to select the components and parameters, we tested an experiment with different parameters and we got good accuracy by using 224 x 224pixel image size, 64 batch-size, Adam optimizer with le-1 learning rate, GlobalAveragePoling2D, 0.001 Dropout and we use Batch Normalization. We also vary our epochs from 5-20 epochs. The epoch number is

estimated by using the EARLY STOP algorithm. The experiment in this section is done to measure the training, validation, and testing accuracy of the proposed VGG 19 model.

As we stated, above We have got good training accuracy of 96.3%, validation accuracy of 91%, and 89.01% testing accuracy by using a stated parameter. The training and validation process of the model is shown as follows.

```
[14] model_history = trainModel(model = model, epochs = 5, optimizer = 'adam', batch_size = 64, callbacks=cb)

Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.5949 - accuracy: 0.7576
Epoch 1: accuracy improved from -inf to 0.75761, saving model to best_m.h5
63/63 [==============================] - 557s 9s/step - loss: 0.5949 - accuracy: 0.7576 - val_loss: 1.1396 - val_accuracy: 0.5920
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.3323 - accuracy: 0.8733
Epoch 2: accuracy improved from 0.75761 to 0.87332, saving model to best_m.h5
63/63 [==============================] - 28s 441ms/step - loss: 0.3323 - accuracy: 0.8733 - val_loss: 0.4955 - val_accuracy: 0.8080
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.2125 - accuracy: 0.9222
Epoch 3: accuracy improved from 0.87332 to 0.92219, saving model to best_m.h5
63/63 [==============================] - 27s 421ms/step - loss: 0.2125 - accuracy: 0.9222 - val_loss: 0.3018 - val_accuracy: 0.8820
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.1304 - accuracy: 0.9526
Epoch 4: accuracy improved from 0.92219 to 0.95262, saving model to best_m.h5
63/63 [==============================] - 28s 446ms/step - loss: 0.1304 - accuracy: 0.9526 - val_loss: 0.3177 - val_accuracy: 0.8920
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.1049 - accuracy: 0.9633
Epoch 5: accuracy improved from 0.95262 to 0.96334, saving model to best_m.h5
63/63 [==============================] - 27s 423ms/step - loss: 0.1049 - accuracy: 0.9633 - val_loss: 0.2938 - val_accuracy: 0.9100
```

Figure 4.4 Training and validation process of the VGG-19 model

As clearly described in the above figure, the training accuracy and validation accuracy are increased throughout the epochs. The validation accuracy of the trained model is 91.00 %, which performs in a good way.

The precision, recall, and f1-score for the VGG 19 model are shown in the below Figure.

.

	precision	recall	f1-score	support
BES	0.79	0.81	0.80	100
CBD	0.99	1.00	1.00	100
CLR	0.93	0.88	0.90	100
Healthy	0.95	0.95	0.95	100
PLS	0.94	0.95	0.95	100
accuracy			0.92	500
macro avg	0.92	0.92	0.92	500
weighted avg	0.92	0.92	0.92	500

Figure 4.5 Precision, recall, and f1-score for the VGG 19 model

As described in the above Figure, we evaluate the model by using performance metrics precision, recall, and f1-score. The confusion matrix of the VGG 19 model is depicted in the below Figure.

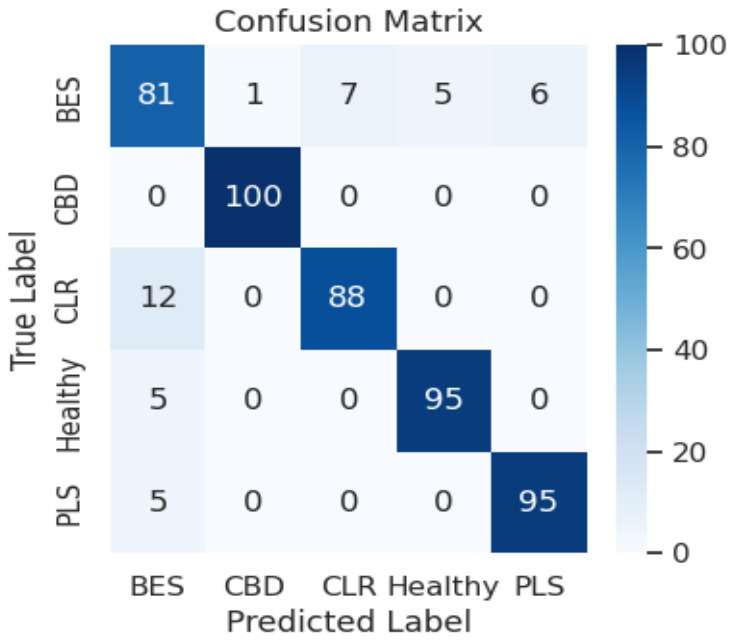


Figure 4.6 Confusion matrix of the VGG-19 model

Experiment on Resnet-50 Model

As we have done for VGG 19, when we developed the Resnet 50 model, we performed many experiments as we explained before. we have tested an experiment with different parameters and

we got good accuracy by using 224 x 224-pixel image size, 64 batch-size, Adam optimizer without initializing learning rate, GlobalAveragePoling2D, 0.001 Dropout and we use Batch Normalization. We also vary our epochs from 5-20 epochs. We have achieved a training accuracy of 97.6% validation accuracy of 94.4% and 95.0% test accuracy.

```
✓ 9m [13] model_history = trainModel(model = model, epochs = 5, optimizer = "Adam", batch_size = 64, callbacks=cb)

Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.4766 - accuracy: 0.8129
Epoch 1: accuracy improved from -inf to 0.81294, saving model to best_m.h5
63/63 [==============================] - 418s 6s/step - loss: 0.4766 - accuracy: 0.8129 - val_loss: 0.5938 - val_accuracy: 0.7580
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.2134 - accuracy: 0.9214
Epoch 2: accuracy improved from 0.81294 to 0.92139, saving model to best_m.h5
63/63 [==============================] - 25s 403ms/step - loss: 0.2134 - accuracy: 0.9214 - val_loss: 0.3537 - val_accuracy: 0.8520
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.1118 - accuracy: 0.9582
Epoch 3: accuracy improved from 0.92139 to 0.95821, saving model to best_m.h5
63/63 [==============================] - 25s 403ms/step - loss: 0.1118 - accuracy: 0.9582 - val_loss: 0.2157 - val_accuracy: 0.9120
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.0750 - accuracy: 0.9744
Epoch 4: accuracy improved from 0.95821 to 0.97438, saving model to best_m.h5
63/63 [==============================] - 26s 407ms/step - loss: 0.0750 - accuracy: 0.9744 - val_loss: 0.2179 - val_accuracy: 0.9180
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.0694 - accuracy: 0.9761
Epoch 5: accuracy improved from 0.97438 to 0.97612, saving model to best_m.h5
63/63 [==============================] - 26s 409ms/step - loss: 0.0694 - accuracy: 0.9761 - val_loss: 0.1811 - val_accuracy: 0.9440
```

Figure 4.7 Training and validation process of the Resnet50 model

As clearly described in the above figure, the training accuracy and validation accuracy of the ResNet50 model are increased throughout the epochs. The validation accuracy of the trained model is 94.4%, which is a good performance. This indicates that the model can generalize well to unseen data. The precision, recall, and F1-score for the ResNet50 model are shown in the following figure. Precision is the fraction of predicted positive instances that are positive. Recall is the fraction of actual positive instances that are predicted positive. F1-score is a measure of both precision and recall. These scores are all very good, which indicates that the model can accurately identify both positive and negative instances.

As clearly described in Figure 4.7, to evaluate the accuracy of each class in the model. The training and validation accuracy vs. loss curve of Resnet 50 is shown in Figure 4.8.

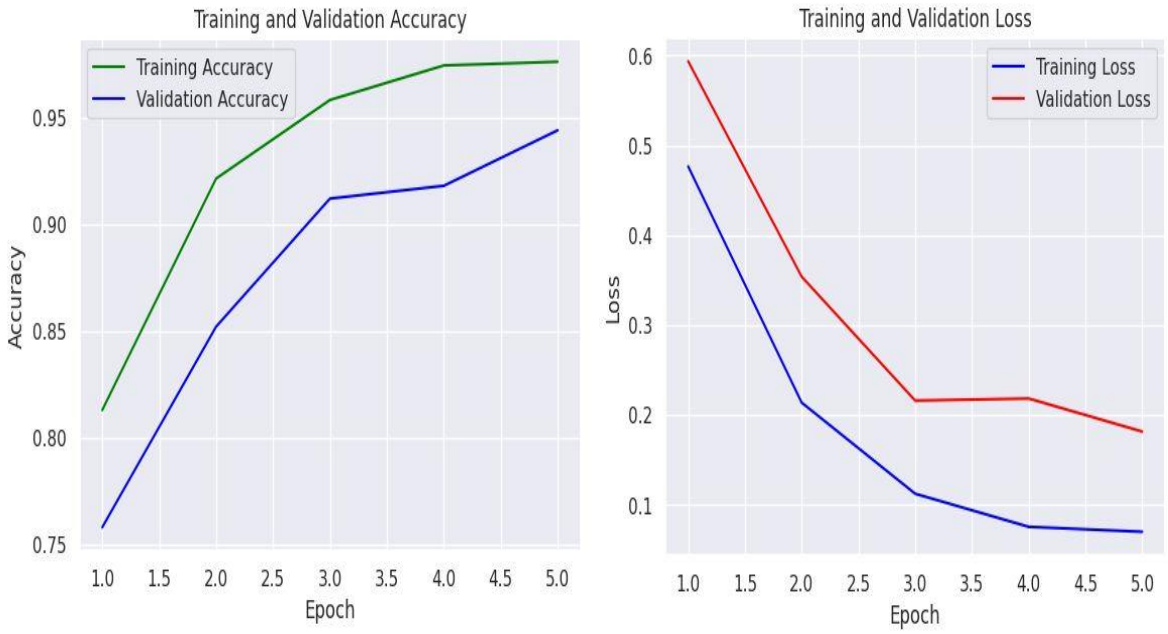


Figure 4.8 Training and Validation Accuracy and loss Resnet50 model

As clearly described in Figure 4.8, the training accuracy and validation accuracy are increased throughout the curve. The gap between the training and validation accuracy is very few, almost similar. The training and validation loss decreases throughout the curve and there is also a low gap between them.

Overall, the results of this experiment show that the ResNet50 model is a good choice for image classification tasks like coffee disease detection. It can achieve high accuracy and good precision, recall, and F1-scores.

The precision, recall, and f1-score for the Resnet 50 model are shown in the below Figure.

	precision	recall	f1-score	support
BES	0.91	0.84	0.87	100
CBD	1.00	1.00	1.00	100
CLR	0.92	0.97	0.94	100
Healthy	0.96	0.99	0.98	100
PLS	0.96	0.95	0.95	100
accuracy			0.95	500
macro avg	0.95	0.95	0.95	500
weighted avg	0.95	0.95	0.95	500

Figure 4.9 Precision, recall, and f1-score for the Resnet50 model

As described in the above Figure, we evaluate the model by using performance metrics precision, recall, and f1-score. The confusion matrix of the Resnet 50 model is depicted in the below Figure.

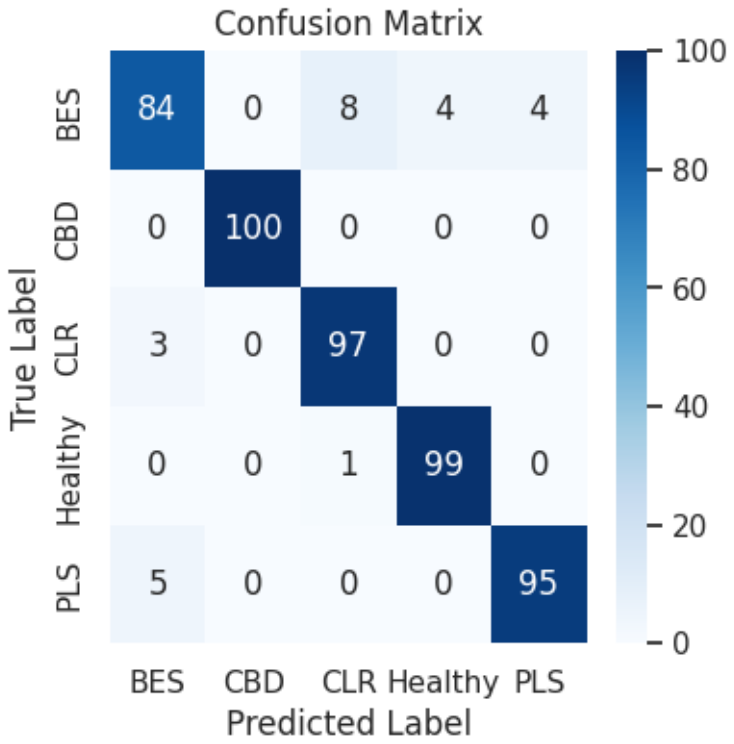


Figure 4.10 Confusion matrix of Resnet50 model

Experiment On Inceptionv3 (Google Net)

In this experiment, when developing the InceptionV3 model, we tested different parameters and found that the best accuracy was achieved with the following settings:

- Image size: 299 x 299 pixels
- Batch size: 128
- Optimizer: Adam with a learning rate of 1e-3
- Pooling layer: GlobalAveragePooling2D
- Dropout rate: 0.001
- Batch normalization: Yes
- Epochs: 10

With these settings, we achieved a training accuracy of 93.3% and a validation accuracy of 88.7%.

Experiment on CNN Model from scratch

Convolutional Neural Network is a powerful neural network that uses filters to extract features from images. Convolutional Neural Networks, like neural networks, are made up of neurons with learnable weights and biases. The output of a neuron is determined by the weighted sum of its inputs and an activation function. The whole network has a loss function and all the tips and tricks that we developed for neural networks still apply to Convolutional Neural Networks. There are four layers in Convolutional Neural Networks: Convolution, ReLU, Pooling, and Full Connectedness (Fully Connected Layer). The convolution layer extracts features of an image using a filter and image patch that strides over the input image. The ReLU layer replaces all negative pixel values in the feature map with zero while the pooling layer allows the feature map to be down-sampled after the ReLU layer to reduce the dimensionality. Max pooling computes the maximum of the local feature map. We use 256 x 256 x3 RGB channel. The model can be fit on the training dataset and passed to both the training dataset and the validation dataset. We used ReLU for the input Activation layer. The following shows the CNN model parameters.

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 conv2d (Conv2D)             (None, 222, 222, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 111, 111, 128)    0
 )

 conv2d_1 (Conv2D)           (None, 109, 109, 256)     295168

 max_pooling2d_1 (MaxPooling  (None, 54, 54, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 52, 52, 128)       295040

 max_pooling2d_2 (MaxPooling  (None, 26, 26, 128)      0
 2D)

 flatten (Flatten)           (None, 86528)             0

 final_dense (Dense)         (None, 5)                 432645

=================================================================
Total params: 1,026,437
Trainable params: 1,026,437
Non-trainable params: 0
```

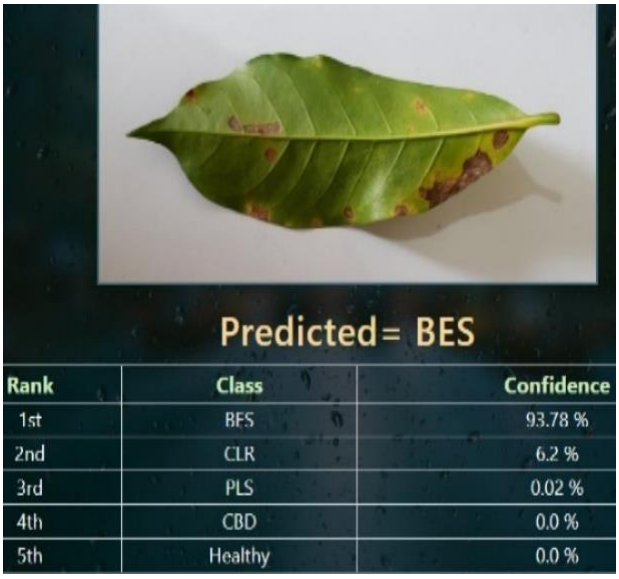
Figure 4.11 Number of parameters in the CNN model

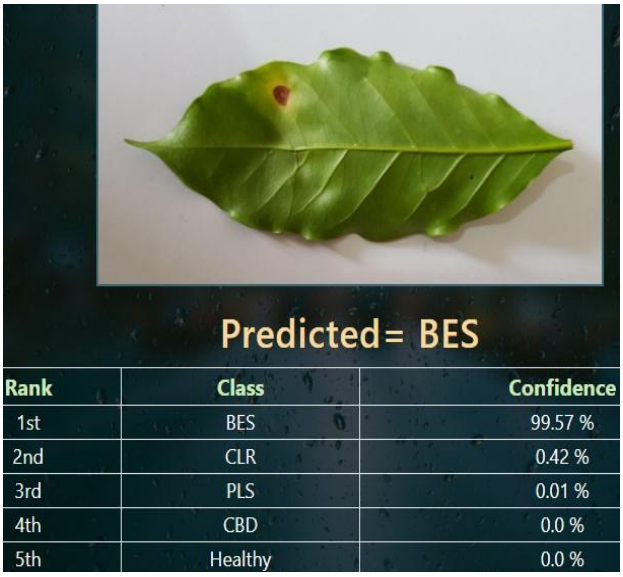
We have implemented CNN with Keras. We achieved a training accuracy of 94% validation accuracy of 90% test accuracy of 89.4% and test loss is 0.2364.

Testing Model with Confidence Scores for Each Disease

After fine-tuning a ResNet-50 model, we tested its performance on coffee disease identification using test images from the test dataset. The test results with the confidence score for each class of coffee disease are shown below. The results show that the model has a high confidence score, which indicates that it is good at identifying coffee diseases. Therefore, our proposed model can be used by coffee producers to get recommendations for managing coffee diseases.

Test Result of BES (Brown Eye Spot) class with its confidence





a. BES with 93.78% confidence score b. BES with 99.57% confidence score

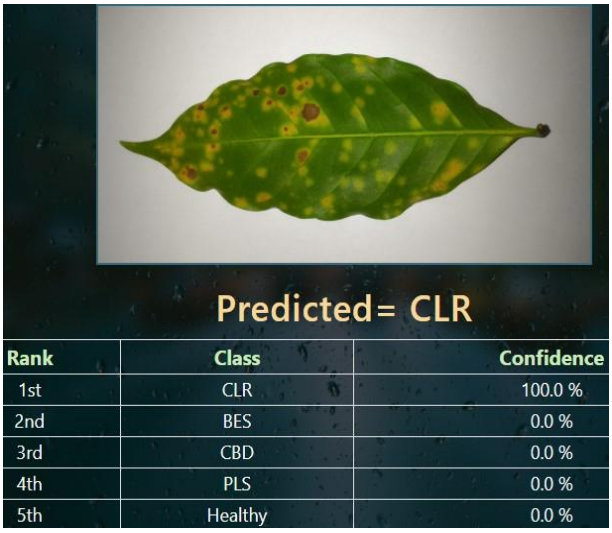
Figure 4.12 BES with confidence score

From the first image a, the model predicts Brown eye spot (BES) with 93.78% confidence, misclassifies it as Coffee Leaf Rust (CLR) with 6.2% confidence, and misclassifies it as Phoma Leaf Spot (PLS) with 0.02% confidence.

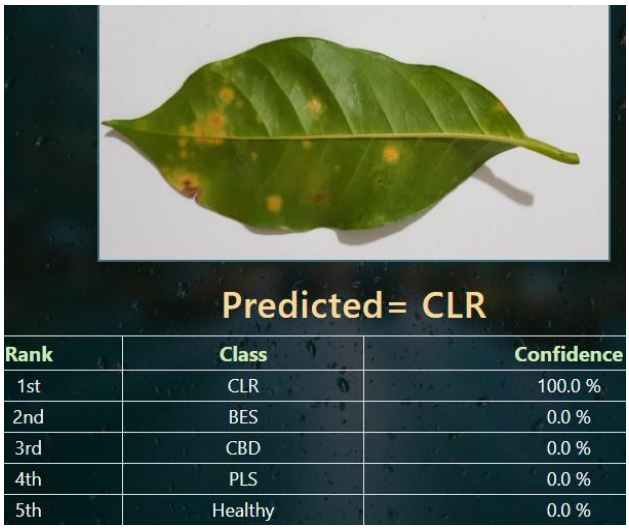
From the second image b, the model predicts Brown eye spot (BES) with 99.57% confidence, misclassifies it as Coffee Leaf Rust (CLR) with 0.42% confidence, and misclassifies it as Phoma Leaf Spot (PLS) with 0.01% confidence.

Test Result of ClR (Coffee Leaf Rust) class with its confidence

when we test ClR class our model is 100% confident as we can see from the below image.



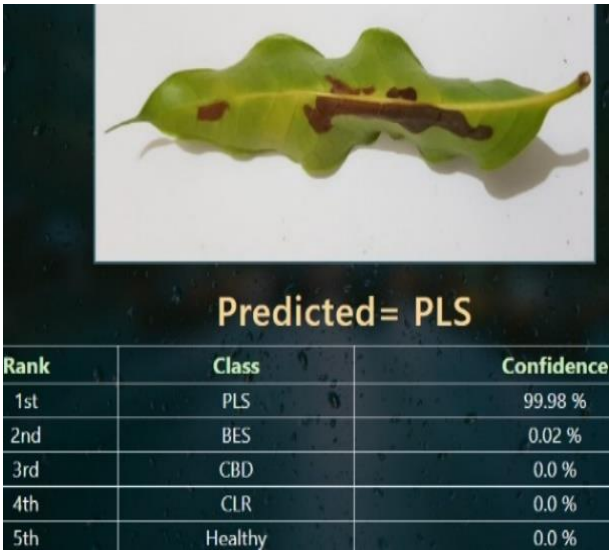
Rank	Class	Confidence
1st	CLR	100.0 %
2nd	BES	0.0 %
3rd	CBD	0.0 %
4th	PLS	0.0 %
5th	Healthy	0.0 %



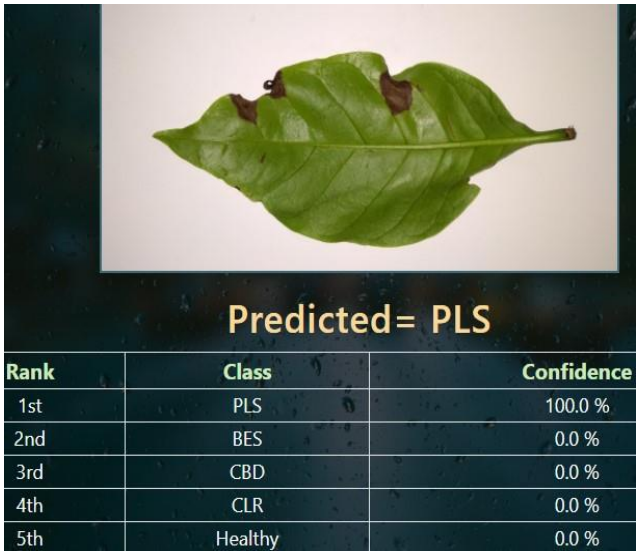
Rank	Class	Confidence
1st	CLR	100.0 %
2nd	BES	0.0 %
3rd	CBD	0.0 %
4th	PLS	0.0 %
5th	Healthy	0.0 %

Figure 4.13 CLR with confidence score

Test Result of PLS (Phoma leaf spot) class with its confidence



Rank	Class	Confidence
1st	PLS	99.98 %
2nd	BES	0.02 %
3rd	CBD	0.0 %
4th	CLR	0.0 %
5th	Healthy	0.0 %



Rank	Class	Confidence
1st	PLS	100.0 %
2nd	BES	0.0 %
3rd	CBD	0.0 %
4th	CLR	0.0 %
5th	Healthy	0.0 %

a. PLS with 99.98% confidence score b. PLS with 100% confidence score

Figure 4.14 PLS with confidence score

From the first image a, the model predicts Phoma Leaf Spot (PLS) with 99.98% confidence, misclassifies it as Brown Eye Spot (BES) with 0.02% confidence. And the second image b, model classifies correctly with 100% confidence.

Test Result of CBD (Coffee Berry Disease) class with its confidence

when we test CBD class our model is 100% confident as we can see from the below image.

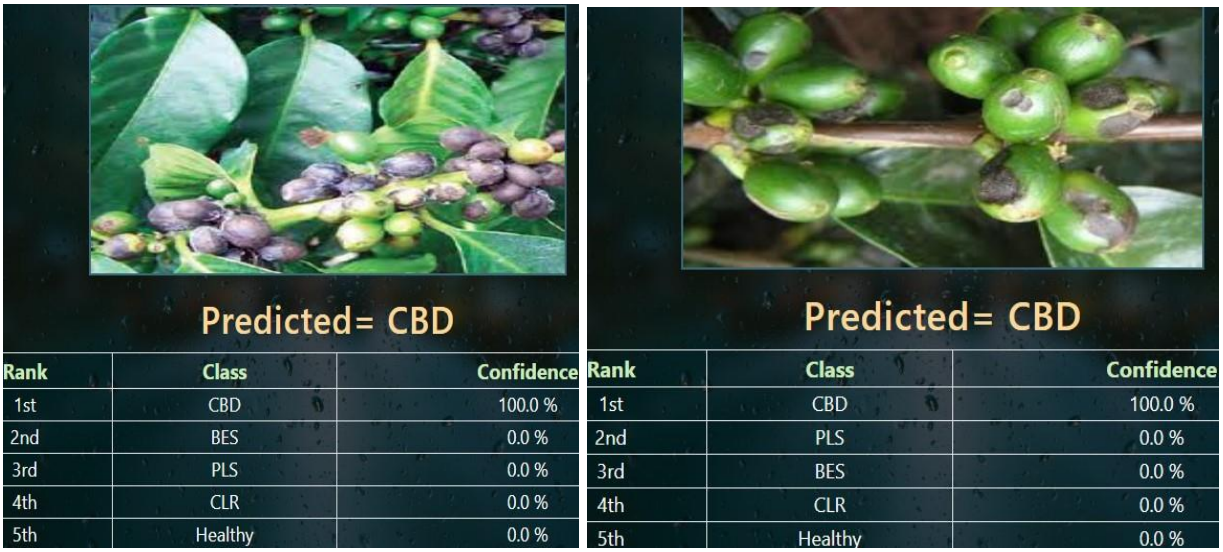


Figure 4.15 CBD with confidence score

4.7 Performance Comparison of VGG 19, Resnet 50, Inception v3, and CNN from scratch using our Datasets.

In our research, we tried to compare three CNN architectures and select the best state-of-the-art model for classifying coffee diseases. We conducted different comparative analyses based on various evaluation metrics. Based on this, we compared three pre-trained models, VGG19, ResNet50, and Inception V3, with a CNN trained from scratch.

As shown in the table below, ResNet50 achieved the most promising results in terms of both training and validation accuracy. It achieved a training accuracy of 97.6% and a validation accuracy of 94.4%. The ResNet50 model also had the fastest training time, taking only 8 minutes and 37 seconds to train for 5 epochs. The weight files matrix for ResNet50 is 122.58 MB.

In contrast, the VGG19 model took the same amount of time to train, but its weight files matrix is 180.55 MB. The VGG19 model also had the lowest validation accuracy than the ResNet50 models. The Inception V3 model had a validation accuracy that was slightly lower than the VGG19 model, but it took 13 minutes to train for 10 epochs. The weight files matrix for Inception V3 is 199.26 MB.

Overall, the ResNet50 model is the best choice for classifying coffee diseases. It has the highest accuracy, the fastest training time, and the smallest weight file matrix. In addition to the accuracy metrics, we also considered the computational cost of training and inference. The ResNet50 model has a lower computational cost than the other models, making it a more practical choice for real-world applications.

We believe that the ResNet50 model is a promising approach for classifying coffee diseases. It is accurate, efficient, and easy to use. The table below explains how we get good results for those algorithms.

Table 4.2 model comparisons

Parameter	Resnet 50	VGG 19	Inception V3	CNN
Input size	**224x224**	**224x224**	**299x299**	**256x256**
Batch size	**64**	**64**	**128**	**32**
Optimizer	**Adam**	**Adam**	**Adam**	**Adam**
Learning rate	**1e-1**	**1e-5**	**1e-1**	**1e-1**
Dropout	**[0-1]**	**[0-1]**	**[0-1]**	**[0-1]**
Activation function	**ReLU, SoftMax**	**ReLU, SoftMax**	**ReLU, SoftMax**	**ReLU, SoftMax**
Batch Normalization	**Yes**	**Yes**	**Yes**	**Yes**
Weight	**ImageNet**	**ImageNet**	**ImageNet**	**-**
Epoch	**10**	**10**	**15**	**50**
Training Time	**8 min and 37 sec**	**9 min**	**13 min**	**20**

Model size	**122.58 MB.**	**180.55 MB.**	**199.26 MB.**	**198 MB**
Resource	**GPU**	**GPU**	**GPU**	**GPU**
Training Accuracy	**97.6%**	**96.3%**	**93.3%**	**94%**
Validation Accuracy	**94.4%**	**91%**	**88.7%**	**90%**
Test Accuracy	**95.0%**	**92%**	**88%**	**89.3%**
Validation loss	**0.18%**	**0.29%**	**0.56**	**0.39%**
Test loss	**0.069%**	**0.10%**	**0.31%**	**0.23%**

4.8 Result Discussion

The overall experimental evaluation, conducted through the performance measure of Coffee leaf and bean disease identification models shows good results. Our algorithm for classifying Coffee leaf and bean disease was tested using sample data selected from the test dataset. We have implemented CNN and a pre-trained model to test the performance of the proposed model. The developed model has performed better in terms of a manual method than the proposed automated approach. The results of the experimental evaluation show that the developed model is a promising approach for automated coffee leaf and bean disease identification. The model can achieve high accuracy, precision, and recall, which makes it a valuable tool for coffee farmers and other stakeholders in the coffee industry.

For classifying Coffee leaf and bean disease identification to distinguish the better performing model we use four Dl algorithms within the same dataset. In this study classifying Coffee leaf and beans diseases was used to evaluate the accuracy and performance of the DL models/architectures. We developed CNN, VGG19, Resnet 50, and InceptionV3 DL models/architectures; we got good performance on training and testing results on the Resnet 50 model. By using the VGG 19, Resnet 50, and InceptionV3, training has been undertaken with three stages by increasing the epoch values of 5,10, and 15, Performing parameter tuning which we have stated in the above table as shown in the result with training and validation graph and test result of the Resnet 50 fine-tuned model gives a good result with epoch 5 ;(Training

accuracy 97.6%) and at epoch 10 (Training accuracy 97.6%).In addition to this, the loss and validation loss in this stage gives good results with epoch 5 as compared to epoch 10 and epoch 15; that is for epoch 5(loss=0.0694) and (val_loss=0.1811) For VGG 19 model it gives a good result with epoch 5(Training accuracy 96.3%) and (Validation accuracy 91%) and again for VGG 19 with epoch 5 (loss=0.1049) and with epoch 5(val_loss=0.2938) as shown before. From this result, the fine-tuned Resnet 50 performs better result as compared to other Dl algorithms in the identification of Coffee leaf and bean disease. This indicates that the ResNet50 fine-tuned model performs a good result as compared to other models while training both models with manually collected data sets of coffee from scratch.

Deep learning architectures require large datasets to train and have many parameters and layers, which makes them computationally expensive. To address this challenge, we propose a model that achieves high accuracy on coffee leaf and bean disease detection and classification with a minimal dataset. Our model uses a convolutional neural network (CNN) algorithm, which is well-suited for image data processing. The model achieves 97.6% accuracy on the training set, 94.4% accuracy on the validation set, and 95% accuracy on the test set.

Finally, we discuss how we answer our research question.

RQ1. We developed a ResNet 50 model that classifies coffee leaf and bean diseases with high training and testing accuracy (**97.6%** and **95%,** respectively). This is a promising result for coffee leaf and bean disease detection, as it suggests that the model can be used to accurately identify these diseases in real-world. The model also achieves high confidence scores for its predictions, which further supports its reliability.

RQ2. Data augmentation is a powerful technique that can be used to improve the performance of machine learning models on image classification. By carefully selecting the right techniques and parameters, you can create a more diverse and robust training dataset, which can lead to better model performance. The most effective data augmentation technique for model performance depends on the dataset and specific task. For this thesis, we try common data augmentation techniques like **Geometric augmentation techniques**, such as flipping, cropping, shearing, Zooming and rotating images, which are often the most effective at improving model performance. These techniques help our models to become more robust to variations in object

orientation, pose, and scale. The study implements augmentation using the Keras Image-Data Generator. We also tried Data Augmentation using GAN (Generative adversarial network) but we didn't get the data we wanted because this algorithm needs a huge amount of data to train and generate fake data.

It is important to note that data augmentation should be used carefully. Too much augmentation can lead to overfitting, where the model learns the training data too well and is unable to generalize to new data. It is important to experiment with different augmentation techniques and levels to find the best combination for your task and dataset.

RQ3. To answer this question, we use a grid search cross-validation (CV) algorithm to evaluate different combinations of hyperparameter values. We can vary our parameters and use the grid search CV algorithm to obtain a set of hyperparameters for our hyperparameter tuning.

Grid search CV is a hyperparameter tuning algorithm that exhaustively searches through a grid of hyperparameter values to find the best combination of values for a given model. It works by first creating a grid of all possible combinations of hyperparameter values. Then, it trains the model on each combination of values and evaluates the model's performance on a held-out validation set. The combination of hyperparameter values that produces the best performance on the validation set is then selected as the best combination of hyperparameter values for the model. Grid search CV is a powerful tool for hyperparameter tuning, but it can be computationally expensive, especially for models with many hyperparameters. However, there are a number of techniques that can be used to make grid search CV more efficient, such as using early stopping.

Number of epochs: We train our model for as many epochs as needed until it achieves good performance on a validation set. However, you should stop training before the model overfits the training data. In our experiments with a ResNet 50 model, we varied the number of epochs from 5, 10, 15, and 20 using a grid search cv algorithm. We found that the model achieved the best accuracy on the validation set after 5 epochs.

Batch size: In our experiments, we varied the batch size from 16, 32, 64, and 128. We found that the model achieved the best accuracy on the 64-batch size.

Optimizer: The optimizer controls how the model updates its parameters. There are many different optimizers available, each with its own advantages and disadvantages. Some popular optimizers we use include Adam, SGD, and RMSprop. We get good accuracy with Adam and RMSprop optimizer.

Learning rate: The learning rate controls how quickly the model updates its parameters.

RQ4. Transfer learning is a deep learning technique where a model developed for one task is reused as the starting point for a model on a second task. This can be particularly useful for image-based coffee leaf and bean disease detection, as it can save a lot of time and effort in training a new model from scratch. To use transfer learning for image-based coffee leaf and bean disease detection, we will need to start with a pre-trained model that has been trained on a large dataset of images, such as ImageNet. This model will have already learned to extract generic features from images, such as edges, shapes, and colors. Then we fine-tune this pre-trained model on our specific dataset of images to learn to classify the specific objects or scenes in our images.

Fine-tuning typically involves training only the last few layers of the pre-trained model. This is because the early layers of the model are responsible for extracting generic features, which are likely to be useful for both the original task and the new task. The last few layers of the model are responsible for learning the specific features of the objects in the dataset.

ResNet50: ResNet50 is a deep convolutional neural network (CNN) that was trained on the ImageNet dataset, a large dataset of images from various categories. ResNet50 has been shown to be effective for a variety of image classification tasks, including coffee leaf and bean disease detection.

4.9 Prototype of the System

Model deployment is the process of making a machine-learning model available for use in a production environment. This involves integrating the model into the existing infrastructure and making it accessible to users who need to make predictions or decisions based on the model's output. It is one of the last stages in the machine learning life cycle and can be one of the heaviest. To start using a model for practical prediction and get a recommendation, it needs to be

effectively deployed into production. For the model deployment, we used the Flask webserver. For the model development, we used Google Collab, after we trained the data, we converted the model to a Keras.h5 model to create a web app to identify Coffee leaf and bean disease.

Figure 4.16 Prototype of the proposed model

CHAPTER FIVE

5 CONCLUSION AND RECOMMENDATION

5.1 Conclusion

The goal of this research was to create a system for identifying coffee plant diseases using transfer learning and image processing techniques. Because coffee is one of the agricultural products that earns the country's foreign currency, aiding the production and processing of coffee with new technology is critical for raising productivity and strengthening the country's economy. As a result, we created and tested deep learning (DL) models that diagnose coffee plant diseases using images of the plan. Finally, the created models were loaded on an easy-to-use flask app, allowing farmers to utilize it to obtain precise information about their plant, allowing them to take suitable steps and get recommendation.

We first conducted a literature survey on the application of deep learning (DL) in the agricultural domain. Our survey indicated that only a few research studies have used DL, even though it has shown promising results in other areas of application. One of the challenges of using DL for real-world applications is the need for a very large amount of data to successfully train the model. The main purpose of this research work is to develop an automated classification system for coffee leaf and bean diseases through image processing and transfer learning techniques. Transfer learning is a machine learning technique where a model trained on a large dataset is reused as the starting point for a model on a smaller dataset. This can be helpful when there is not enough data to train a model from scratch.

Our work is composed of four main components: Data collection, preprocessing, feature extraction, and identification. We collect up to 2000 data from the coffee production area. The preprocessing component reduces the size of input images of coffee leaves/beans, converts the images into arrays using NumPy, and removes background features. We extract features using a deep learning algorithm because one advantage of deep learning is that it can extract features automatically, instead of manually extracting features and then giving them to the algorithm.

The feature extraction component extracts features from the images that are relevant to the classification task**.** The identification component includes prediction and recommendation**.** For prediction, a designed model predicts which type of disease is affecting the coffee and recommends what the farmer can do after the disease is detected. We collected 2,000 sample images for all diseases and health conditions. These images were collected from the Wondoganat Agricultural Research Awada branch in Yirgalem. After we collected all the images, we augmented them using a variety of techniques, such as flipping, rotating, cropping, and adding noise. This process allowed us to create 5,010 new images, which we used for training and testing purposes.

We trained three modified pre-trained models (VGG19, ResNet50, and Inception v3) and a CNN model from scratch. All four models achieved good classification results. The ResNet50 model achieved the best results, with a training accuracy of 97.6% and a testing accuracy of 95%. The VGG19 model achieved a training accuracy of 96.3% and a testing accuracy of 92%. The proposed CNN model achieved a training accuracy of 94% and a testing accuracy of 89.3%. We compared the performance of the proposed CNN model with the transfer learning classifiers. The results show that the ResNet50 model outperformed the other models, with an overall classification accuracy of 95% for coffee leaf and bean diseases. The true positive rates for the classes coffee berry disease, coffee leaf rust, brown eye spot, phoma leaf spot, and healthy were 100%, 97%, 84%, 95%, and 99%, respectively. We evaluated our system on a dataset of coffee leaf and bean images. The results showed that our system achieved an accuracy of 95%. This suggests that our system is a promising approach for the automated classification of coffee leaf and bean diseases.

5.2 Recommendations

Ethiopia's economy is heavily reliant on agriculture, and coffee plant diseases can have a devastating impact on productivity and livelihoods. Image analysis techniques are a promising tool for early detection and classification of coffee plant diseases. While some research has been conducted on coffee leaf diseases in Ethiopia, more research is needed to develop image analysis techniques for detecting coffee diseases in coffee beans and other parts of the coffee plant. This

thesis aims to fill this gap by developing a deep learning model for coffee disease detection and classification using images of coffee leaves and beans. The proposed model achieves high accuracy, suggesting that it is a promising approach for developing automated disease detection systems for coffee farmers and researchers.

For the future, the study recommended the following three recommendations.

- In this study, features are obtained from the coffee plant's leaf and berry portions. However, the stem and root part of the plant also shows symptoms of other threat and diseases of the coffee plant. Obtaining features from the stem and root of the coffee plant presents a challenge, thus a potential future work by collecting data during times of year when the diseases are actively displaying symptoms.
- Transfer learning improved the classification performance of the coffee plant disease identification model with a minimal training dataset. However, the model can only detect a single disease in an image and fails when multiple diseases are present in a single leaf. This is a potential area for future research.
- The coffee plant dataset was collected at only Wondoganat Agricultural Research Center. This means that the dataset does not include images of leaves and beans from other regions of the country, which may have different physical characteristics depending on the area where they were grown. We augmented our dataset using the Keras image data generator. However, the application of GANs (generative adversarial networks) for data augmentation and classification should also be investigated in the future.

REFERENCES

[1] B. Minten, S. Tamru, T. Kuma, and Y. Nyarko, "Structure and performance of Ethiopia ' s coffee export sector," *Int. food policy Res. Inst.*, no. June, pp. 2–3, 2014.

[2] R. Cerda, J. Avelino, C. Gary, P. Tixier, E. Lechevallier, and C. Allinne, "Primary and secondary yield losses caused by pests and diseases: Assessment and modeling in coffee," *PLoS One*, vol. 12, no. 1, pp. 1–17, 2017, doi: 10.1371/journal.pone.0169133.

[3] L. X. Boa Sorte, C. T. Ferraz, F. Fambrini, R. D. R. Goulart, and J. H. Saito, "Coffee leaf disease recognition based on deep learning and texture attributes," *Procedia Comput. Sci.*, vol. 159, pp. 135–144, 2019, doi: 10.1016/j.procs.2019.09.168.

[4] D. Teferi, "Status of Major Coffee Diseases of Coffea Arabica L . in Afromontane Rainforests of Ethiopia . A Review," *Food Sci. Qual. Manag.*, vol. 76, pp. 35–40, 2018.

[5] M. A. Rutherford and N. Phiri, "Pests and Diseases of Coffee in Eastern Africa : A Technical and Advisory Manual," pp. 1–70, 2006.

[6] H. Hindorf and C. O. Omondi, "A review of three major fungal diseases of Coffea arabica L. in the rainforests of Ethiopia and progress in breeding for resistance in Kenya," *J. Adv. Res.*, vol. 2, no. 2, pp. 109–120, 2011, doi: 10.1016/j.jare.2010.08.006.

[7] G. C. Khadabadi, V. S. Rajpurohit, A. Kumar, and V. B. N. Nargund, "Disease Detection in Vegetables Using Image Processing Techniques : A Review," *Int. J. Emerg. Technol. Comput. Sci. Electron.*, vol. 14, no. 2, pp. 954–960, 2015.

[8] D. Novtahaning, H. A. Shah, and J. M. Kang, "Deep Learning Ensemble-Based Automated and High-Performing Recognition of Coffee Leaf Disease," *Agric.*, vol. 12, no. 11, 2022, doi: 10.3390/agriculture12111909.

[9] F. Martínez, H. Montiel, and F. Martínez, "A Machine Learning Model for the Diagnosis of Coffee Diseases," *Int. J. Adv. Comput. Sci. Appl.*, vol. 13, no. 4, pp. 968–974, 2022, doi: 10.14569/IJACSA.2022.01304110.

[10] K. Belachew, "STATUS OF COFFEE DISEASES IN ETHIOPIA AND THEIR," in *" Ethiopia Management Of Coffee Leaf Rust By Combination Of Fertilizers And Supplementary Irrigation*, Togo, Lome, 2022, pp. 1–12.

[11] K. Alemu, G. Adugna, F. Lemessa, and D. Muleta, "Induction of systemic resistance in Arabica coffee (Coffea arabica L.) against coffee berry disease (Colletotrichum kahawae Waller & Bridge) mediated through plant defense activator," *Int. J. Pest Manag.*, vol. 65, no. 4, pp. 313–323, 2019, doi: 10.1080/09670874.2018.1506190.

[12] E. B. - and D. R. -, "Coffee Leaf Diseases Classification and the Effect of Fine-tuning on Deep Convolutional Neural Networks," *Int. J. Multidiscip. Res.*, vol. 4, no. 5, pp. 1–13, 2022, doi: 10.36948/ijfmr.2022.v04i05.861.

[13] W. Haymanot Taddese, "Coffee Disease Detection Using Convolutional Neural Network:

an Image Processing Approach," *Addis Ababa Univ. Institutional Repos.*, no. Thesis, pp. 1–89, 2021.

[14] A. K. Yimeru, "Automatic Coffee Disease and Pest Damage Identification," *Addis Ababa Univ. Institutional Repos.*, no. Thesis, pp. 1–87, 2017.

[15] E. B. Paulos and M. M. Woldeyohannis, "Detection and Classification of Coffee Leaf Disease using Deep Learning," *2022 Int. Conf. Inf. Commun. Technol. Dev. Africa, ICT4DA 2022*, no. November, pp. 1–6, 2022, doi: 10.1109/ICT4DA56482.2022.9971300.

[16] A. D. Mengistu, D. M. Alemayehu, and S. G. Mengistu, "Ethiopian Coffee Plant Diseases Recognition Based on Imaging and Machine Learning Techniques," *Int. J. Database Theory Appl.*, vol. 9, no. 4, pp. 79–88, 2016, doi: 10.14257/ijdta.2016.9.4.07.

[17] M. B. and R. J. H. J.M. Waller, *COFFEE PESTS, DISEASES AND THEIR MANAGEMENT*. CABI, 2007.

[18] K. Belachew, Teferi Demelash, N. Hundessa, and S. Tesfaye, "The Statue and Management of Coffee Wilt Disease (Gibberella xylarioides) in Ethiopian Coffee Production," *J. Nat. Sci. Res.* , vol. 6, no. 5, pp. 16–21, 2016, [Online]. Available: https://pdfs.semanticscholar.org/9bf3/2ee5d4884ac6384091020bfb4d878ca82f6c.pdf

[19] M. Kumar, P. Gupta, P. Madhav, and Sachin, "Disease detection in coffee plants using convolutional neural network," *Proc. 5th Int. Conf. Commun. Electron. Syst. ICCES 2020*, no. Icces, pp. 755–760, 2020, doi: 10.1109/ICCES48766.2020.09138000.

[20] et al. 201. Moat, "Coffee Farming and Climate Change in Ethiopia: Impacts, Forecasts, Resilience and Opportunities. - Summary," *R. Bot. Gard. Kew*, p. 37, 2017, [Online]. Available: https://www.kew.org/sites/default/files/Coffee Farming and Climate Change in Ethiopia.pdf

[21] A. K. Jain, "Fundamentals of Digital Image Processing, ISBN:978-0-13-336165-0." pp. 1–565, 1989.

[22] D. T. and K. Belachew, "A Review of Coffee Diseases Research in Ethiopia," *Int. J. Agric. Biosci.*, vol. 8, no. 2, pp. 89–98, 2013.

[23] J. M. M. E.K. Kathurima, G. Aluora, "IDENTIFICATION AND MANAGEMENT OF COFFEE BERRY DISEASE (CBD) AND COFFEE LEAF RUST (CLR)".

[24] A. de M. Catarino, E. A. Pozza, A. A. A. Pozza, L. S. dias Santos, G. B. Vasco, and P. E. de Souza, "Calcium and potassium contents in nutrient solution on Phoma leaf spot intensity in coffee seedlings," *Rev. Ceres*, vol. 63, no. 4, pp. 486–491, 2016, doi: 10.1590/0034-737X201663040008.

[25] M. Ziyad, "Artificial Intelligence Definition, Ethics and Standards," *Artif. Intell. Defin. Ethics Stand.*, pp. 1–11, 2019.

[26] P. Xu, "Review on Studies of Machine Learning Algorithms," *J. Phys. Conf. Ser.*, vol. 1187, no. 5, 2019, doi: 10.1088/1742-6596/1187/5/052103.

[27] R. C. Gonzalez and R. E. Woods, *4TH EDITION Digital image processing*. 2018.

[28] W. K. Pratt, “Digital Image Processing,” *Eur. J. Eng. Educ.*, vol. 19, no. 3, p. 377, 1994, doi: 10.1080/03043799408928319.

[29] I. G. and Y. B. and A. Courville, *Deep learning 简介 ー、 什么是 Deep Learning ？*, vol. 29, no. 7553. 2016. [Online]. Available: http://deeplearning.net/

[30] R. S. Dwivedi, *Remote sensing of soils*, no. 2. 2017. doi: 10.1007/978-3-662-53740-4.

[31] A. Kamilaris and F. X. Prenafeta-Boldú, “Deep learning in agriculture: A survey,” *Comput. Electron. Agric.*, vol. 147, pp. 70–90, 2018, doi: 10.1016/j.compag.2018.02.016.

[32] D. Soydaner, “A Comparison of Optimization Algorithms for Deep Learning,” *Int. J. Pattern Recognit. Artif. Intell.*, vol. 34, no. 13, 2020, doi: 10.1142/S0218001420520138.

[33] B. B. Traore, B. Kamsu-Foguem, and F. Tangara, “Deep convolution neural network for image recognition,” *Ecol. Inform.*, vol. 48, pp. 257–268, 2018, doi: 10.1016/j.ecoinf.2018.10.002.

[34] S. S. Cross, R. F. Harrison, and R. L. Kennedy, *Introduction to neural networks*, vol. 346, no. 8982. 1995. doi: 10.1016/S0140-6736(95)91746-2.

[35] A. Khan, A. Sohail, U. Zahoora, and A. S. Qureshi, “A survey of the recent architectures of deep convolutional neural networks,” *Artif. Intell. Rev.*, vol. 53, no. 8, pp. 5455–5516, 2020, doi: 10.1007/s10462-020-09825-6.

[36] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018-Janua, no. August, pp. 1–6, 2018, doi: 10.1109/ICEngTechnol.2017.8308186.

[37] 3 & Richard Kinh Gian Do2 & Kaori Togashi1 Rikiya Yamashita1, 2 & Mizuho Nishio1 and Received:, “Convolutional neural networks: an overview and application in radiology,” *nsights Imaging*, pp. 611–629, 2018, doi: 10.1007/978-981-15-7078-0_3.

[38] M. Buscema, “Back propagation neural networks,” *Subst. Use Misuse*, vol. 33, no. 2, pp. 233–270, 1998, doi: 10.3109/10826089809115863.

[39] S. Bansari, “Introduction to how CNNs Work,” 2019.

[40] L. Alzubaidi *et al.*, “Review of deep learning: concepts, CNN architectures, challenges, applications, future directions,” *J. Big Data*, no. 1, pp. 2–74, 2021, doi: 10.1186/s40537-021-00444-8.

[41] I. Verma, U. Marhatta, S. Sharma, and V. Kumar, “Age Prediction using Image Dataset using Machine Learning,” *Int. J. Innov. Technol. Explor. Eng.*, vol. 8, no. 12s3, pp. 107–113, 2020, doi: 10.35940/ijitee.l1020.10812s319.

[42] Raúl Gómez, “Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names,” 2019. https://gombru.github.io/2018/05/23/cross_entropy_loss/

[43] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” 2012, [Online]. Available: http://arxiv.org/abs/1212.5701

[44] L. Liu *et al.*, “On the Variance of the Adaptive Learning Rate and Beyond,” pp. 1–14, 2019, [Online]. Available: http://arxiv.org/abs/1908.03265

[45] Z. Yao, Y. Cao, S. Zheng, G. Huang, and S. Lin, “Cross-Iteration Batch Normalization,” 2021.

[46] S. Cai, Y. Shu, G. Chen, B. C. Ooi, W. Wang, and M. Zhang, “Effective and Efficient Dropout for Deep Convolutional Neural Networks,” *IEEE*, pp. 1–12, Apr. 2019.

[47] G. E. H. Alex Krizhevsky, Ilya Sutskever, “ImageNet Classification with Deep Convolutional Neural Networks”, doi: 10.1201/9781420010749.

[48] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 770–778, 2016, doi: 10.1109/CVPR.2016.90.

[49] C. Szegedy *et al.*, “Going deeper with convolutions,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 07-12-June, pp. 1–9, 2015, doi: 10.1109/CVPR.2015.7298594.

[50] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” *3rd Int. Conf. Learn. Represent. ICLR 2015 - Conf. Track Proc.*, pp. 1–14, 2015.

[51] A. Afifi, A. Alhumam, and A. Abdelwahab, “Convolutional neural network for automatic identification of plant diseases with limited data,” *Plants*, vol. 10, no. 1, pp. 1–16, 2021, doi: 10.3390/plants10010028.

[52] Z. Zheng, L. Zheng, and Y. Yang, “Unlabeled Samples Generated by GAN Improve the Person Re-identification Baseline in Vitro,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 3774–3782, 2017, doi: 10.1109/ICCV.2017.405.

[53] C. Shorten and T. M. Khoshgoftaar, “A survey on Image Data Augmentation for Deep Learning,” *J. Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.

[54] I. Goodfellow *et al.*, “Generative adversarial networks,” *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2014, doi: 10.1145/3422622.

[55] S. Wallelign, “An Intelligent System for Coffee Grading and Disease To cite this version : HAL Id : tel-02506162 An Intelligent System for Coffee Grading and Disease Identification,” vol. 2, p. 135, 2020.

[56] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” *Front. Plant Sci.*, vol. 7, no. September, 2016, doi: 10.3389/fpls.2016.01419.

[57] K. J. Danjuma, “Performance Evaluation of Machine Learning Algorithms in Post-operative Life Expectancy in the Lung Cancer Patients,” 2015, [Online]. Available: http://arxiv.org/abs/1504.04646

APPENDIX

Resnet 50 Sample Code

```
[ ] #COFFEE LEAF AND BEAN DISEASE DETECTION USING TRANSFER LEARNING
    from google.colab import drive
    drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# Import Necessary Library
import os
import numpy as np
import pandas as pd
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam, SGD,RMSprop, Adadelta, Nadam
from tensorflow.keras.regularizers import l2
from tensorflow.keras.callbacks import ReduceLROnPlateau, EarlyStopping, ModelCheckpoint
from tensorflow import keras
from keras import Sequential
from keras.layers import Activation, BatchNormalization, Conv2D, Dense, Dropout, Flatten, MaxPooling2D, GlobalAveragePooling2D
from keras.applications import ResNet50
from sklearn.metrics import confusion_matrix, classification_report
import random
import cv2 as cv
import matplotlib.pyplot as plt
import tensorflow.keras as keras
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img, img_to_array
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential, Model
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input
from matplotlib.pyplot import imread
```

```
#Load data
train_path = '/content/drive/MyDrive/Coffee Diseaese Dataset/train/'
val_path = '/content/drive/MyDrive/Coffee Diseaese Dataset/val/'
test_path = '/content/drive/MyDrive/Coffee Diseaese Dataset/test/'
train_folders = sorted(os.listdir(train_path))
val_folders = sorted(os.listdir(val_path))
test_folders = sorted(os.listdir(test_path))
print('Training classes: {}'.format(train_folders))
print('Validation classes: {}'.format(val_folders))
print('Testing classes: {}'.format(test_folders))
```

```
Training classes: ['Brown Eye Spot', 'Coffee Berry Disease', 'Coffee Leaf Rust', 'Healthy', 'Phoma Leaf Spot']
Validation classes: ['Brown Eye Spot', 'Coffee Berry Disease', 'Coffee Leaf Rust', 'Healthy', 'Phoma Leaf Spot']
Testing classes: ['Brown Eye Spot', 'Coffee Berry Disease', 'Coffee Leaf Rust', 'Healthy', 'Phoma Leaf Spot']
```

```
datagen = ImageDataGenerator()
# training data
train_generator = datagen.flow_from_directory(
    directory="/content/drive/MyDrive/Coffee Diseaese Dataset/train",
    target_size=(224, 224),
    batch_size=64,
    class_mode="categorical",
)
# validation data
valid_generator = datagen.flow_from_directory(
    directory="/content/drive/MyDrive/Coffee Diseaese Dataset/val",
    target_size=(224, 224),
    batch_size=64,
    class_mode="categorical",
)
# test data
test_generator = datagen.flow_from_directory(
    directory="/content/drive/MyDrive/Coffee Diseaese Dataset/test",
    target_size=(224, 224),
    batch_size=64,
    class_mode="categorical",
)
```

```
Found 4010 images belonging to 5 classes.
Found 500 images belonging to 5 classes.
Found 500 images belonging to 5 classes.
```

```
[ ] Resnet_50=tf.keras.applications.resnet50.ResNet50
    model_arch=Resnet_50()
    model_arch.summary()
```

```
Model: "resnet50"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_4 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 230, 230, 3)  0           ['input_4[0][0]']

 conv1_conv (Conv2D)            (None, 112, 112, 64  9472        ['conv1_pad[0][0]']
                                )

 conv1_bn (BatchNormalization)  (None, 112, 112, 64  256         ['conv1_conv[0][0]']
                                )

 conv1_relu (Activation)        (None, 112, 112, 64  0           ['conv1_bn[0][0]']
                                )

 pool1_pad (ZeroPadding2D)      (None, 114, 114, 64  0           ['conv1_relu[0][0]']
                                )

 pool1_pool (MaxPooling2D)      (None, 56, 56, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_1_conv (Conv2D)   (None, 56, 56, 64)   4160        ['pool1_pool[0][0]']
```

load the model

```
[ ] # ResNet50 model
    resnet_50 = ResNet50(include_top=False, weights='imagenet', input_shape=(224,224,3))
    for layer in resnet_50.layers:
        layer.trainable = False
```

```
# build the entire model
x = resnet_50.output
x = layers.GlobalAveragePooling2D()(x)
x = layers.BatchNormalization()(x)
x = layers.Dense(512, activation='relu')(x)
x = layers.Dropout(0.001)(x)
x = layers.Dense(256, activation='relu')(x)
x = layers.Dropout(0.001)(x)
x = layers.Dense(128, activation='relu')(x)
x = layers.Dropout(0.001)(x)
x = layers.Dense(64, activation='relu')(x)
x = layers.Dropout(0.001)(x)
predictions = layers.Dense(5, activation='softmax')(x)
model = Model(inputs = resnet_50.input, outputs = predictions)
```

```python
#Early stopping and model check point
from keras.callbacks import ModelCheckpoint, EarlyStopping

#Early Stopping
es  = EarlyStopping(monitor='val_accuracy',
                    min_delta=0.01,
                    patience= 3,
                    verbose=1)
#ModelCheckpoint
mc= ModelCheckpoint(filepath="best_m.h5",
                    monitor='accuracy',
                    min_delta=0.01,
                    patience= 3,
                    verbose=1 ,
                   save_best_only=True)
cb=[es, mc]
```

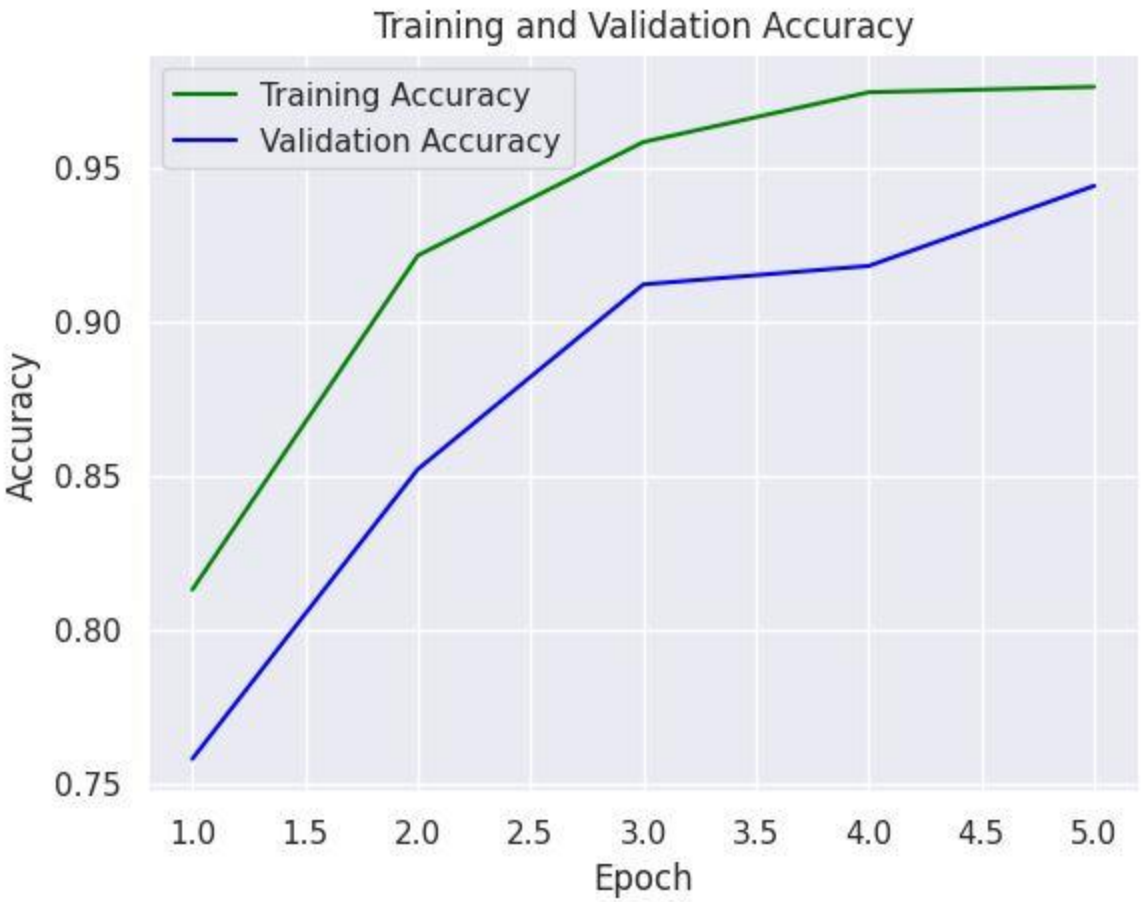
```python
# define training function
def trainModel(model, epochs, optimizer, batch_size, callbacks):
    model.compile(optimizer=optimizer, loss="categorical_crossentropy", metrics=["accuracy"])
    return model.fit(train_generator, validation_data=valid_generator, epochs=epochs, batch_size=batch_size, callbacks=cb)
```

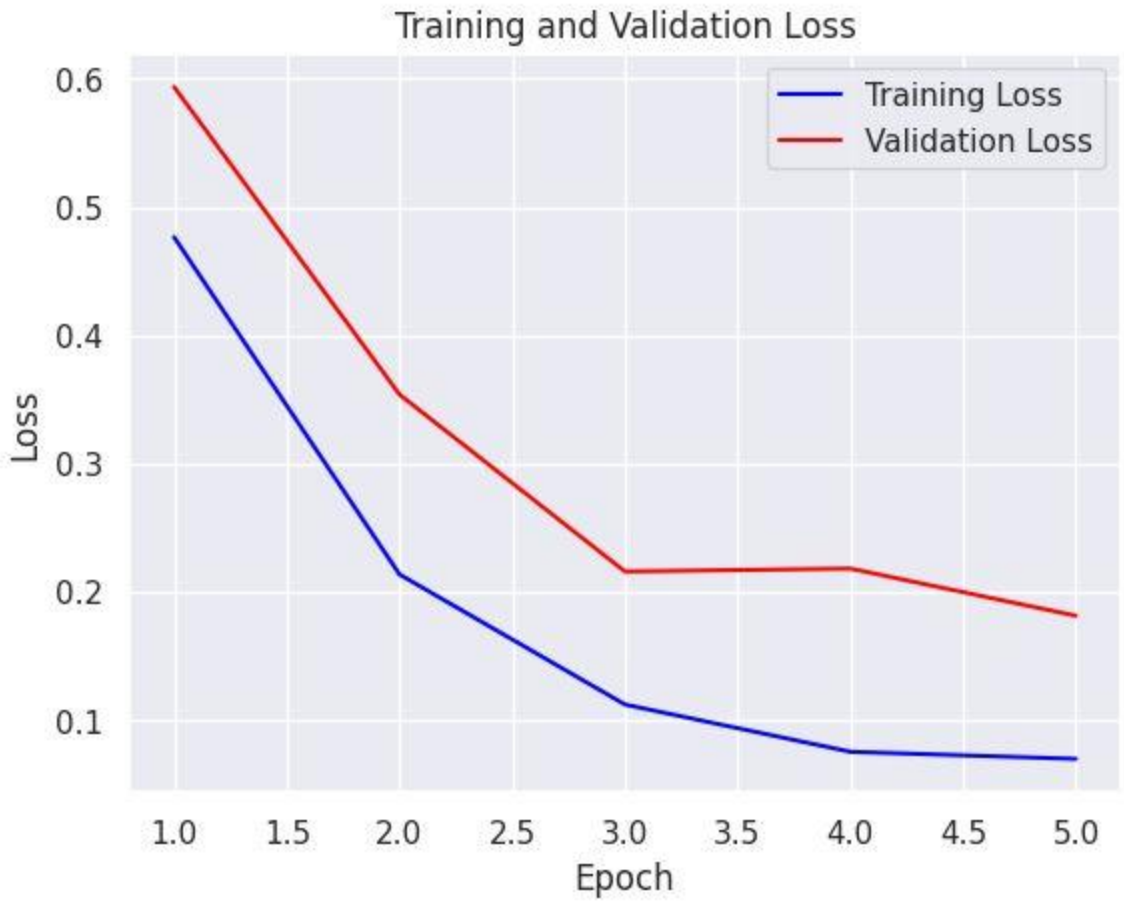
```python
model_history = trainModel(model = model, epochs = 5, optimizer = 'Adam', batch_size = 64, callbacks=cb)
```

```
Epoch 1/5
63/63 [==============================] - ETA: 0s - loss: 0.4766 - accuracy: 0.8129
Epoch 1: accuracy improved from -inf to 0.81294, saving model to best_m.h5
63/63 [==============================] - 418s 6s/step - loss: 0.4766 - accuracy: 0.8129 - val_loss: 0.5938 - val_accuracy: 0.7580
Epoch 2/5
63/63 [==============================] - ETA: 0s - loss: 0.2134 - accuracy: 0.9214
Epoch 2: accuracy improved from 0.81294 to 0.92139, saving model to best_m.h5
63/63 [==============================] - 25s 403ms/step - loss: 0.2134 - accuracy: 0.9214 - val_loss: 0.3537 - val_accuracy: 0.8520
Epoch 3/5
63/63 [==============================] - ETA: 0s - loss: 0.1118 - accuracy: 0.9582
Epoch 3: accuracy improved from 0.92139 to 0.95821, saving model to best_m.h5
63/63 [==============================] - 25s 403ms/step - loss: 0.1118 - accuracy: 0.9582 - val_loss: 0.2157 - val_accuracy: 0.9120
Epoch 4/5
63/63 [==============================] - ETA: 0s - loss: 0.0750 - accuracy: 0.9744
Epoch 4: accuracy improved from 0.95821 to 0.97438, saving model to best_m.h5
63/63 [==============================] - 26s 407ms/step - loss: 0.0750 - accuracy: 0.9744 - val_loss: 0.2179 - val_accuracy: 0.9180
Epoch 5/5
63/63 [==============================] - ETA: 0s - loss: 0.0694 - accuracy: 0.9761
Epoch 5: accuracy improved from 0.97438 to 0.97612, saving model to best_m.h5
63/63 [==============================] - 26s 409ms/step - loss: 0.0694 - accuracy: 0.9761 - val_loss: 0.1811 - val_accuracy: 0.9440
```

```
#VISUALIZING THE RESULTS
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()

acc = model_history.history['accuracy']
val_acc = model_history.history['val_accuracy']
loss = model_history.history['loss']
val_loss = model_history.history['val_loss']
epochs = range(1, len(loss) + 1)
```

```
#accuracy plot
plt.plot(epochs, acc, color='green', label='Training Accuracy')
plt.plot(epochs, val_acc, color='blue', label='Validation Accuracy')
plt.title('Training and Validation Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend()

plt.figure()
#loss plot
plt.plot(epochs, loss, color='blue', label='Training Loss')
plt.plot(epochs, val_loss, color='red', label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()

plt.show()
```


Training and Validation Accuracy
Training Accuracy
Validation Accuracy
Accuracy
0.95
0.90
0.85
0.80
0.75
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Epoch




```
test_loss, test_acc = model.evaluate(test_generator)
acc = model.evaluate(valid_generator)[1]
acc1 = model.evaluate(train_generator)[1]
print(f"Train Accuracy of your model is  = {acc1*100}%")
print("Test Accuracy of your model is  : ", test_acc*100)
print(f"Validation accuracy of your model is ={acc*100}%")
print("The test loss is: ", test_loss)
```

```
8/8 [==============================] - 3s 318ms/step - loss: 0.2490 - accuracy: 0.9200
8/8 [==============================] - 3s 317ms/step - loss: 0.1811 - accuracy: 0.9440
63/63 [==============================] - 22s 349ms/step - loss: 0.0408 - accuracy: 0.9863
Train Accuracy of your model is  = 98.63184094429016%
Test Accuracy of your model is  :  92.00000166893005
Validation accuracy of your model is =94.40000057220459%
The test loss is:  0.24896560609340668
```

```
# save it as a h5 file
from tensorflow.keras.models import load_model
model.save('/content/drive/MyDrive/Coffee Diseaese Dataset/Saved_Model/Resnet50.h5')
```

```
[ ]  ref = dict(zip(list(train_generator.class_indices.values()) , list(train_generator.class_indices.keys())))
```

```
def prediction(path):
  img= load_img(path, target_size=(224,224))
  i=img_to_array(img)
  im=preprocess_input(i)
  img= np.expand_dims(im, axis=0)
  pred = np.argmax(model.predict(img))
  print(f"The disease belongs to {ref[pred]}")
```

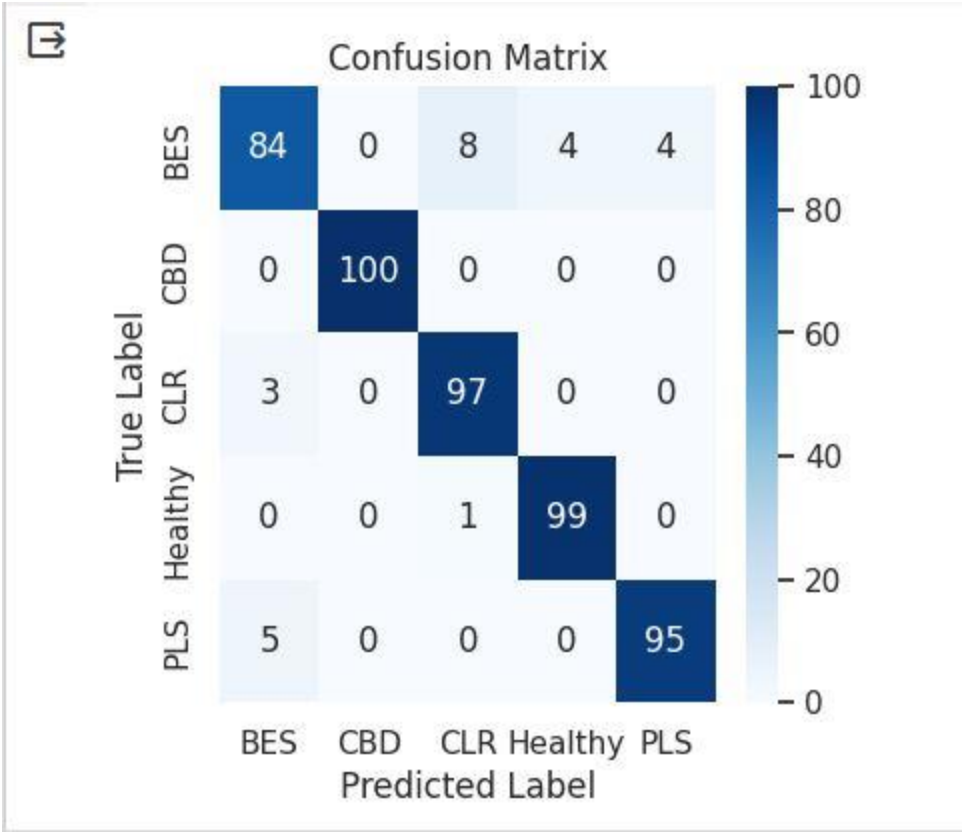
```
path='/content/drive/MyDrive/Coffee Diseaese Dataset/test/Brown Eye Spot/104.PNG'

prediction(path)
```

```
1/1 [==============================] - 2s 2s/step
The disease belongs to Brown Eye Spot
```

```
Y_pred = model.predict(test_generator)
```

```
8/8 [==============================] - 3s 422ms/step
```

```
y_pred = np.argmax(Y_pred, axis=1)
confusion_mtx = confusion_matrix(test_generator.classes, y_pred)
f,ax = plt.subplots(figsize=(4, 4))
sns.heatmap(confusion_mtx,cmap='Blues', annot=True, fmt='d')
sns.color_palette("rocket", as_cmap=True)

plt.xlabel("Predicted Label")
plt.ylabel("True Label")
ax.xaxis.set_ticklabels(test_generator.class_indices)
ax.yaxis.set_ticklabels(test_generator)
plt.title("Confusion Matrix")
plt.show()
```



```
y_true = test_generator.classes
Y_pred = model.predict(test_generator)
f1 = f1_score(y_true, y_pred, average='macro')
print(classification_report(y_true, y_pred, target_names=test_generator.class_indices.keys()))
```

```
8/8 [==============================] - 3s 391ms/step
              precision    recall  f1-score   support

         BES       0.91      0.84      0.87       100
         CBD       1.00      1.00      1.00       100
         CLR       0.92      0.97      0.94       100
     Healthy       0.96      0.99      0.98       100
         PLS       0.96      0.95      0.95       100

    accuracy                           0.95       500
   macro avg       0.95      0.95      0.95       500
weighted avg       0.95      0.95      0.95       500
```